

Constructionism

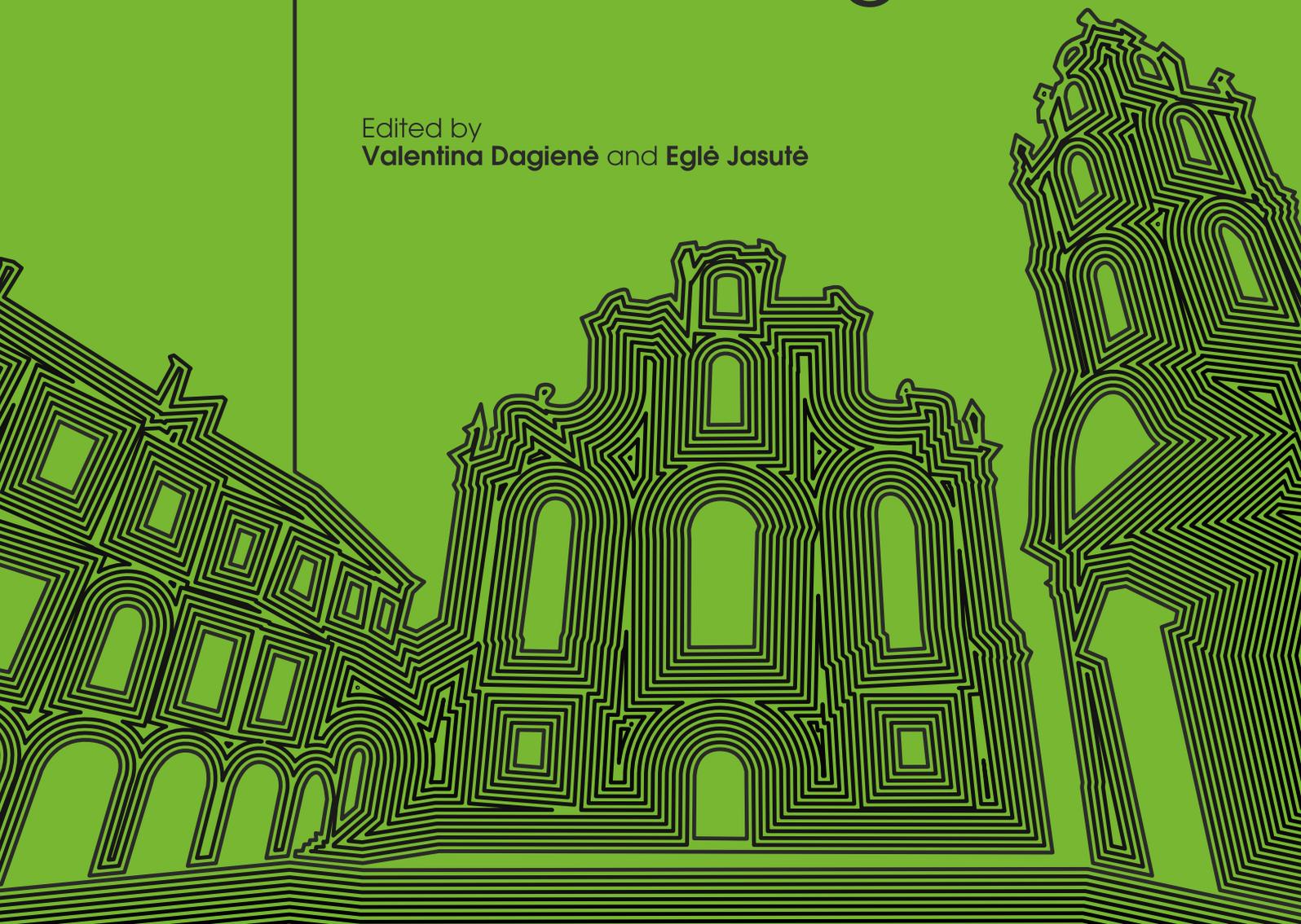
2018, Vilnius

Constructionism, Computational Thinking
and Educational Innovation

August 20-25, Vilnius, Lithuania

Conference Proceedings

Edited by
Valentina Dagienė and **Eglė Jasutė**



Constructionism 2018 Conference

Constructionism 2018

Constructionism, Computational Thinking and
Educational Innovation: conference proceedings

Organizers:

Vilnius University
Faculty of Philosophy and
Institute of Data Science and Digital Technologies
in cooperation with Lithuanian Computer Society

August 20-25, Vilnius, Lithuania

Edited by

Valentina Dagienė and Eglė Jasutė

ISBN 978-609-95760-1-5

The conference is partially supported by Research Council of Lithuania

All publications are copyright © 2018 by the authors unless specified otherwise.
Permission to make digital or hard copies of portions of this work for personal or classroom use is granted provided that the copies are not made or distributed for profit or commercial advantage and that the full citation is included. To copy otherwise, to republish, to post on servers, or to redistribute to lists, articles that are copyright by the author requires a written request to the authors.

CONFERENCE VENUE

Faculty of Philosophy
Vilnius University
Universiteto str. 9
01513 Vilnius, Lithuania
Located at the Old Campus of Vilnius University

The conference website: <http://www.constructionism2018.fsf.vu.lt>

Book of abstracts is available at: <http://www.constructionism2018.fsf.vu.lt/book-of-abstracts>

Full papers are available at: <http://www.constructionism2018.fsf.vu.lt/proceedings>

CONSTRUCTIONISM 2018 COMMITTEES

Conference Chairs

Valentina Dagienė, Vilnius University Institute of Data Science and Digital Technologies, Lithuania
Arūnas Poviliūnas, Vilnius University Faculty of Philosophy, Lithuania
Arnan (Roger) Sipitakiat, Chiang Mai University, Thailand

Scientific Committee / Reviewers

Tim Bell, University of Canterbury, New Zealand
Paulo Blikstein, Stanford University, USA
Pavel Boytchev, Sofia University, Portugal
James Clayson, American University of Paris, France
Secundino Correia, Imagina, Coimbra, Portugal
Barbara Demo, University of Torino, Italy
Vladimiras Dolgopolas, Vilnius University, Lithuania
Michael Eisenberg, University of Washington, USA
Gerald Futschek, Vienna University of Technology, Austria
Carina Girvan, Cardiff University, Cardiff, Wales,
Paul Goldenberg, Education Development Center, USA
Brian Harvey, University of California, USA
Bulgaria
Arthur Hjorth, Northwestern University, USA
Celia Hoyles, University College London Institute of Education, UK
Eglė Jasutė, Vilnius University, Lithuania
Tatjana Jevsikova, Vilnius University, Lithuania
Anita Juškevičienė, Vilnius University, Lithuania
Ivan Kalaš, Comenius University, Slovakia
Eric Klopfer, Massachusetts Institute of Technology, USA
Witold Kranas, OEIiZK, Poland
Chronis Kynigos, National and Kapodistrian University of Athens, Greece
Manolis Mavrikis, University of London, UK
Mattia Monga, Università degli Studi di Milano, Italy

Jens Mönig, SAP, Germany
Erich Neuwirth, University of Vienna, Austria
Richard Noss, University College London Institute of Education, UK
Mareen Przybylla, University of Potsdam, University of Potsdam, Germany
Mitchel Resnick, MIT Media Lab, Massachusetts Institute of Technology, USA
Ralf Romeike, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Ana Isabel Sacristán, Cinvestav, Mexico
Jenny Sendova, Institute of Mathematics, Bulgaria
Giovanni Serafini, ETH Zurich, Switzerland, Italy
Wolfgang Slany, Graz University of Technology Institute of Software Technology, Austria
Gary Stager, Constructing Modern Knowledge, Torrance, USA
Eliza Stefanova, Sofia University "St. Kliment Ohridski", Bulgaria
Gabrielė Stupurienė, Vilnius University, Lithuania
Maciej Sysło, University of Wrocław, Poland
Márta Turcsányi-Szabó, Eötvös Loránd University, Hungary
José Armando Valente, State University of Campinas, UNICAMP, Brazil
Jiří Vaníček, University of South Bohemia, Czechia
Michael Weigend, University of Münster, Germany
Uri Wilensky, Northwestern University, Evanston, USA
Michelle Wilkerson, University of California, USA

Local Organizing Committee

Vida Jakutienė
Aldona Mačiūnienė
Saulius Maskeliūnas
Olga Suprun



Photo E. Kurauskas



Photo E. Kurauskas

Welcome Message

Dear Colleagues

We would like to welcome you to the Constructionism conference.

The Constructionism conference celebrates its fifth anniversary under this name, building on the 27-year tradition of biennial *Eurologo* conferences established by the European Logo community. Logo is the computer programming language for learners in which the constructionist approach was first developed. Seymour Papert, who has coined the term *constructionism*, in order to define its meaning, started from the comparison with the term *constructivism*:

Constructionism shares constructivism's connotation of learning as 'building knowledge structures' irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" (Papert, S. & Harel, I. *Constructionism*. New York: Ablex Publ. Corp., 1991).

The constructionism shares the main idea of genetic epistemology elaborated by Jean Piaget about the immanence of the cognitive development. In Piaget's version, the immanent algorithm of the cognitive development includes the sensorimotor, preoperational, concrete operational, and formal operational stages.

The pioneer of the radical constructivism Ernst von Glasersfeld has enriched the constructivist educational discourse by introducing ideas from the Italian philosopher Giambattista Vico. In his Magnum opus *Scienza Nuova*, Vico has elaborated the principle *verum esse ipsum factum* ("What is true is precisely what is made"). According to Vico, ".../ the world of civil society has certainly been made by men, and that its principles are therefore to be found within the modifications of our own human mind." The world of civil society differs from the world of nature, "which, since God made it, He alone knows" (Vico, J.-B. *The New Science*. Transl. by T. G. Bergin & M. H. Fisch. Ithaca, N Y: Cornell Univ. Press, 1948 [1744]: 85).

Another important idea brought from Vico is the application of tropology for the description of different modes of consciousness or phases of its development. American post-structuralist Hayden White has applied tropes for the explanation of different phases of cognitive development discovered by Piaget. It seems that it is important for the development of constructionism to track the processes in the field of constructivism and vice versa.

Seymour Papert has stressed that the picture of how scientists actually work should be shared with children but not in the way of verbally-expressed formal knowledge. Closer cooperation with the Science

and Technology Studies could assist in identifying more adequate forms how to communicate scientific knowledge and how to engage children into the attractive and actual scientific research.

What have we achieved until today? Has our concept of education in general and computer science (computing or informatics) education in particular changed and its quality improved? Can we use the lessons of the past to prepare for the future? In an increasingly interdependent and complex world, how is technology and informatics changing society and affecting education through the subject areas of humanities, science, mathematics, and arts? The Constructionism conference has addressed these questions by offering experts from across the world the opportunity to exchange ideas and knowledge, and to generate a more informed understanding of the issues of informatics and digital technologies in education. The Conference offered a number of themes:

- Constructionist epistemology
- Informal learning
- Innovative computing education
- Learning from pioneers
- Methodologies, tools and technologies
- Monitoring, evaluation and research
- Outreach and communication
- Progressive education in national and regional contexts
- Social justice, equity and citizenship
- Teacher training and educational policies
- Technology unplugged activities
- Visual and creative arts

The Conference brings together delegates from all over the world to address pressing issues in computing education. In addition to keynote speakers, research and practice papers, panels, posters, demonstrations, and workshops, the Conference provides facilities and exposure for working groups for the first time. The working groups are formed by participants with a common interest in a topic. Participating in a working group provides a unique opportunity to work with people from different countries who are interested and knowledgeable in the area of the working group. It is also one of the best ways to become part of the constructivist community. Seven working groups have been accepted, covering a broad spectrum of topics. Participants present their preliminary results to conference attendees at a special working group presentation session, and submit a final report after the conference concludes.

The Constructionism conference continues to be truly international with about 150 submissions from 40 countries. The accepted submissions consisted of 18 keynote talks, 57 research and practice papers, 3 panels, 7 working group proposals, and 27 proposals for posters, demonstrations and workshops. In addition, a special Teachers' Day is organized before the conference: 13 workshops were proposed for more than 150 Lithuanian teachers.

Selected research papers will be published in the international peer-reviewed journals "Informatics in Education" and "Problemos". After the conference, there will be possibility to extend the best papers and publish them in the international peer-reviewed journal "Constructivist Foundations".

We are grateful to all committee members participating in any way in the conference.

Welcome to Vilnius, the capital of Lithuania! Enjoy the Constructionism conference in the country celebrating 100 years of independence!

Conference Co-Chairs

Valentina Dagienė

Arūnas Poviliūnas

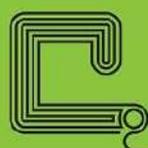


Program at a glance

Monday, August 20		Tuesday August 21 Location: Main Building, The Theatre Hall*	Time	Wednesday August 22 Location: The Faculty of Philosophy**	Thursday August 23 Location: The Faculty of Philosophy**	Friday August 24 Location: The Faculty of Philosophy**	Saturday August 25 Location: The Faculty of Philosophy**
TEACHERS DAY	9 ⁰⁰ -10 ⁰⁰ Registration (in front of the Theatre Hall)	8 ³⁰ -all day Registration	8 ³⁰ -10 ³⁰	In parallel: • Plenary session III • Plenary session IV	In parallel: • Plenary session V • Plenary session VI	In parallel: • Plenary session VII • Plenary session VIII	In parallel: • Plenary session IX • Plenary session X
	10 ⁰⁰ -12 ⁰⁰ Welcome Speakers (Main Building, The Theatre Hall*)	10 ⁰⁰ -11 ³⁰ Excursion to Old Vilnius University I	10 ³⁰ -11 ⁰⁰	In parallel: • Working Group presentations I • Panel discussion II	In parallel: • Poster session I • Poster session II	In parallel: • Paper session 9 • Demo session 1	In parallel: • Demo session 2 • Demo session 3 • Workshop 6
	12 ⁰⁰ -13 ⁰⁰ Lunch (Registration in the Faculty of Philosophy**)	11 ³⁰ -13 ⁰⁰ Excursion to Old Vilnius University II	11 ⁰⁰ -11 ³⁰	Coffee break	Coffee break	Coffee break	11 ³⁰ -12 ⁰⁰ Closing: Farewell buffet Location: Grand Courtyard next to the Main Building*
	13 ⁰⁰ -15 ⁰⁰ Workshops in parallel (The Faculty of Philosophy**)		13 ⁰⁰ -13 ³⁰	In parallel: • Paper session 1 • Paper session 2 • Workshop 1 • Workshop 2	In parallel: • Paper session 6 • Paper session 7 • Paper session 8 • Panel discussion III	In parallel: • Paper session 10 • Paper session 11 • Paper session 12	
	15 ⁰⁰ -15 ³⁰ Coffee Break	14 ⁰⁰ -16 ⁰⁰ Plenary session I	13 ³⁰ -14 ³⁰	Lunch	Lunch	Lunch	13 ⁰⁰ -20 ⁰⁰ Post Conference Excursions (not included in conference fee)
	15 ³⁰ -17 ⁰⁰ Workshops in parallel (The Faculty of Philosophy**)	Coffee break	14 ³⁰ -16 ⁰⁰	In parallel: • Working Group presentations II • Working Group presentations III	14 ³⁰ -22 ⁰⁰ Excursion Dinner	In parallel: • Paper session 13 • Paper session 14 • Workshop 3	
	17 ⁰⁰ -18 ⁰⁰ Reflections and Panel Discussion (The Faculty of Philosophy**, room 301)	16 ³⁰ -17 ³⁰ Plenary Session II 17 ³⁰ -18 ³⁰ Panel Discussion I	16 ⁰⁰ -16 ³⁰	Coffee break		Coffee break	
		18 ³⁰ -19 ³⁰ Welcome Reception	17 ³⁰ -18 ³⁰	In parallel: • Paper session 3 • Paper session 4 • Paper session 5		In parallel: • Paper session 15 • Paper session 16 • Workshop 4 • Workshop 5	

* Main Building, The Theatre Hall: Universiteto St.3

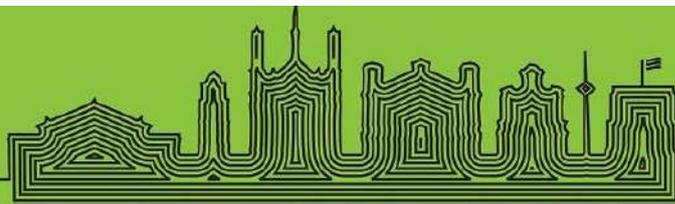
**The Faculty of Philosophy: Universiteto St. 9



Constructionism 2018

Constructionism, computational thinking
and educational innovation

Vilnius, Lithuania, August 21 to 25



Conference program

Monday, August 20		Teachers' day (Pre-Conference)	
9 ⁰⁰ –10 ⁰⁰	Registration & Coffee (in front of the Theatre Hall)		
10 ⁰⁰ –12 ⁰⁰	Opening Session: Main Building, The Theatre Hall (Universiteto St. 3) Welcome Giedrius Vaidelis (Lithuania). <i>Updating Educational Content: Challenges and Possibilities</i> Evgenia Sendova (Bulgaria). <i>The Beauty in Science and the Science in Beauty</i> Rimantas Želvyis (Lithuania). <i>Future Education: New Challenges for Lithuania?</i>		
12 ⁰⁰ –13 ⁰⁰	Lunch Registration in the Faculty of Philosophy (Universiteto St. 9)		
13 ⁰⁰ –15 ⁰⁰	Workshops (WS) in parallel: The Faculty of Philosophy (Universiteto St. 9) WS 1: room 106. Judith Bell (New Zealand). <i>Dynamic Teaching Ideas for Teaching Music Theory.</i> Target audience: primary school teachers & music teachers WS 3: room 204. Paul Goldenberg, Cynthia J. Carter (USA). <i>Developing Algebraic Habits of Mind in Students.</i> Target audience: mathematics teachers for students ages 11–18 WS 7: room 111. Petra Enges-Pyykönen (Finland). <i>VILLE – Electronic Learning Path for Mathematics and Programming.</i> Target audience: primary school teachers WS 8: room 201. Evgenia Sendova, Nikolina Nikolova (Bulgaria). <i>Constructionism in Action: Do we Need to Start from Scratch?</i> Target audience: all teachers WS 9: room 313. Gary Stager (USA). <i>Teaching Coding and Physical Computing.</i> Target audience: all teachers WS 10: room 112. Jacqueline Staub (Switzerland). <i>The Essence of Programming at School – Logo in a Spiral Curriculum.</i> Target audience: primary and lower secondary school teachers WS 11: room 308. Carol Sperry Suziedelis (USA). <i>How to Create and Sustain a Progressive Pedagogy in a Traditional Setting.</i> Target audience: all teachers WS 13: room 306. Annalise Duca, Angele Giuliano (Malta), Sofia Nikitopoulou, Nikoleta Yiannoutsou, Chronis Kynigos (Greece). <i>The ER4STEM Repository for Educational Robotics.</i> Target audience: all teachers		
15 ⁰⁰ –15 ³⁰	Coffee break		
15 ³⁰ –17 ⁰⁰	Workshops (WS) in parallel: The Faculty of Philosophy (Universiteto St. 9). WS 2: room 106. Tim Bell (New Zealand). <i>Computer Science Unplugged for Teachers.</i> Target audience: primary school teachers WS 4: room 204. Paul Goldenberg, Cynthia J. Carter (USA). <i>Puzzles & Programming to Develop Mathematical Habits of Mind in 6–10-year Olds.</i> Target audience: primary school teachers for students ages: 6–10 WS 5: room 201. Ivan Kalaš (Slovakia). <i>Powerful Ideas in Lower Primary Programming: High Time to Recognize Them.</i> Target audience: educators interested in lower primary computing (pupils aged 5 to 9) and general primary teachers WS 6: room 111. Witek Kranas (Poland). <i>SNAP! - Beauty & Joy of Computing (visually).</i> Target audience: informatics teachers, lower and upper secondary schools (6-12 grades) WS 8: room 401. Evgenia Sendova, Nikolina Nikolova (Bulgaria). <i>Constructionism in Action: Do we Need to Start from Scratch?</i> Target audience: all teachers WS 9: room 313. Gary Stager (USA). <i>Teaching Coding and Physical Computing.</i> Target audience: all teachers WS 10: room 112. Jacqueline Staub (Switzerland). <i>The Essence of Programming at School – Logo in a Spiral Curriculum.</i> Target audience: primary and lower secondary school teachers WS 11: room 308. Carol Sperry Suziedelis (USA). <i>How to Create and Sustain a Progressive Pedagogy in a Traditional Setting.</i> Target audience: all teachers WS 12: room 205. Igor Verner, Khayriah Massarwe, Daoud Bshouty (Israel). <i>Joyful Learning of Geometry in Cultural Context. Analysis and Construction of Geometric Ornaments.</i> Target audience: all teachers WS 13: room 306. Annalise Duca, Angele Giuliano (Malta), Sofia Nikitopoulou, Nikoleta Yiannoutsou, Chronis Kynigos (Greece). <i>The ER4STEM Repository for Educational Robotics.</i> Target audience: all teachers		
17 ⁰⁰ –18 ⁰⁰	Reflections and Panel Discussion: room 301		

Tuesday, August 21 / Location: Main Building, The Theatre Hall (Universiteto St. 3)				
8 ³⁰ –all day	Registration			
10 ⁰⁰ –11 ³⁰	Excursion to Old Vilnius University I			
11 ³⁰ –13 ⁰⁰	Excursion to Old Vilnius University II			
14 ⁰⁰ –16 ⁰⁰	Session chair: Valentina Dagienė Opening Plenary session I Rimantas Želvys. One Hundred Years of Educational Development in Lithuania James Clayson. <i>Look Closely, Watch What Happens: Visual Modelling and Constructionism</i>			
16 ⁰⁰ –16 ³⁰	Coffee break			
16 ³⁰ –17 ³⁰	Session chair: Arūnas Poviliūnas Plenary session II Gary Stager. <i>Making Constructionism Great Again</i>			
17 ³⁰ –18 ³⁰	Panel Discussion I <i>Inside the Trojan Horse – A Discussion Among the Next Generation of Constructionists</i> Sylvia Martinez (moderator), Gary Stager, Amy Dugré, Angela Sofia Lombardo, Susana Tesconi, Tracy Rudzitis, Brian C. Smith, Jaymes Dec			
18 ³⁰ –19 ³⁰	<i>Welcome Reception / Location: Grand Courtyard</i>			
Wednesday, August 22 / Location: The Faculty of Philosophy (Universiteto St. 9)				
8 ³⁰ –10 ³⁰	Session chair: Gerald Futschek Plenary session III: room 301 Carol Sperry Suziedelis. <i>The Evolution of a Constructionist Teacher (with Reminders from Seymour Papert)</i> Evgenia Sendova. <i>Back 100 000(2)</i>			
10 ³⁰ –11 ⁰⁰	Session chair: Chronis Kynigos Plenary session IV: room 302 Celia Hoyles, Richard Noss. <i>Scratchmaths: A Positive Outcome for Constructionism at Scale</i> Ivan Kalaš. <i>Programming in Lower Primary Years: Design Principles and Powerful Ideas</i>			
11 ⁰⁰ –11 ³⁰	Session chair: Chronic Kynigos Panel discussion II room 302 <i>Constructionism at Scale.</i> Nathan Holbert (moderator), Matthew Berland, Yasmin Kafai, Richard Noss, Celia Hoyles, Kylie Pepler, Debbie Fields			
11 ⁰⁰ –11 ³⁰	Coffee break			
11 ³⁰ –13 ³⁰	<table border="0" style="width: 100%;"> <tr> <td style="width: 33%; vertical-align: top;"> Session chair: Natasa Grgurina Paper session 1: room 301 Education and innovations Arthur Hjorth, Corey Brady, Uri Wilensky. <i>Sharing is Caring in the Commons – Students’ Conceptions about Sharing and Sustainability in Social-Ecological Systems</i> Arthur Hjorth, Uri Wilensky. <i>Urban Planning-in-Pieces: A Computational Approach to Understanding Conceptual Change and Causal Reasoning about Urban Planning</i> Sugat Dabholkar, Gabriella Anton, Uri Wilensky. <i>Developing Mathetic Content Knowledge Using an Emergent Systems Microworld</i> Elmara Pereira de Souza, Luísa Moura. <i>Constructionism as an Epistemological Option in Courses of Youth Center for Science and Culture – Bahia – Brazil</i> </td> <td style="width: 33%; vertical-align: top;"> Session chair: Evgenia Sendova Paper session 2: room 302 Constructionism in Mathematics Chantal Buteau, Ana Isabel Sacristán, Eric Muller. <i>Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics</i> Maite Mascaró, Ana Isabel Sacristán. <i>Assessing Learning through Exploratory Projects in Constructionist R-based Statistics Courses for Environmental Science Students</i> Christina Todorova, Carina Girvan, Nikoleta Yiannoutsou, Marianthi Grizioti, Ivaylo Gueorguiev, Pavel Varbanov, George Sharkov. <i>Visualizing Mathematics with the MathBot: A Constructionist Activity to Explore Mathematical Concepts through Robotics</i> Einari Kurvinen, Valentina Dagiene, Mikko-Jussi Laakso. <i>The Impact and Effectiveness of Technology Enhanced Mathematics Learning</i> </td> <td style="width: 33%; vertical-align: top;"> Session chair: Eglė Jasutė Workshop 1: room 111 Jacqueline Staub. <i>The Essence of Programming at School – Learning for Life</i> Workshop 2: room 111 Stephen Howell, Lizbeth Goodman. <i>Developing Body Tracking Software with Scratch and Kinect</i> </td> </tr> </table>	Session chair: Natasa Grgurina Paper session 1: room 301 Education and innovations Arthur Hjorth, Corey Brady, Uri Wilensky. <i>Sharing is Caring in the Commons – Students’ Conceptions about Sharing and Sustainability in Social-Ecological Systems</i> Arthur Hjorth, Uri Wilensky. <i>Urban Planning-in-Pieces: A Computational Approach to Understanding Conceptual Change and Causal Reasoning about Urban Planning</i> Sugat Dabholkar, Gabriella Anton, Uri Wilensky. <i>Developing Mathetic Content Knowledge Using an Emergent Systems Microworld</i> Elmara Pereira de Souza, Luísa Moura. <i>Constructionism as an Epistemological Option in Courses of Youth Center for Science and Culture – Bahia – Brazil</i>	Session chair: Evgenia Sendova Paper session 2: room 302 Constructionism in Mathematics Chantal Buteau, Ana Isabel Sacristán, Eric Muller. <i>Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics</i> Maite Mascaró, Ana Isabel Sacristán. <i>Assessing Learning through Exploratory Projects in Constructionist R-based Statistics Courses for Environmental Science Students</i> Christina Todorova, Carina Girvan, Nikoleta Yiannoutsou, Marianthi Grizioti, Ivaylo Gueorguiev, Pavel Varbanov, George Sharkov. <i>Visualizing Mathematics with the MathBot: A Constructionist Activity to Explore Mathematical Concepts through Robotics</i> Einari Kurvinen, Valentina Dagiene, Mikko-Jussi Laakso. <i>The Impact and Effectiveness of Technology Enhanced Mathematics Learning</i>	Session chair: Eglė Jasutė Workshop 1: room 111 Jacqueline Staub. <i>The Essence of Programming at School – Learning for Life</i> Workshop 2: room 111 Stephen Howell, Lizbeth Goodman. <i>Developing Body Tracking Software with Scratch and Kinect</i>
Session chair: Natasa Grgurina Paper session 1: room 301 Education and innovations Arthur Hjorth, Corey Brady, Uri Wilensky. <i>Sharing is Caring in the Commons – Students’ Conceptions about Sharing and Sustainability in Social-Ecological Systems</i> Arthur Hjorth, Uri Wilensky. <i>Urban Planning-in-Pieces: A Computational Approach to Understanding Conceptual Change and Causal Reasoning about Urban Planning</i> Sugat Dabholkar, Gabriella Anton, Uri Wilensky. <i>Developing Mathetic Content Knowledge Using an Emergent Systems Microworld</i> Elmara Pereira de Souza, Luísa Moura. <i>Constructionism as an Epistemological Option in Courses of Youth Center for Science and Culture – Bahia – Brazil</i>	Session chair: Evgenia Sendova Paper session 2: room 302 Constructionism in Mathematics Chantal Buteau, Ana Isabel Sacristán, Eric Muller. <i>Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics</i> Maite Mascaró, Ana Isabel Sacristán. <i>Assessing Learning through Exploratory Projects in Constructionist R-based Statistics Courses for Environmental Science Students</i> Christina Todorova, Carina Girvan, Nikoleta Yiannoutsou, Marianthi Grizioti, Ivaylo Gueorguiev, Pavel Varbanov, George Sharkov. <i>Visualizing Mathematics with the MathBot: A Constructionist Activity to Explore Mathematical Concepts through Robotics</i> Einari Kurvinen, Valentina Dagiene, Mikko-Jussi Laakso. <i>The Impact and Effectiveness of Technology Enhanced Mathematics Learning</i>	Session chair: Eglė Jasutė Workshop 1: room 111 Jacqueline Staub. <i>The Essence of Programming at School – Learning for Life</i> Workshop 2: room 111 Stephen Howell, Lizbeth Goodman. <i>Developing Body Tracking Software with Scratch and Kinect</i>		
13 ³⁰ –14 ³⁰	Lunch			
14 ³⁰ –16 ⁰⁰	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> Session chair: Gabrielė Stupurienė Working Group (WG) presentations II: room 301 WG 1: Gerald Futschek, Bernhard Standl, Chantal Buteau, Andrew Csizmadia, Lilia Georgieva, Lina Vinikienė, Jane Waite. <i>Constructionist Approaches to Computational Thinking.</i> </td> <td style="width: 50%; vertical-align: top;"> Session chair: Tatjana Jevsikova Working Group (WG) presentations III: room 302 WG 3: Evgenia Sendova, Christos Chytas, Katarzyna Ołędzka, Ralf Romeike, Wolfgang Slany. <i>Creating and Looking at Art with Logo Eyes.</i> </td> </tr> </table>	Session chair: Gabrielė Stupurienė Working Group (WG) presentations II: room 301 WG 1: Gerald Futschek, Bernhard Standl, Chantal Buteau, Andrew Csizmadia, Lilia Georgieva, Lina Vinikienė, Jane Waite. <i>Constructionist Approaches to Computational Thinking.</i>	Session chair: Tatjana Jevsikova Working Group (WG) presentations III: room 302 WG 3: Evgenia Sendova, Christos Chytas, Katarzyna Ołędzka, Ralf Romeike, Wolfgang Slany. <i>Creating and Looking at Art with Logo Eyes.</i>	
Session chair: Gabrielė Stupurienė Working Group (WG) presentations II: room 301 WG 1: Gerald Futschek, Bernhard Standl, Chantal Buteau, Andrew Csizmadia, Lilia Georgieva, Lina Vinikienė, Jane Waite. <i>Constructionist Approaches to Computational Thinking.</i>	Session chair: Tatjana Jevsikova Working Group (WG) presentations III: room 302 WG 3: Evgenia Sendova, Christos Chytas, Katarzyna Ołędzka, Ralf Romeike, Wolfgang Slany. <i>Creating and Looking at Art with Logo Eyes.</i>			

	<p>WG 5: Michael Weigend, Kazunari Ito, Anita Juškevičienė, Igor Pesek, Zsuzsa Pluhár, Jiří Vaniček. <i>Constructionism in the Classroom: Creative Learning Activities on Computational Thinking</i>.</p> <p>WG 6: Mattia Monga, Michael Lodi, Dario Malchiodi, Anna Morpurgo, Oluwakemi Oduwole, Bamidele Oluchi, Bernadette Spieler. <i>Learning to Program in a Constructionist Way</i>.</p>	<p>WG 4: Lilija Duoblienė, Jūratė Baranova, Luc Anckaert, Wilfried Baumann. <i>The Constructive Strategies in Teaching Humanities with Films</i>.</p> <p>WG 7: Ana Isabel Sacristán, Richard Akrofi Kwabena Baafi, Lina Kaminskienė, Michaela Sabin. <i>Constructionism in Upper Secondary and Tertiary Levels</i>.</p>	
16 ⁰⁰ –16 ³⁰	Coffee break		
16 ³⁰ –18 ³⁰	<p>Session chair: Mattia Monga Paper session 3: room 301</p> <p>Computational Thinking Judith Bell, Tim Bell. <i>Computational Thinking and Music Learning</i> Marianthi Grizioti, Chronis Kynigos. <i>Programming Approaches to Computational Thinking: Integrating Turtle Geometry, Dynamic Manipulation and 3D Space</i> Marianthi Grizioti, Chronis Kynigos. <i>Constructionist Approaches to Computational Thinking: A Case of Game Modding with ChoiCo</i> Anita Juškevičienė, Valentina Dagienė. <i>Interconnection Between Computational Thinking and Digital Competence</i></p>	<p>Session chair: Jiří Vaniček Paper session 4: room 302</p> <p>Constructionist approaches Valentina Dagienė, Gabrielė Stupurienė. <i>Short Tasks – Big Ideas: Constructive Approach for Learning and Teaching of Informatics Concepts in Primary Education</i>. Miroslava Černočová, Radek Čuma, Hasan Selcuk. <i>Forming Concepts for Programming Conditional Statements in the Primary School</i> Jean Griffin. <i>Constructionism and De-Constructionism as Complementary Pedagogies</i> Tilman Michaeli, Stefan Seegerer, Ralf Romeike. <i>Enabling Collaboration and Tinkering: A Version Control System for Block-based Languages</i> Jake Rowan Byrne, Kevin Sullivan, Katriona O’sullivan. <i>Active Learning of Computer Science Using a Hackathon-like Pedagogical Model</i></p>	<p>Session chair: Arthur Hjorth Paper session 5: room 306</p> <p>Reflections Nicolas Pope, Jonathan Foss, Meurig Beynon. <i>Reconstructing Constructionism by Construal</i> Deborah Fields, Mia Shaw, Yasmin Kafai. <i>Personal Learning Journeys: Reflective Portfolios as “Objects-to-Learn-With” in an E-textiles High School Class</i> Evgeny Patarakin. <i>Using Agent-based Modelling of Collaboration for Social Reflection</i> Francesca Agatolio, Alfredo Asiain, Alfredo Pina, Gabriel Rubio, Michele Moro. <i>Constructive and Collaborative Digital Storytelling for Enhancing Creativity and Cooperation In and Out of School</i></p>
Thursday, August 23 / Location: The Faculty of Philosophy (Universiteto St. 9)			
8 ³⁰ –10 ³⁰	<p>Session chair: Gary Stager Plenary session V: room 301</p> <p>Gerald Futschek. <i>Computational Thinking and Creativity</i> Tim Bell. <i>CS Unplugged and Computational thinking</i></p>	<p>Session chair: James Clayton Plenary session VI: room 302</p> <p>Uri Wilensky. <i>Reempowering powerful ideas</i> Paulo Blikstein. <i>Constructionism Won, Now What? The Role of Constructionist Research in the Age of Ubiquitous Computing</i></p>	
10 ³⁰ –11 ⁰⁰	<p>Session chair: Anita Juškevičienė Poster Session I: room 301</p> <p>Nalin Tutiyauphuengprasert. <i>Applied Constructionism: Critical Reflection and Learning Through Play in Adult Learning</i> Sawaros Thanapornsanguth, Nathan Holbert, Monica Chan. <i>Towards Girls’ Self-perception in Technology and Craft: Challenges and Implications</i> Enric Ortega Torres, Vincent Sanjosé López, Joan-Josep Solaz Portolés. <i>Influence of Students’ Self-perceived Use of Metacognitive Strategies and Sensory Preferences on Academic Achievement in Science and Technology</i> Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, Yasushi Kuno. <i>Programming Lessons for Kindergarten Children in Japan</i> Sayaka Tohyama, Yugo Takeuchi. <i>Collaborative Creative Music Activity with ICT: A Case Study for Children in Grade Five</i> Yoshiaki Matsuzawa, Misako Noguchi, Issei Nakano. <i>Exploration of Algorithm Abstraction Process with Cubetto and Middle Grade Elementary Kids</i> Aoi Yoshida, Kazunari Ito, Kazuhiro Abee. <i>A Practical Report on a Programming Course with “Making” Using micro:bit</i> Liudmyla Kryvoruchka. <i>Heuristic Potential of Open Institutional Models in Researchers Education.</i></p>	<p>Session chair: Lina Vinikienė Poster Session II: room 302</p> <p>Michael Tan. <i>Constructing what? Knowledge of the powerful, and powerful knowledge</i> Carina Girvan, Wilfried Lepuschitz, Ivaylo Gueorguiev, Christina Todorova, Chronis Kynigos, Marianthi Grizioti, Angele Giuliano, Annalise Duca, Julian Mauricio Angel-Fernandez, Markus Vincze. <i>Educational Robotics for STEM: From Workshops to Curricula and Framework</i> Ivaylo Gueorguiev, Christina Todorova, Nikoleta Yiannoutsou, Xristina Greka, Pavel Varbanov, George Sharkov, Carina Girvan, Julian Mauricio Angel-Fernandez, Lisa Vittori, Annalise Duca. <i>Towards a Generic Curriculum for Educational Robotics in STEM: From Scientific Concepts to Technologies and Powerful Ideas</i> Barbara Sabitzer. <i>Modeling Across the Subjects</i> Jinbao Zhang. <i>An Experimental Exploration of the Development of Design Thinking in University Maker Courses</i> Márton Visnovitz, Győző Horváth. <i>The Web – A Platform for Creation</i> Pekka Mäkiaho, Timo Poranen, Katriina Vartiainen. <i>Construction of a Project Monitoring Application Iteratively and Incrementally</i> Lina Vinikienė, Valentina Dagienė. <i>Different Cultures – Different Approaches to Reasoning and Algorithms</i></p>	

11 ⁰⁰ –11 ³⁰	Coffee break		
11 ³⁰ –13 ³⁰	<p>Session chair: Jacqueline Staub Paper session 6: room 301</p> <p>Programming education Jiří Vaníček. <i>Concept-building Oriented Programming Education</i> Ungyeol Jung, Young Jun Lee. <i>The Direction and Possibility for Social Justice in Informatics Education based on Bebras Challenge in Republic of Korea</i> Ken Kahn, Niall Winters. <i>AI Programming by Children</i> Elisabeth Wetzinger, Gerald Futschek, Bernhard Standl. <i>A Creative Learning Sequence in an Introductory Programming MOOC</i></p>	<p>Session chair: Wolfgang Slany Paper session 7: room 302</p> <p>Robotics Julian Mauricio Angel Fernandez, Nikoleta Yiannoutsou, Chronis Kynigos, Carina Girvan, Markus Vincze. <i>Towards a Framework for Educational Robotics</i> Flavio Campos. <i>Design Curriculum for Educational Robotics: Constructionist Pedagogical Experience in Formal Education</i> Dave Catlin, Martin Kandlhofer, Stephanie Holmquist, Andrew Paul Csizmadia, Julian Mauricio Angel Fernandez, John-John Cabibihan. <i>EduRobot Taxonomy and Papert's Paradigm</i> Karolína Mayerová, Michaela Veselovská. <i>How Students Struggled with Preparation of Activities for a Leisure Time Robotic Workshop</i></p>	<p>Session chair: Ana Isabel Sacristán Paper session 8: room 306</p> <p>Jose Armando Valente, Paulo Blikstein. <i>The Construction of Knowledge in Maker Education: A Constructivist Perspective</i></p> <p>Panel discussion III: room 306 <i>Constructionism across Cultures: Commonalities and Differences of Constructionist Implementations Around the World</i></p> <p>joined with papers</p> <p>Jose Armando Valente, Paulo Blikstein. <i>Constructionism in Different Cultures: the case of Brazil &</i> Deborah Fields, Paulo Blikstein. <i>What Is Constructionism? Views from a Thai Perspective</i> Jose Armando Valente (moderator), Paulo Blikstein, Deborah Fields, Michael Tan</p>
13 ³⁰ –14 ³⁰	Lunch		
14 ³⁰ –22 ⁰⁰	Excursion & Dinner		
Friday, August 24 / Location: The Faculty of Philosophy (Universiteto St. 9)			
8 ³⁰ –10 ³⁰	<p>Session chair: Ivan Kalaš Plenary session VII: room 301</p> <p>Paul Goldenberg. <i>Teaching Children to be Problem Posers and Puzzle-creators in Mathematics</i> Ana Isabel Sacristán. <i>Constructionist Experiences for Mathematics across Educational Levels</i></p>	<p>Session chair: Jose Armando Valente Plenary session VIII: room 302</p> <p>Wolfgang Slany. <i>Rock Bottom, the World, the Sky: Catrobat, an Extremely Large-scaling and Long-term Visual Coding Project Relying Purely on Smartphones</i> Gary Stager, co-speaker Sylvia Martinez. <i>Turning Theory Into Practice – Spreading Constructionism</i></p>	
10 ³⁰ –11 ⁰⁰	<p>Paper session 9: room 301</p> <p>Tiina Partanen, Pia Niemelä, Timo Poranen. <i>Racket Programming Material for Finnish Elementary Math Education</i></p>	<p>Demo session 1: room 302</p> <p>Markus Klein, Clemens Koza, Wilfried Lepuschitz, Gottfried Koppensteiner. <i>Hedgehog: A Versatile Controller for Educational Robotics</i></p>	
11 ⁰⁰ –11 ³⁰	Coffee break		
11 ³⁰ –13 ³⁰	<p>Session chair: Miroslava Černochová Paper session 10: room 301</p> <p>Designing activities Yasmin Kafai, Deborah Fields. <i>Some Reflections on Designing Constructionist Activities for Classrooms</i> Kit Martin, Michael Horn, Uri Wilensky. <i>Ant Adaptation: A Complex Interactive Multitouch Game About Ants Designed for Museums</i> Michael Weigend, Fenja Göcking, Alexander Knuth, Patrick Pais Pereira, Laura Schmidt. <i>Media Parkour – Experiential Learning Activities for Media Education</i> Brendan Tangney, Ian Boran, Tony Knox, Aibhin Bray. <i>Constructionist STEM Activities Using the Bridge21 Model</i></p>	<p>Session chair: Márton Visnovitz Paper session 11: room 302</p> <p>Teacher education Daniel Hickmott, Elena Prieto-Rodriguez. <i>To Assess or Not to Assess: Tensions Negotiated in Six Years of Teaching Teachers about Computational Thinking</i> Daniel Hickmott, Elena Prieto-Rodriguez. <i>Constructionist Experiences in Teacher Professional Development: A Tale of Five Years</i> Igor Verner, Khayriah Massarwe, Daoud Bshouty. <i>Ethnomathematics in Teacher Education: Analysis and Construction of Geometric Ornaments</i> Xiaoxue Du, Kay Chioma Igwe. <i>Computational Thinking in Teacher Professional Development Programs</i></p>	<p>Session chair: Don Passey Paper session 12: room 306</p> <p>Methodologies Sven Jatzlau, Ralf Romeike. <i>How High is the Ceiling? Applying Core Concepts of Block-based Languages to Extend Programming Environments</i> Anton Chukhnov, Sergei Pozdniakov, Ilya Posov, Athit Maytarattanakhon. <i>Analysis of Constructive and Cognitive Activities of Participants in Online Competitions in Computer Science</i> Vladimiras Dolgopolas, Valentina Dagienė, Eglė Jasutė, Tatjana Jevsikova. <i>Design Science Research for Computational Thinking in Constructionist Education: A Pragmatic Perspective</i> Aleksandra Klačnja-Milićević, Mirjana Ivanović. <i>Learning Analytics in Education: Objectives, Application Possibilities and Challenges</i></p>

13 ³⁰ –14 ³⁰	Lunch		
14 ³⁰ –16 ⁰⁰	<p>Session chair: Mihaela Sabin Paper session 13: room 301 Curriculum matters Eva Klimeková. <i>Curriculum Intervention for Learning Programming in Python with Turtle Geometry</i> Carol Angulo, Alberto J. Cañas, Ana Gabriela Castro, Leda Muñoz, Natalia Zamora. <i>Think, Create and Program: Evolving to a K-9 Nationwide Computational Thinking Curriculum in Costa Rica</i> Michael Weigend. <i>Coding to Learn - Informatics in Science Education</i></p>	<p>Session chair: Carina Girvan Paper session 14: room 302 Girls in computing Bernadette Spieler, Wolfgang Slany. <i>Female Teenagers and Coding: Create Gender Sensitive and Creative Learning Environments</i> Caitlin Davey, Sawaros Thanapornsanguth, Nathan Holbert. <i>Making Together: Cultivating Community of Practice in an All-Girl Constructionist Learning Environment</i> Sawaros Thanapornsanguth, Nathan Holbert. <i>Exploring Girls' Values and Perspectives in Making for Others</i></p>	<p>Session chair: Wilfried Baumann Workshop 3: room 111 Corey Brady, Walter Stroup, Tony Petrosino, Uri Wilensky. <i>Group-based Simulation and Modelling: Technology Supports for Social Constructionism</i></p>
16 ⁰⁰ –16 ³⁰	Coffee break		
16 ³⁰ –18 ³⁰	<p>Session chair: Witold Kranas Paper session 15: room 301 Constructionist environments Christos Chytas, Ira Diethelm. <i>Designing Constructionist Learning Environments with Computational Design and Digital Fabrication</i> Kazunari Ito. <i>Pictogramming: Learning Environment Using Human Pictograms Based on Constructionism</i> Kazunari Ito, Aoi Yoshida, Takashi Yoneda, Yuichi Oie. <i>Human Pictogram Unplugged: Unified Learning Environment of Computer Science Unplugged Using Human Pictograms</i> Nobuko Kishi, Mari Yoshida, Minori Yoshizawa, Aoi Yoshida. <i>VISURATCTH: Visualization Tool for Finding Characteristics of Teaching and Learning Process of Scratch Programmers</i></p>	<p>Session chair: Michael Weigend Paper session 16: room 302 Modeling Natasa Grgurina, Erik Barendsen, Cor Suhre, Klaas van Veen, Bert Zwaneveld. <i>Assessment of Modeling Projects in Informatics Class</i> Kit Martin, Gabriella Anton. <i>Modeling Time</i> Ümit Aslan, Uri Wilensky. <i>Agent-based Construction (a-b-c) Interviews: A Generative Case Study</i> Yu Guo, Uri Wilensky. <i>Mind the Gap: Teaching High School Students about Wealth Inequality through Agent-Based Participatory Simulations</i></p>	<p>Session chair: Brian Harvey Workshop 4: room 111 Ken Kahn. <i>AI Programming in Snap!</i> Workshop 5: room 111 Stephen Howell, Neeltje Berger, Peter Heldens, Kevin Marshall, Clare Riley. <i>Developing Affordable STEM Maker Projects with BBC Micro:bits and Microsoft MakeCode</i></p>
Saturday, August 25 / Location: The Faculty of Philosophy (Universiteto St. 9)			
8 ³⁰ –10 ³⁰	<p>Session chair: Deborah Fields Plenary session IX: room 301 Chronis Kynigos. <i>In Support of Integrated Approaches to Constructionist Designs and Interventions: The Case of ChoiCo and MaLT</i> Arthur Hjorth. <i>Social Gears - a Constructionist Approach to Social Studies</i></p>	<p>Session chair: Paul Goldenberg Plenary session X: room 302 Brian Harvey. <i>May I Teach an Algorithm?</i> Jens Möning. <i>Bones, Gears and Witchcraft</i></p>	
10 ³⁵ –11 ³⁰	<p>Session chair: Eugenijus Kurilovas Demo session 2: room 301 Nevin Akcay, Hulya Avci, Ali Güngör, Tufan Adiguzel. <i>The Relationship between Computer Programming and English Language Skills</i> Monica Chan, Gary Lee. <i>Synthesizing the Mesh: Using Constructible Authentic Representations to Gain Intuitive Understanding of Bayesian Reasoning</i></p>	<p>Session chair: Christos Chytas Demo session 3: room 302 Ken Kahn. <i>Interpolating (and Extrapolating) 3D Turtle Programs in Beetle Blocks</i> Stephen Howell. <i>Teaching Computational Thinking with Minecraft & Microsoft MakeCode</i></p>	<p>Session chair: Ralf Romeike Workshop 6: room 111 Brian Broll, Corey Brady, Ákos Lédeczi. <i>NetsBlox: A Constructionist Environment for Creating Distributed Applications</i></p>
11 ³⁰ –12 ⁰⁰	Closing: Farewell buffet / Location: Grand Courtyard		
13 ⁰⁰ –20 ⁰⁰	Post Conference Excursions (not included in conference fee)		

Content

Opening / Plenaries.....	19
One Hundred Years of Educational Development in Lithuania.....	20
<i>Rimantas Želvys</i>	
CS Unplugged and Computational thinking	21
<i>Tim Bell</i>	
Constructionism Won, Now What? The Role of Constructionist Research in the Age of Ubiquitous Computing	29
<i>Paulo Blikstein</i>	
Look Closely, Watch What Happens: Visual Modelling and Constructionism	30
<i>James Clayson</i>	
Computational Thinking and Creativity	38
<i>Gerald Futschek</i>	
Teaching Children to be Problem Posers and Puzzle Creators in Mathematics	39
<i>Paul Goldenberg</i>	
May I Teach an Algorithm?	53
<i>Brain Harvey</i>	
Social Gears – a Constructionist Approach to Social Studies	68
<i>Arthur Hjorth</i>	
Scratchmaths: a Positive Outcome for Constructionism at Scale.....	69
<i>Richard Noss, Celia Hoyles</i>	
Programming in Lower Primary Years: Design Principles and Powerful Ideas	71
<i>Ivan Kalaš</i>	
In Support of Integrated Approaches to Constructionist Designs and Interventions: The Case of ChoiCo and MaLT	81
<i>Chronis Kynigos</i>	
Bones, Gears and Witchcraft	82
<i>Jens Monig</i>	
Constructionist Experiences for Mathematics Across Educational Levels	83
<i>Ana Isabel Sacristán</i>	
Back 100 000 ₍₂₎	94
<i>Evgenia Sendova</i>	
Rock Bottom, the World, the Sky: Catrobat, an Extremely Large-scale and Long-term Visual Coding Project Relying Purely on Smartphones.....	104
<i>Wolfgang Slany, Kirshan Kumar Luhana, Matthias Mueller, Christian Schindler, Bernadette Spieler</i>	
Making Constructionism Great Again.....	120
<i>Gary S. Stager</i>	
Turning Theory into Practice – Spreading Constructionism.....	121
<i>Gary S. Stager, Sylvia Martinez</i>	
The Evolution of a Constructionist Teacher (with Some Reminders from Seymour).....	122
<i>Carol Sperry Suziedelis</i>	
Reempowering Powerful Ideas	123
<i>Uri Wilensky</i>	
Research papers	124
Agent-based Construction (a-b-c) Interviews: A Generative Case Study.....	125
<i>Umit Aslan, Uri Wilensky</i>	

Active Learning of Computer Science Using a Hackathon-like Pedagogical Model	138
<i>Jake Rowan Byrne, Kevin Sullivan, Katriona O’Sullivan</i>	
EduRobot Taxonomy and Papert’s Paradigm	150
<i>Dave Catlin, Martin Kandlhofer, Stephanie Holmquist, Andrew Paul Csizmadia, Julian M. Angel-Fernandez, John-John Cabibihan</i>	
Analysis of Constructive and Cognitive Activities of Participants in Online Competitions in Computer Science	160
<i>Anton Chukhnov, Sergei Pozdniakov, Ilya Posov, Athit Maytarattanakhon</i>	
Short Tasks – Big Ideas: Constructive Approach for Learning and Teaching of Informatics Concepts in Primary Education	169
<i>Valentina Dagienė, Gabrielė Stupurienė</i>	
Design Science Research for Computational Thinking in Constructionist Education: A Pragmatic Perspective	180
<i>Vladimiras Dolgopolas, Valentina Dagienė, Eglė Jasutė, Tatjana Jevsikova</i>	
Computational Thinking in Teacher Professional Development Programs	193
<i>Xiaoxue Du, Kay Chioma Igwe</i>	
What is Constructionism? Views from a Thai Perspective	203
<i>Deborah A. Fields, Paulo Blikstein</i>	
Personal Learning Journeys: Reflective Portfolios as “Objects-to-Learn-With” in an E-textiles High School Class	214
<i>Deborah A. Fields, Mia S. Shaw, Yasmin B. Kafai</i>	
Constructionism and De-Constructionism as Complementary Pedagogies	225
<i>Jean M. Griffin</i>	
Mind the Gap: Teaching High School Students about Wealth Inequality through Agent-based Participatory Simulations	238
<i>Yu Guo, Uri Wilensky</i>	
To Assess or Not to Assess: Tensions Negotiated in Six Years of Teaching Teachers about Computational Thinking	251
<i>Daniel Hickmott, Elena Prieto-Rodriguez</i>	
Sharing is Caring in the Commons – Students’ Conceptions about Sharing and Sustainability in Social-Ecological Systems	263
<i>Arthur Hjorth, Corey Brady, Uri Wilensky</i>	
Urban Planning-in-Pieces: A Computational Approach to Understanding Conceptual Change and Causal Reasoning about Urban Planning	274
<i>Arthur Hjorth, Uri Wilensky</i>	
How High is the Ceiling? Applying Core Concepts of Block-based Languages to Extend Programming Environments	285
<i>Sven Jatzlau, Ralf Romeike</i>	
The Direction and Possibility for Social Justice in Informatics Education based on Bebras Challenge in Republic of Korea	295
<i>Ungyeol Jung, Young-jun Lee</i>	
Interconnection between Computational Thinking and Digital Competence	305
<i>Anita Juškevičienė, Valentina Dagienė</i>	
AI Programming by Children	315
<i>Ken Kahn, Niall Winters</i>	
VISURATCH: Visualization Tool for Finding Characteristics of Teaching and Learning Process of Scratch Programmers	325
<i>Nobuko Kishi, Mari Yoshida, Minori Yoshizawa, Aoi Yoshida</i>	

Curriculum Intervention for Learning Programming in Python with Turtle Geometry	334
<i>Eva Klimeková</i>	
The Impact and Effectiveness of Technology Enhanced Mathematics Learning	344
<i>Einari Kurvinen, Valentina Dagienė, Mikko-Jussi Laakso</i>	
Constructionist Approaches to Computational Thinking: A Case of Game Modding with ChoiCo	357
<i>Grizioti Marianthi, Chronis Kynigos</i>	
Programming Approaches to Computational Thinking: Integrating Turtle Geometry, Dynamic Manipulation and 3D Space.....	369
<i>Grizioti Marianthi, Chronis Kynigos</i>	
Modeling Time.....	380
<i>Kit Martin, Gabriella Anton</i>	
Ant Adaptation: A Complex Interactive Multitouch Game about Ants Designed for Museums	392
<i>Kit Martin, Michael Horn, Uri Wilensky</i>	
Enabling Collaboration and Tinkering: A Version Control System for Block-based Languages.....	405
<i>Tilman Michaeli, Stefan Seegerer, Ralf Romeike</i>	
Racket Programming Material for Finnish Elementary Math Education	415
<i>Tiina Partanen, Pia Niemelä, Timo Poranen</i>	
Using Agent-based Modelling of Collaboration for Social Reflection.....	426
<i>Evgeny Patarakin</i>	
Reconstructing Constructionism by Construal	437
<i>Nicolas Pope, Jonathan Foss, Meurig Beynon</i>	
Constructionist STEM Activities Using the Bridge21 Model.....	449
<i>Brendan Tangney, Ian Boran, Tony Knox, Aibhín Bray</i>	
Exploring Girls' Values and Perspectives in Making for Others	460
<i>Sawaros Thanapornsanguth, Nathan Holbert</i>	
The Construction of Knowledge in Maker Education: A Constructivist Perspective	472
<i>José Armando Valente, Paulo Blikstein</i>	
Constructionism in Different Cultures: the Case of Brazil	481
<i>José Armando Valente, Paulo Blikstein,</i>	
Concept-building Oriented Programming Education.....	488
<i>Jiří Vaníček</i>	
Practice papers	497
Constructive and Collaborative Digital Storytelling for Enhancing Creativity and Cooperation In and Out of School.....	498
<i>Francesca Agatolio, Alfredo Asiain, Alfredo Pina, Gabriel Rubio, Michele Moro</i>	
Towards a Framework for Educational Robotics	506
<i>Julian M. Angel-Fernandez, Nikoleta Yiannoutsou, Chronis Kynigos, Carina Girvan, Markus Vincze</i>	
Think, Create and Program: Evolving to a K-9 Nationwide Computational Thinking Curriculum in Costa Rica	514
<i>Carol Angulo, Alberto J. Cañas, Ana Gabriela Castro, Leda Muñoz, Natalia Zamora</i>	
Computational Thinking and Music Learning	520
<i>Judith Bell, Tim Bell</i>	
Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics	528
<i>Chantal Buteau, Ana Isabel Sacristán, Eric Muller</i>	
Design Curriculum for Educational Robotics: Constructionist Pedagogical Experience in Formal Education	536
<i>Flavio Campos</i>	

Forming Concepts for Programming Conditional Statements in the Primary School	543
<i>Miroslava Černočová, Radek Čuma, Hasan Selcuk</i>	
Designing Constructionist Learning Environments with Computational Design and Digital Fabrication	547
<i>Christos Chytas, Ira Diethelm</i>	
Developing Mathetic Content Knowledge using an Emergent Systems Microworld.....	554
<i>Sugat Dabholkar, Gabriella Anton, Uri Wilensky</i>	
Making Together: Cultivating Community of Practice in an All-Girl Constructionist Learning Environment	561
<i>Caitlin Davey, Sawaros Thanapornsanguth, Nathan Holbert</i>	
Assessment of Modeling Projects in Informatics Class	570
<i>Natasa Grgurina, Erik Barendsen, Cor Suhre, Klaas van Veen, Bert Zwaneveld</i>	
Constructionist Experiences in Teacher Professional Development: A Tale of Five Years	577
<i>Daniel Hickmott, Elena Prieto-Rodriguez</i>	
Pictogramming: Learning Environment Using Human Pictograms Based on Constructionism	585
<i>Kazunari Ito</i>	
Human Pictogram Unplugged: Unified Learning Environment of Computer Science Unplugged Using Human Pictograms	593
<i>Kazunari Ito, Aoi Yoshida, Takashi Yoneda, Yuichi Oie</i>	
Some Reflections on Designing Constructionist Activities for Classrooms.....	601
<i>Yasmin B. Kafai, Deborah A. Fields</i>	
Learning Analytics in Education: Objectives, Application Possibilities and Challenges.....	608
<i>Aleksandra Klačnja-Milićević, Mirjana Ivanović</i>	
Assessing Learning through Exploratory Projects in Constructionist R-based Statistics Courses for Environmental Science Students.....	615
<i>Maite Mascaró, Ana Isabel Sacristán</i>	
How Students Struggled with Preparation of Activities for a Leisure Time Robotic Workshop.....	623
<i>Karolína Mayerová, Michaela Veselovská</i>	
Constructionism as an Epistemological Option in Courses of Youth Center for Science and Culture – Bahia – Brazil.....	631
<i>Elmara Pereira de Souza, Luísa Souza Moura</i>	
Female Teenagers and Coding: Create Gender Sensitive and Creative Learning Environments	637
<i>Bernadette Spieler, Wolfgang Slany</i>	
Visualizing Mathematics with the MathBot: a Constructionist Activity to Explore Mathematical Concepts through Robotics.....	649
<i>Christina Todorova, Carina Girvan, Nikoleta Yiannoutsou, Marianthi Grizioti, Ivaylo Gueorguiev, Pavel Varbanov, George Sharkov</i>	
Ethnomathematics in Teacher Education: Analysis and Construction of Geometric Ornaments	657
<i>Igor Verner, Khayriah Massarwe, Daoud Bshouty</i>	
Coding to Learn – Informatics in Science Education.....	664
<i>Michael Weigend</i>	
Media Parkour– Experiential Learning Activities for Media Education.....	672
<i>Michael Weigend, Fenja Göcking, Alexander Knuth, Patrick Pais Pereira, Laura Schmidt</i>	
A Creative Learning Sequence in an Introductory Programming MOOC	678
<i>Elisabeth Wetzinger, Gerald Futschek, Bernhard Standl</i>	
Posters	685
Educational Robotics for STEM: From Workshops to Curricula and Framework	686
<i>Carina Girvan, Wilfried Lepuschitz, Ivaylo Gueorguiev, Christina Todorova, Chronis Kynigos, Marianthi Grizioti, Angele Giuliano, Annalise Duca, Julian M. Angel-Fernandez, Markus Vincze</i>	

Towards a Generic Curriculum for Educational Robotics in STEM: From Scientific Concepts to Technologies and Powerful Ideas	690
<i>Ivaylo Gueorguiev, Christina Todorova, Nikoleta Yiannoutsou, Xristina Greka, Pavel Varbanov, George Sharkov, Carina Girvan, Julian M. Angel-Fernandez, Lisa Vittori, Annalise Duca</i>	
Heuristic Potential of Open Institutional Models in Researchers Education	694
<i>Liudmyla Kryvoruchka</i>	
Construction of a Project Monitoring Application Iteratively and Incrementally	698
<i>Pekka Mäkiäho, Timo Poranen, Katriina Vartiainen</i>	
Exploration of Algorithm Abstraction Process with Cubetto and Middle Grade Elementary Kids.....	703
<i>Yoshiaki Matsuzawa, Misako Noguchi, Issei Nakano</i>	
Influence of Students' Self-perceived Use of Metacognitive Strategies and Sensory Preferences on Academic Achievement in Science and Technology	707
<i>Enric Ortega Torres, Vincent Sanjosé López, Joan-Josep Solaz Portolés</i>	
Modeling Across the Subjects	711
<i>Barbara Sabitzer</i>	
Constructing What? Knowledges of the Pwerful, and Powerful Knowledges	714
<i>Michael Tan</i>	
Towards Girls' Self-perception in Technology and Craft: Challenges and Implications	718
<i>Sawaros Thanapornsanguth, Nathan Holbert, Monica Chan</i>	
Collaborative Creative Music Activity with ICT: A Case Study for Children in Grade Five	723
<i>Sayaka Tohyama, Yugo Takeuchi</i>	
Applied Constructionism: Critical Reflection and Learning Through Play in Adult Learning	727
<i>Nalin Tutiyaphuengprasert</i>	
Different Cultures – Different Approaches to Reasoning and Algorithms	731
<i>Valentina Dagienė, Lina Vinikienė</i>	
The Web – A Platform for Creation	736
<i>Márton Visnovitz, Győző Horváth</i>	
Programming Lessons for Kindergarten Children in Japan	741
<i>Takeshi Watanabe, Yuriko Nakayama, Yasunori Harada, Yasushi Kuno</i>	
A Practical Report on a Programming Course with “Making” Using Micro:bit	745
<i>Aoi Yoshida, Kazunari Ito, Kazuhiro Abee</i>	
An Experimental Exploration of the Development of Design Thinking in University Maker Courses	750
<i>Jinbao Zhang</i>	
Panels / Workshops / Demonstrations / Working groups	753
Panels	754
Constructionism across Cultures: Commonalities and Differences of Constructionist Implementations around the World	754
<i>Paulo Blickstein</i>	
Constructionism at Scale.....	754
<i>Celia Hoyles, Richard Noss, Yasmin B. Kafai, Kylie Peppler, Deborah A. Fields, Nathan Holbert</i>	
Inside the Trojan Horse – A Discussion Among the Next Generation of Constructionists	755
<i>Gary Stager, Sylvia Martinez, Amy Dugré, Angela Lombardo, Susana Tesconi, Tracy Rudzitis, Brian C. Smith, Jaymes Dec</i>	
Workshops.....	756
WS1: The Essence of Programming at School – Learning for Life.....	756
<i>Jacqueline Staub</i>	

WS2: Developing Body Tracking Software with Scratch and Kinect.....	759
<i>Stephen Howell, Lizbeth Goodman</i>	
WS3: Group-based Simulation and Modelling: Technology Supports for Social Constructionism	761
<i>Corey Brady, Walter Stroup, Tony Petrosino, Uri Wilensky</i>	
WS4: AI Programming in Snap!.....	765
<i>Ken Kahn</i>	
WS5: Developing Affordable STEM Maker Projects with BBC Micro:bits and Microsoft MakeCode	767
<i>Stephen Howell, Neeltje Berger, Peter Heldens, Kevin Marshall, Clare Riley</i>	
WS6: NetsBlox: A Constructionist Environment for Creating Distributed Applications	769
<i>Brian Broll, Corey Brady, Ákos Lédeczi</i>	
WS7: The ER4STEM Repository for Educational Robotics	773
<i>Annalise Duca, Angele Giuliano, Sofia Nikitopoulou, Nikoleta Yiannoutsou, Chronis Kynigos</i>	

Demonstrations 777

The Relationship between Computer Programming and English Language Skills	777
<i>Nevin Akcay, Hulya Avci, Ali Gungor, Tufan Adiguzel</i>	
<i>Synthesizing the Mesh: Using Constructible Authentic Representations to Gain Intuitive Understanding of Bayesian Reasoning.....</i>	781
<i>Monica Chan, Gary C. F. Lee</i>	
Teaching Computational Thinking with Minecraft & Microsoft MakeCode	786
<i>Stephen Howell</i>	
Interpolating (and Extrapolating) 3D turtle Programs in Beetle Blocks	788
<i>Ken Kahn</i>	
Hedgehog: A Versatile Controller for Educational Robotics.....	791
<i>Markus Klein, Clemens Koza, Wilfried Lepuschitz, Gottfried Koppensteiner</i>	

Working groups 794

WG1: Constructionist Approaches to Computational Thinking	794
<i>Bernhard Standl, Gerald Futschek, Jane Waite, Andrew Paul Csizmadia, Lina Vinikienė, Janne Fagerlund</i>	
WG2: Developing Constructionism, or a New Learning Concept, across the Ages	838
<i>Don Passey, Loice Victorine Atieno, Wilfried Baumann, Valentina Dagienė, Arūnas Poviliūnas</i>	
WG3: Creating and Looking at Art with Logo Eyes	855
<i>Evgenia Sendova, Christos Chytas, Katarzyna Olędzka, Ralf Romeike, Wolfgang Slany</i>	
WG4: The Constructive Strategies in Teaching Humanities with Films	868
<i>Lilija Duoblienė, Jūratė Baranova, Luc Anckaert, Wilfried Baumann</i>	
WG5: Constructionism in the Classroom: Creative Learning Activities on Computational Thinking	884
<i>Michael Weigend, Zsuzsa Pluhár, Anita Juškevičienė, Jiří Vaníček, Kazunari Ito, Igor Pesek</i>	
WG6: Learning to Program in a Constructionist Way.....	901
<i>Mattia Monga, Michael Lodi, Dario Malchiodi, Anna Morpurgo, Bernadette Spieler</i>	
WG7: Constructionism in Upper Secondary and Tertiary Levels	925
<i>Ana Isabel Sacristán, Lina Kaminskienė, Mihaela Sabin, Richard Akrofi Kwabena Baafi</i>	

Teachers' Day 940

The beauty in science and the science in beauty or mathematics, informatics and science teaching as an eye-opener of the beauty of ideas	941
<i>Evgenia Sendova</i>	
Ateities švietimas: naujos galimybės Lietuvai?	944
<i>Rimantas Želvys</i>	

WS1: Dynamic Teaching Ideas for teaching Music Theory	945
<i>Judith Bell</i>	
WS2: Computer Science Unplugged for Teachers	945
<i>Tim Bell</i>	
WS3: Developing Algebraic Habits of Mind in Students	945
<i>Paul Goldenberg, Cynthia J. Carter</i>	
WS4: Puzzles & Programming to Develop Mathematical Habits of Mind in 6–10 year Olds	946
<i>Paul Goldenberg, Cynthia J. Carter</i>	
WS5: Powerful Ideas in Lower Primary Programming: High Time to Recognize Them	946
<i>Ivan Kalaš</i>	
WS6: Snap! - Beauty & Joy of Computing (visually)	947
<i>Witek Kranas</i>	
WS7: ViLLE – E. Learning Path for Mathematics and Programming.....	954
<i>Mikko-Jussi Laakso, Petra Enges-Pyykönen</i>	
WS8: Constructionism in Action: Do we Need to Start from Scratch?	954
<i>Evgenia Sendova, Nikolina Nikolova</i>	
WS9: Teaching Coding and Physical Computing	955
<i>Gary S. Stager</i>	
WS10: The Essence of Programming at School – Logo in a Spiral Curriculum	955
<i>Jacqueline Staub</i>	
WS11: How to Create and Sustain a Progressive Pedagogy in a Traditional Setting (Roundtable Discussion)	956
<i>Carol Sperry Suziedelis</i>	
WS12: Joyful Learning of Geometry in Cultural Context. Analysis and Construction of Geometric Ornaments	956
<i>Igor Verner, Khayriah Massarwe, Daoud Bshouty</i>	

Opening / Plenaries

One Hundred Years of Educational Development in Lithuania

Rimantas Želvys rimantas.zelvys@fsf.vu.lt
Vilnius University, Lithuania

Abstract

This year we celebrate the 100th anniversary of declaring the independent Republic of Lithuania, and the main focus of our attention is the development of education since 1918, which can be divided into at least four different periods.

1918-1940. The period of building. After declaration of the independence in February 1918, the Ministry of Education was established. After the devastating World War I the Ministry discovered in the territory of the newly founded state 8 functioning gymnasiums (upper secondary schools) and 11 progymnasiums (lower secondary schools) with 360 teachers. Besides that, 1232 teachers worked in primary schools, so the whole teaching corps in the country was 1592 teachers. There were no institutions of higher education as the only university in the country – Vilnius University – was closed down in 1832. The system of education had to be built practically from nothing. Lithuanian university was re-established in 1922. Compulsory primary education – four years – was introduced during the period of 1928-1931. In two decades of intense work the fully-functioning system of education was created and the illiteracy rate in 1940 dropped down to 2 percent.

1940-1944. The period of destruction. The Soviet occupation in 1940 and the World War II had an enormous destructive effect on our educational system. Just in a single day – June 14, 1941 – 11 percent of all Lithuanian teachers were deported to Siberia. Many of the teachers died during the war, perished in exile or fled to the West in fear of repressions when the Red Army was approaching. In 1943 the Nazi German authorities closed down Vilnius University as an act of revenge for non-cooperating with the occupational administration. The system of education met the end of the World War II with demolished schools and few remaining teachers.

1944-1990. The period of adaptation. After the war the country had to adapt to the imposed Soviet model of education. Mass education was one of the essential elements of the model. In 1949 compulsory seven-year education, and in 1958 – compulsory eight-year education was introduced. In 1986 the eleven-year long general secondary education was extended to twelve-year long general secondary education. Massification of general secondary schooling was accompanied by the centralization, monopolization, unification, and, most important, strong ideologization of education.

1990-2018. The period of transformation. There are many challenges facing our education. We have to deal with an ideological challenge – what will substitute the previously imposed communist ideology? Do we accept the new “global ideology” – neoliberalism – or shall we look for something else? We face a strategic challenge – what are the long-term goals and mission of our education? Do we accept the prevailing outlook that education is a service which has to supply the global labour market with a necessary workforce, or is it something else? We face a structural challenge – due to demographic reasons is shrinking instead of expansion. We came to the understanding that closing down schools is more difficult than building the new ones. We face the challenge of economical efficiency – how to achieve the desired level of quality and equity in education at the costs which are both available and acceptable to our society? We acknowledge that the current state of our education is far from perfect and that there are still many questions to be answered and many solutions to be found. However, when we look back at the starting position we had one hundred years ago, there is no doubt that we can be really proud with what we have achieved.

CS Unplugged and Computational thinking

Tim Bell, tim.bell@canterbury.ac.nz
University of Canterbury, New Zealand

Abstract

The CS Unplugged activities (csunplugged.org) provide a scaffolding for a constructivist approach to introducing topics in computer science, without the need to learn programming first. It has been widely used to support Computational Thinking in school curricula. This paper discusses the connection between CS Unplugged activities and one of the (many!) definitions of Computational Thinking, and discusses how it should be used in this context based on research into using the Unplugged approach effectively in education. In addition to considering the value of an Unplugged approach for teaching students, we will look at broader applications, including supporting teachers who are new to the subject, and using it in integrated learning where computational thinking is exercised as part of other curriculum areas.

Keywords

CS Unplugged; computational thinking; teacher PLD; integrated learning

Introduction

Computer Science Unplugged (CSUnplugged.org) was originally intended as an outreach tool to explain computer science to young students, without the overhead of having to learn programming first. However, it is now used in a variety of contexts, and with the recent adoption of elements of computer science into school curricula around the world (Heintz et al., 2016), the Unplugged approach has often found a role in the classroom. The new curricula are commonly based around the idea of *Computational Thinking*, an idea that came to prominence after the publication of a paper by Jeanette Wing in 2006 (Wing, 2006), although the term was used by Papert as early as 1980 in his widely read “Mindstorms” book (Papert, 1980), and the general concept predates Papert’s work (Tedre and Denning 2016).

Here we look at what Computational Thinking is, how it relates to Computer Science Unplugged activities, and how this connects to previous research on the use of CS Unplugged for teachers and students.

Computational Thinking

The heading of “Computational Thinking” (CT) is commonly used in the context of introducing computer science into primary and high school education. There are varied views on what Computational Thinking is actually about, and its broad value outside of teaching computer programming, although there is general agreement that it relates to the kind of thinking students need to develop in order to effectively be able to program digital devices, that there are particular skills that are relevant, and that it has an enormous impact on the world.

To understand CT, it is helpful to focus on what *computation* is. The notion of computation and computability has been explored in depth over the years, with fundamental ideas being based on Turing’s work in the 1930s about what he called an “automatic machine” (Turing, 1937), which is now commonly referred to as a Turing machine. The limitations of a Turing machine still define the boundaries of digital computation today (with the possible exception of the new developments in quantum computing), so both the power and limitations of computing are reflected in any Turing-complete programming language, which includes many widely used educational languages such as Scratch and Python. From this point of view, it could be argued that CT is centred around learning to use such languages to their full extent (Denning, 2017), since the machines dictate exactly what can and can’t be done in computation. It is reasonable to be concerned that analogies to computation (such as cooking recipes) are imperfect and might even teach against some of the deeper principles, but we

need to acknowledge that beginners in any subject are often given simplified models to help scaffold their learning (such as the Rutherford-Bohr model of the atom, Newtonian physics, or the division of history into discrete periods).

However, it is also possible to enforce computationally authentic elements of computing without using a digital device. For example, the CS Unplugged resources (Bell et al., 2009; Bell, Rosamond and Casey 2012) have an activity on sorting algorithms (<https://csunplugged.org/sorting-algorithms>), which puts a simple rule in place that only two values can be compared at once using a balance scale, and the comparison is done by a third party so that there is no memory of previous comparisons other than placement of the weights. This forces students to explore the same kinds of algorithms that a digital device would have to use when sorting by comparison, so it isn't just an analogy, but an alternative physical implementation of the kind of computation that is possible if one could program a digital device. It has the advantage that students don't need to learn about programming first before engaging with the algorithm, although ultimately a computer program is needed to see all of the limitations encountered when implementing the algorithm.

Broader definitions of Computational Thinking have been derived based on the general skills needed to reason about computation. For example, Wing (2010) uses the definition of Computational Thinking as "... the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." This leads to a more general view of programming, where CT includes knowing how to:

- describe a problem,
- identify the important details needed to solve this problem,
- break the problem down into small, logical steps,
- use these steps to create a process (algorithm) that solves the problem,
- and then evaluate this process.

The underlying skills needed to achieve these steps have been presented in various forms, but common elements generally include some sort of variation on a list such as algorithmic thinking, abstraction, decomposition, generalization, evaluation and logic. Such lists have been produced by national organisations supporting new curricula, including Computing at School in England (Csizmadia et al, 2015) and the CSTA (CSTA, 2011).

Denning (2017) warns against overgeneralising such underlying skills; for example, decomposition can be applied to many situations, such as breaking down a large (non-computer) project into components, but in a computational context such an activity has constraints imposed by the nature of a computing environment, as well as good practice (such as decomposing a large program into modules with meaningful functions). Here we focus on applying these concepts in the context of computer science, which helps us to connect the general ideas with what they mean in a computational context.

Figure 1 shows the skill list that we have chosen to use to connect CS Unplugged to CT. They are based on a combination of lists commonly found in the literature mentioned above.

Rather than try to define these here, we will illustrate them with examples from CS Unplugged activities.

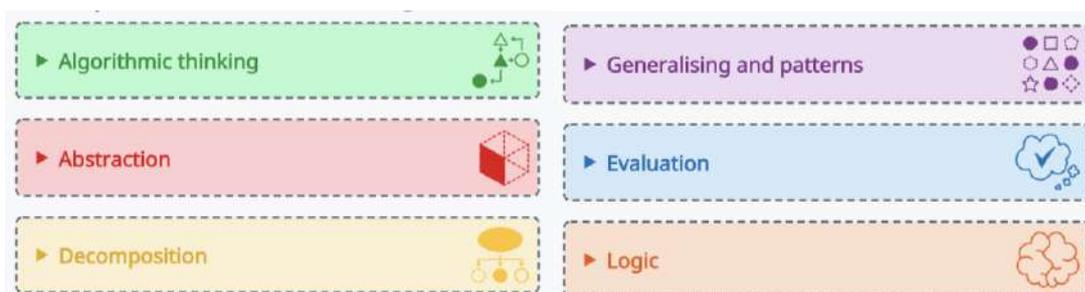


Figure 1. Computational Thinking skills

Computational Thinking and CS Unplugged

The CS Unplugged material was developed from around 1992 (Bell, Rosamond and Casey, 2012), while the concept of Computational Thinking became widely promoted after Wing's work around 2006; since CS Unplugged was intended to convey ideas from computer science, and CT has been described as "thinking like a Computer Scientist" (Wing, 2006), matching Unplugged activities with CT ideas serves as a way to show how they are both serving a similar purpose.

Here we will use three contrasting activities: the binary numbers demonstration (about data), searching (about algorithms), and sorting networks (applying a parallel algorithm to data). Note that the Unplugged activities are intended to be used in a constructivist manner, so that students are discovering patterns and rules for themselves based on a very short description of a challenge, rather than being told the rules and then applying them. This means that they are exercising the CT skills themselves as they solve the challenges that they are given.

In the *binary representation* activity, students manipulate cards that represent the powers of two (Figure 2). They follow the simple rule that the dots on a card may either be completely visible or not, and are given exercises around finding ways to display a given number of dots, counting, and exploring patterns in the representation, always with the constraint that a card is entirely visible or not (this is enforcing a constraint that a physical computing device would have). The constructivist approach means that the students are given little more instruction than the rule that each card is either visible or not; even the number of dots on each card is deduced by the students after being shown the first three.



Figure 2. The CS Unplugged binary representation activity

This activity is exercising the CT skills as follows; note that it could also be used to evaluate how well a student is applying the skills.

- **Algorithmic thinking:** Although this activity is about data, students are applying algorithms to the bits. Working out the representation of a decimal value can be done using a greedy algorithm working from left to right ("Do you want to include the 16-dot card?" etc.) Initially students may take a haphazard approach, but by scaffolding the idea of working from left to right, it becomes clear that the decisions are easy. Other algorithms that come up are incrementing by one (students can be scaffolded to find the pattern that this can be done by flipping cards from right to left until a white card comes up), doubling a value (shift left), and determining if a number is odd or even (simply check the right hand bit!)
- **Abstraction:** Although binary representations are commonly said to be made of zeroes and ones, there are no such physical digits on a computer, only abstract representations. Students can experiment with a variety of abstraction; we commonly start with "yes" and "no" for each card, but can then ask the students to be creative with other binary symbols, such as using two different musical pitches, dance moves, or even animal sounds.
- **Decomposition:** the problem of working out a number representation can be overwhelming at first, but when decomposed into a left-to-right algorithm ("do you want this card?"), it becomes a lot simpler to comprehend.
- **Generalising and patterns:** There are many patterns to be explored here; the first generalisation is working out the number of dots on the n th card, but students can also discover many other patterns, for example, that the maximum value that can be represented with k bits is one less than

the value of bit $k+1$, or that when counting, each card is being flipped with half the frequency of the one to its right.

- **Logic:** There are several rules that students can deduce using logic. One useful one is the uniqueness of a binary representation. For example, suppose they have found the representation 01001 for the number 9. The students can then be asked “is it possible to have a representation of 9 where the first bit is 1?” They will argue that it’s not possible because you would have 16 dots – too many. The first bit *must* be 0. Then ask if the second bit could be a 0. Students will soon realise that there are only 7 dots left in that case, and can argue themselves that the second bit *must* be 1. This reasoning can be applied to all the bits of any binary representation, and students will have created an informal proof that a particular value has a unique representation.
- **Evaluation:** Being able to convert between decimal and binary numbers isn’t a widely used skill, but being able to *evaluate* the limits of a representation is. For example, students can evaluate the largest number possible with, say, 5 bits, and then with 6 bits, and with scaffolding, realise that each extra bit doubles the range of possibilities. This can lead to reasoning about the effectiveness of, say, a 256-bit security key vs. a 512-bit key; or an 8-bit character representation vs. 16-bit. In both cases the increase in representation is considerably more than the factor of 2 that might appear on the surface.

In the *searching algorithms* activity, students are given hidden values to search in several contexts: it could be hidden “treasure chests” held by a friend who will reveal the contents only one chest at a time (Figure 3), or cups that have values hidden under them, which can only be turned over one at a time. In both cases, the goal is to find a value without looking at more items than necessary. At first the values are unsorted, but students are later given a series of values that are sorted, which they can constructively use to apply a form of binary search to avoid frustratingly long searches.

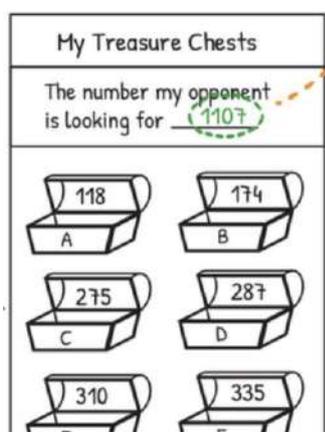


Figure 3. The CS Unplugged treasure hunt searching activity

Computational thinking appears in the searching activities as follows.

- **Algorithmic thinking:** Students constructively discover a (variation of) the binary search algorithm motivated by minimising the cost to them of finding a given object or value.
- **Abstraction:** The values being searched (keys) are usually an abstraction of some item that needs to be found, such as a person’s name being searched to find some information associated with them. Working with the search key is sufficient to understand the algorithm.
- **Decomposition:** Each key comparison is decomposing the solution space into smaller parts; in the case of a sequential search it is a small gain (the solution space is reduced one by one), but students can explore the power of divide-and-conquer through binary search, where a half of the problem space is eliminated in one step.
- **Generalising and patterns:** The different guises of searching (treasure chests, cups, envelopes and so on) are all the same problem with the same possible solutions, but presented in different ways. This enables students to recognise what the general algorithm is, rather than just a specific application of it.

- **Logic:** There are a number of ideas that can be reasoned about here: for example, that binary search can only work on a sorted list, and that binary search is guaranteed to find what the student is looking for even though many items are never inspected.
- **Evaluation:** This is a key reason for students to explore algorithms; sequential and binary search have quite different performance, and although students may not use mathematical language like “logarithms”, they can appreciate that even billions of items can be searched in a very small time with binary search, and that sequential search becomes arbitrarily worse than binary search as the size of the list increases.

In the *sorting network* activity, students traverse a network drawn on the ground, making a simple comparison of values at each node and taking the left or right exit based on the comparison (Figure 4). Again, the instructions given are very simple (compare values and go left or right), but the activity allows the students to construct a range of understandings based on their experience.



Figure 4. The CS Unplugged sorting network activity

- **Algorithmic thinking:** The students are physically engaging with a parallel algorithm, and seeing how a complex outcome (sorting) can be achieved by the combination of many simple steps (in this case, comparisons of pairs of values). They also have the opportunity to design their own parallel algorithms for smaller sorting networks.
- **Abstraction:** The sorting network is a physical representation of what happens inside a computer. The values being compared (keys) are also an abstraction of some item that is being sorted, which may have more data than just the sort key.
- **Decomposition:** A key aspect of this activity is that the complex task is decomposed into a very simply described task. When comparing words or large numbers, that is further decomposed into a character-by-character comparison to determine which value comes first.
- **Generalising and patterns:** The sorting network can be used to sort any values that can be compared for order; numbers are in order of increasing value, while words are in alphabetical order, but the comparisons can also be used for other types of data, such as musical notes (higher and lower), and stories (which plot element comes before another?).
- **Logic:** Students are able to reason about the correctness of the configuration by applying logic (an exhaustive test would take $n!$ time). A first step is to apply logic to reason that the smallest item must end up in the correct place, regardless of where it starts. A full proof of correctness is likely beyond students, but for small sorting networks there is a lot of opportunity to reason about what will happen.
- **Evaluation:** A sorting network can be evaluated in terms of the number of nodes required (three at a time in the case of a 6-way network), but also in terms of the number of parallel steps required (the length of the network, and therefore time required). Two different networks for the same number of inputs can be compared based on these metrics.

These activities have been used as examples; on the CS Unplugged website these CT skills are made explicit for every activity to help teachers see the bigger picture of why a particular activity is relevant, and also to appreciate which finer details of an activity are important to fully engage students in CT.

Integrated learning

Computer science isn't an end in itself, and is used in many practical contexts. When taught in schools it can be used effectively in integrated contexts, where the concept is applied to other subjects to support learning in both at the same time, even with an Unplugged approach.

For example, the binary representation activity includes the possibility of threading beads of two colours into bracelets, necklaces or bag-tags. Making up chains of beads gives students the chance to think about language, and what they would like to communicate with the beads; it also happens to exercises fine motor skills. More generally, the activity can be extended to art, where two symbols or images are used to embed information in a picture; or music, where the two values can be used as note pitches or lengths. Both of these can be used to introduce the idea of steganography, where a message is communicated in plain sight through an artefact that appears to have a different purpose. Topics like binary numbers can also be integrated with history and writing – where did the idea come from, and how have people communicated in the past over distance? Looking into representations like Braille and Morse code can reveal how communication has influenced history, but also how it is natural for humans to develop codes for communication over distance or for efficient storage. Students can construct their own codes based on their new understanding, and this provides a richer experience than simply learning standard codes (such as ASCII and Unicode), as they will face the questions that arise for themselves, such as special characters, using digits as text, and so on.

Sorting networks can be integrated naturally with other curriculum topics, and might be used to compare dates in history, words in alphabetical order, note pitches in music, or numbers written in a foreign language. They provide motivation for students to repeatedly compare the values that they are learning, and to see them in situations other than the sequence normally presented. At the same time, they are becoming familiar with a computational model.

Searching algorithms can also be explored in terms of history – how did people look up information in pre-computer times, and who had access to such information? Who are the people who developed these computer algorithms, and what motivated them? There is also the possibility of acting out such algorithms; and a binary search can even be used to compare an unknown pitch with the notes on the piano to determine what it is.

Applying CS Unplugged

An important element of this style of teaching is to give minimal instructions, and allow students to construct the knowledge for themselves. Once they have done this, it is important to then relate what they have done to the broader context of computing, and what happens on physical devices. Two early studies discovered that without this connection “the program [based on CS Unplugged] had no statistically significant impact on student attitudes toward computer science or perceived content understanding” (Feaster et al., 2011) and that “the students’ attitudes and intentions regarding CS did not change in the desired direction” (Taub et al., 2012). In terms of conveying knowledge using this approach compared with a more conventional approaches, Thies and Vahrenhold (2013) found that “... it is indeed possible to weave Computer Science Unplugged activities into lower secondary Computer science classes without a negative effect on factual, procedural, or conceptual knowledge”, and that it could have some benefit in that “the Computer Science Unplugged materials can prove helpful for ability grouping within a class, since, on average, more students are enabled to reach a higher operational stage.”

Gains from using CS Unplugged were reported by Hermans and Aivaloglou (2017), who combined it with teaching programming for one group, while having a second group spend the same total amount of time learning *only* programming; they found that “...the group taught using CS Unplugged material showed higher self-efficacy and used a wider vocabulary of Scratch blocks.”

Looking at these different contexts, we see that CS Unplugged is best used in combination with “plugged in” work. This is not surprising, given that getting a program to work correctly is an excellent way for a student to show that they have understood the computational concepts they are working with, since the computational agent (the computer running the program) will do exactly what the program says to do.

Based on this, the CS Unplugged website now offers a range of “Plugging it in” exercises to provide follow-up activities that allow students to link their Unplugged learning with computation on a digital device.

An Unplugged approach seems to have promise for helping student learning if used effectively, but another important value of it is for *teachers*. It is important for teachers to be confident in a topic so that they can build student confidence, and given that the new computing curricula appearing around the world are often taught by people new to the subject, ways to build teacher confidence will be important (Gutiérrez and Sanders, 2009). CS Unplugged has been used in a variety of teacher professional learning and development (PLD) initiatives, and the research available on this is reporting positive outcomes. For example, Curzon et al. (2014) report on teacher professional development that had a substantial “Unplugged” component, and noted that it was “inspiring, confidence building and gave [the teachers] a greater understanding of the concepts involved.” An important feature of the constructivist approach of Unplugged activities is that they allow very quick wins, where teachers can understand a new concept (such as binary numbers) very quickly, without the overhead of having to learn to program first. Smith et al. (2015) reported that teachers who were training other teachers (through the UK Master teachers system) commonly included CS Unplugged when providing professional development for colleagues, and both Morreale and Joiner (2011) and Sentance and Csizmadia (2017) found that after attending their workshop, CS Unplugged was widely adopted by teachers.

Conclusion

Computer Science Unplugged activities support computational thinking, although for this to be effective they should be used in a context where they will be linked to implementation on a digital device. By using Unplugged early to introduce concepts, both students and teachers new to the subject can have early success without the overhead of becoming proficient enough at programming to engage properly with ideas that can have an impact in our digital world. This then provides a useful platform to motivate learning the skill of programming, but also a way to connect computer science with other subjects.

References

- Bell, T., Alexander, J., Freeman, I., and Grimley, M. (2009). Computer science unplugged: school students doing real Computing without computers. *New Zealand Journal of Applied Computing and Information Technology*, 13(1), 20-29.
- Bell, T., Rosamond, F., and Casey, N. (2012). Computer Science Unplugged and related projects in math and computer science popularization. In H. L. Bodlaender, R. Downey, F. V Fomin, and D. Marx (Eds.), *The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, LNCS 7370, pp. 398–456. Springer-Verlag.
- Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., and Woollard, J. (2015). Computational thinking: a guide for teachers. Available from <http://computingatschool.org.uk/computationalthinking>.
- CSTA. Operational Definition of Computational Thinking. 2011; <https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CompThinkingFlyer.pdf>
- Curzon, P., McOwan, P. W., Plant, N., and Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education – WiPSCE '14*, 89–92.
- Denning, P. (2017) Remaining trouble spots with computational thinking, *Communications of the ACM*. 60(6): 33-39, June.
- Feaster, Y., Segars, L., Wahba, S. K., and Hallstrom, J. O. (2011). Teaching CS unplugged in the high school (with limited success). In G. Rößling, T. L. Naps, and C. Spannagel (Eds.), *Proceedings of the 16th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2011*, Darmstadt, Germany, June 27-29, 2011 (pp. 248–252). ACM.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful ideas*. Basic Books.

Gutiérrez, J. M., and Sanders, I. D. (2009). Computer Science education in Perú: a new kind of monster? *ACM SIGCSE Bulletin*, 41(2), 86–89.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *Proceedings – Frontiers in Education Conference (FIE)* (pp. 1–9).

Hermans, F., and Aivaloglou, E. (2017). To Scratch or Not to Scratch?: A Controlled Experiment Comparing Plugged First and Unplugged First Programming Lessons. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 49–56). New York, NY, USA: ACM. <http://doi.org/10.1145/3137065.3137072>

Morreale, P., and Joiner, D. (2011). Reaching future computer scientists. *Communications of the ACM*, 54(4), 121.

Sentance, S., and Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2), 469–495. <http://doi.org/10.1007/s10639-016-9482-0>

Smith, N., Allsop, Y., Caldwell, H., Hill, D., Dimitriadi, Y., and Csizmadia, A. P. (2015). Master teachers in computing: What have we achieved? In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 21–24).

Taub, R., Armoni, M., and Ben-Ari, M. (2012). CS Unplugged and Middle-School Students' Views, Attitudes, and Intentions Regarding CS. *Trans. Comput. Educ.*, 12(2), 8:1–8:29. <http://doi.org/10.1145/2160547.2160551>

Tedre, M., and Denning, P. J. (2016). The Long Quest for Computational Thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research*, pp. 120–129.

Thies, R., and Vahrenhold, J. (2013). On Plugging “Unplugged” into CS Classes. In *SIGCSE '13: Proceedings of the 44th ACM technical symposium on Computer Science Education* (pp. 365–370).

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1), 230-265.

Wing, J., (2006) Computational Thinking, *Communications of the ACM*. 49 (3)

Wing, J. M. (2010). Computational Thinking: What and Why? <http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>.

Constructionism Won, Now What? The Role of Constructionist Research in the Age of Ubiquitous Computing

Paulo Blikstein, paulob@stanford.edu
Stanford University, USA

Abstract

Logo was created almost exactly 50 years ago. It might be the right time to take stock on the accomplishments of the last five decades, and the possible directions for the future. The Constructionist community, having rebelliousness in its DNA, has grown used to say that “the revolution has not come yet.” We are still far from realizing Papert’s vision, but in the last 20 years there has been impressive change in schools and in the discourse around educational innovation.

The first change is on computing itself. Several high-profile initiatives have brought coding into the mainstream of education, with several cities and countries advocating and implementing programming as a mandatory topic in pre-college education. The scale of these initiatives is impressive, and even though they are still in early stages, they represent a clear recognition that coding has finally been accepted as a school topic at the highest levels of policy making.

Makerspaces and fab labs are a second phenomenon that has reached surprising popularity in schools in just a few years. Thousands of schools already have well-equipped makerspaces, and even though access is not equitable in many of them, their mere presence in schools point to a crucial recognition of the value of constructionist pedagogies, creativity, student agency, and construction.

Third, even traditional disciplines and national standards are being “infected” by the constructionist virus: many science and math curricula around the world now employ constructionist-inspired pedagogies and principles, and some go as far as incorporating tools such as computational modeling and sensing to science classes. And this is also happening at the national level: for example, in the United States, the Next Generation Science Standards made engineering and design mandatory in basic education. And finally, Constructionist tools such as Scratch, NetLogo, Lego Robotics, GoGo Boards, and the Lilypad, have become much more robust and been use by millions of children worldwide.

Given all the good news, what is the right reaction from this community? Claim “mission accomplished” or double down our efforts? It seems that the main challenge for the next 50 years will not anymore convincing schools that many of these technologies and approaches are useful and effective but will be concentrated in two clusters: (a) Making sure that those new learning opportunities are offered to students with equity, and (b) Battling the forces of trivialization, that for economic or ideological reasons, often try to overly simplify the technologies and methods of Constructionism. To overcome these challenges in the next 50 years, we will have to find innovative forms of doing research in learning, new avenues for public advocacy, and novel ways to reach students.

Keywords

Logo; constructionism; computing education; maker education.

Look Closely, Watch What Happens: Visual Modelling and Constructionism

James Clayson, james@clayson.org
American University of Paris, France

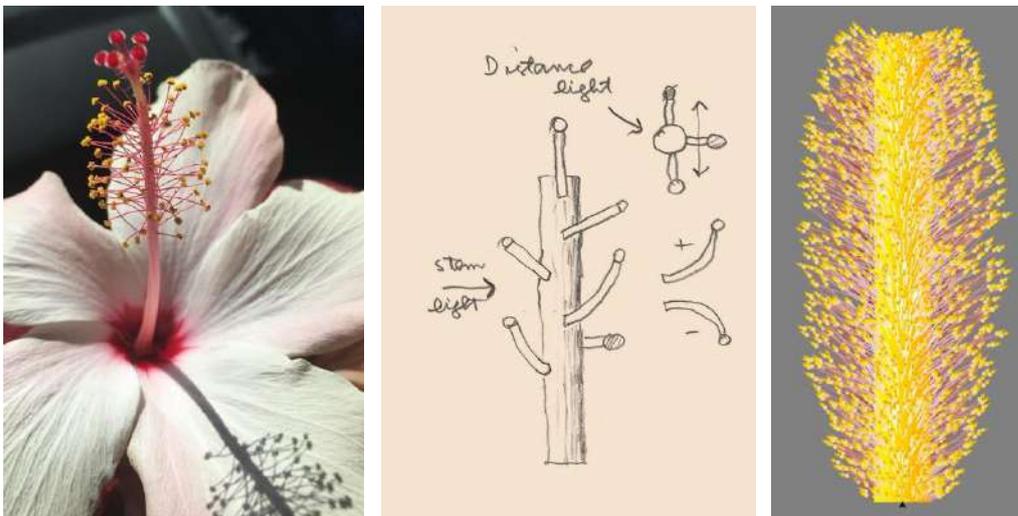
Abstract

The core activity in my approach to visual modelling is a series of exercises that encourage students to embark on individual computational explorations of shape, placement, color and texture themes by looking closely at physical objects meaningful to them. In this paper, I will describe the journey that led me to designing and teaching courses around this central notion and how such modelling works, encouraging students to integrate a wide range of technical and non-technical skills into their work. I will talk about why I think this activity is important for students in all disciplines, including the liberal arts. I will talk specifically about the effects visual modelling has had on my students and I will also describe how the teaching and practice of visual modelling have informed my own development and learning over the past 40 years.

My reasons for integrating greater visual imagery into computational activities are summarized in the form of 13 arguments.

Keywords

constructionism; transformational objects; visual modelling; visual thinking; computational tools; liberal arts; Logo; Python



Example: Look, record, see what happens

Look closely, watch what happens: visual modelling and constructionism

I believe that the image is the great instrument of instruction. What a child gets out of any subject presented to him is simply the images which he himself forms with regard to it.

I believe that if nine tenths of the energy at present directed towards making the child learn certain things, were spent in seeing to it that the child was forming proper images, the work of instruction would be indefinitely facilitated.

I believe that much of the time and attention now given to the preparation and presentation of lessons might be more wisely and profitably expended in training the child's power of imagery and in seeing to it that he was continually forming definite, vivid, and growing images of the various subjects with which he comes in contact in his experience.

John Dewey

... concreteness is not a property of an object but rather a property of a person's relationship to an object. ... The more connections we make between an object and other objects, the more concrete it becomes for us. The richer the set of representations of the object, the more ways we have of interacting with it, the more concrete it is for us.

Uri Wilensky

My personal visual modelling journey

After graduate school I worked in industry for ten years as a manager of operations research (OR). Then I moved to France where I discovered both teaching and the power of imagery and visual literacy for constructionist learning. I managed an art school in the south of France for five summers. There, I took many of the foundation studio courses in art and design which eventually informed my whole teaching approach and led to a radical reexamination of pedagogical styles and cognition. I was already in my early forties when I was hired by the American University of Paris (AUP) to teach OR.

I found my AUP students to be highly motivated, extremely verbal in several languages, and street wise. But they had great trouble identifying appropriate ways for exploring large and messy problems whose parts and relationships were not precisely knowable. Of course, this wasn't too surprising. My operations research students – who were economics, business and computer science majors - had taken scores of separate courses but few truly interdisciplinary ones. They had difficulty integrating the quantitative with the qualitative and breaking large problems down into the smaller parts that could provide a starting point for coming to grips with the whole. They didn't know how to diagram relationships since they had no visual vocabularies. Few of them had ever taken a studio arts class. They weren't yet in possession of personal repertoires of problem-solving techniques. They didn't know the value of speaking with others to help clarify ambiguous tasks and, when this did happen, they didn't yet appreciate the importance of recording verbal clips of shared insight. They were totally unaware that just talking could become a starting point for figuring out difficult problems and that often they didn't need new tools to do so.

One day I was speaking to my colleague, Roger Shephard, who was director of Parsons School of Art in Paris. Roger taught painting and drawing. Unlike my students, he said, his students had no trouble in talking about ambiguity and multiple viewpoints. They all kept journals in which they recorded talk, ideas, diagrams, sketches ... their work process. They loved deconstructing and reconstructing, cutting and pasting. But, he said, they were distrustful of and resistant to the idea that conforming to a more structured plan could be useful to their work. For them everything seemed open for further exploration. But Roger wanted these art students to see some closure, to produce more individual pieces of work that were, if not "finished", then at least a summary of their author's current state. Many of these art students did have some math experience but felt ill at ease with it. And they would often, almost ideologically, reject using whatever math they did know.

That's when Roger and I had the fun idea to put our students together and watch what happened! We paired my economics, business and computer science majors with Roger's art students. We called our experimental course, *Problems in Visual Thinking*. The central constructionist twist and ideological engine for this course was model building with Logo. The class went on for 5 years and was the basis of my book on problems in visual thinking (Clayson 1988). After this stimulating initial experience, I later revised and extended my visual modelling ideas for a team-taught course with AUP colleague and artist, Ralph Petty, that would be open to AUP students from any discipline. Our collaboration lasted 10 years from 2000 through 2009. (Clayson 2007, 2008).

Most recently I translated all my Logo materials from these earlier years into Python and I wrote a new text for a course I gave at Deep Springs College in California where I was a visiting scholar in 2015.

(Clayson 2015). I built, with the help of colleagues from Comenius University (Slovakia), a number of Python modules that let me be “Pythonic” in my own visual and constructionist style. The translation from Logo to Python was a fascinating and thought-provoking exercise that I hope to describe elsewhere.

A closer look at what happened and what I learned

For thirty-five years I encouraged students to look more closely at favored objects in their lives and to record what happened and what they felt when they did this. Computational methods were the necessary catalysts for this visual activity, but they were not sufficient. In the end, it was the critical mixture of qualitative, visual and quantitative methods that lead to students’ seeing more clearly. Seeing clearly provided them an intense emotional and intellectual satisfaction that lasted long after the course was over.

Having kept my own teaching journal throughout these years, I have been able to distill my experience into 13 arguments for adopting this visual approach to modelling. In the constructionist spirit of putting ideas into the public space, I now share them with you. I hope my personal journey as a constructionist teacher with visual affinities might inspire others to look more closely at nearby objects and carefully watch what happens.

Arguments for doing visual modelling

1. The **aesthetic argument**. When we model an object in order to bring it down to a size that we can hold, to view it from all sides, this can give us enormous aesthetic pleasure. Whether the model is a miniature flashing Eiffel tower, a ship model, an embroidered flower or a stage set, we can sense its wholeness, its complexity, without dwelling on individual parts. The model, of course, will be an abstraction, a simplification of the whole. Model authenticity can be measured in the pleasure it gives us. The simplicity of our model illustrates a restructuring of the modelled thing; we can feel it. Levi-Strauss talks about turning physical dimensions into “intelligible dimensions”. (1966). We could also call them meaningful dimensions. Seeing one object more meaningfully through our interaction with it is a skill that can easily be transferred to other objects. (Bateson 1972)

2. The **design literacy argument**. Two-dimensional graphic design explores the aesthetic, emotional and communication inherent in compositions of lines, shapes, typography, signs, symbols, color, texture and depth cues placed on a canvas, page or website. Typically, basic design courses introduce design theory through a structured set of exercises resembling my own constructionist approach (Wilde 1991). Unfortunately, most university students do not have the opportunity to learn anything about art or design concepts, nor to gain insight into how design might be useful if applied to other fields. Visual modelling offers a crash course introduction to the language and tools of design.

3. The **situated computational thinking argument**. There is a public domain book called “How to Think Like a Computer Scientist” (Wentworth 2018) that is considered to be a handy introduction to Python programming. The question is, handy for whom? It is useful to me because I trained as an engineer, have studied computer science, have learned to program in many different languages and have taught applied mathematics. In other words, it is useful because I already know how to think like a computer scientist. This is not the case, however, with most of my visual modelling students. This book, while useful to me, is therefore totally useless and inappropriate for most liberal arts undergraduates. My students already have a personal epistemology. They want to find and use tools that they feel comfortable with that can immediately help them to get on with their own idiosyncratic modes of inquiry (Minsky 1988, Kelly 1955). They don’t want to have to become someone else first. And why should they?

What are some of the simpler more accessible computational tools these students might find useful? Design manipulations like deconstruction/reconstruction, replication, scaling, generating random components, perspective and color operations cry out for computational constructs and I introduce these in class via my solutions to specific design problems. Each suggested computer construct is situated within a series of structured visual tasks. Afterwards, some students may be intrigued enough to look at the programming text mentioned. But for most, reading a text of new material can be extremely hard-

going unless they already have some familiarity with the content. Guide books, as we all know, generally make more sense after the trip than before.

Over the years I have found that recursion, for example, appeals greatly to some non-technical students, especially artists, who want to immediately play with it. Recursion seems to strike at something they feel emotionally at ease with. Linking design and illustration tasks with computer constructs that operationalize these tasks extends the meanings of both design and computational tools. New tools can suggest new design approaches.

There is one big idea from computer science that does stand out in my mind as being personally empowering for most students: simulation. Simulation is the act of turning ideas into computer code that can be manipulated and experimented on. Students can actually watch what happens when they do this and are therefore motivated to continue their explorations. I have found that working in an environment of visualization makes simulation even more potent. Students see what a program, what an idea means by watching what it does. And they don't have to be computer experts to do so.

Proponents of computational thinking often talk about the "power of abstraction" that is implicit in computer programming. I have found that this powerful idea resonates best when abstractions from one medium are compared with those from another. The sketch of an object, or a poem based on it, after all, are abstractions. But it is the comparison between the drawn and the spoken and the programmed abstraction that is the essence of visual modelling.

4. The body syntonicity argument. The word syntonicity was coined in the 1800s to describe alternative musical tuning systems. When two instruments were heard to be in harmony with each other, they were judged to be syntonic. This meaning was later extended to describe individuals whose emotions were in tune with their environment. Freud extended syntonicity's use to his system: a person's manner was ego-syntonic if it supported the needs and desires of their egos.

Papert's notion of body-syntonicity (1982) assumes that all people have a number of separate areas for rational, psychological and physical reasoning. Parts of the body have, in effect, their own reasoning, knowledge and skills to which the mind in the head may or may not have access. Papert's incorporation of turtle graphics into his Logo language was intended to open young programmers to affordances in tune with their body languages.

Papert hoped that turtle geometry would radicalize the way geometry is taught. This has not happened. Nevertheless, I have found that basing my visual modelling classes on turtle geometry does radicalize the way geometry is remembered and rethought for use in physical tasks. The real value added seems to be the pleasure of being able to do this, to remember, to rethink and use something only vaguely remembered. To make known what was not yet known.

There are, of course, other body sources of knowledge that Papert did not seem much interested in: the eye, ear, hand and voice (Arnheim 1969). In visual modelling these resources all have a role to play, and all must be called upon in order to look closely.

5. The meaning-making bricolage argument. The anthropologist Levi-Strauss used the term "bricolage" to describe a direct approach to problem solving, repair work and thinking (1966). Levi-Strauss studied pre-modern societies, but his ideas are remarkably contemporary. The bricoleur acts quickly using notions, improvisations and tools that are already at hand. Speed is often important. We can think of the bricoleur as a repair person who carries around a bag of tools that can be used on the spot. The repair plan often emerges from the doing itself, through iteration.

Visual modelling uses bricolage tools often overlooked in higher education: learning to talk and write rapidly about what we see, making quick diagrams of the structures we observe and the ideas we have about them. I discovered that when students act like the bricoleur, and have to improvise quickly, pieces of remembered algebra, geometry and trigonometry often pop out. They will test if and how these mathematical notions might help in solving some visual task. Often this means relearning the half-remembered math. Often it means learning the math for the first time, but in a context more meaningful to the student.

6. The **thinking journal argument**. Papert has warned us that the phrase “thinking about thinking” is not very useful. Rather, he advises, we should think about a specific person thinking about specific tasks. In this regard, constructionists suggest not only that teachers should focus on individual learners, but that students, in turn, should focus on their own thinking processes in specific contexts. How students learn to do this, to watch themselves in the act of meaning-making, is a critical part of the constructionist project.

Unless meaning-makers can watch themselves in action, see how they dialogue with themselves, view how they share their own meaning-making activities with others, how can they study themselves formally? Visual modelling requires that modellers keep a journal of their modelling activities: the words, sketches, codes, code play/code change. Journaling then is the trace of these activities, a trace of thinking. We need to catch and record these acts of thinking so that we and others are able to reflect upon them later. (Clayson 2015)

7. The **extreme graphical distinguishability of visual modelling as a constructionist tool argument**. Perhaps the most obvious feature of visual modelling – the sheer physicality of the target subject – is not fully appreciated. The target is viewable; the modelling methods are viewable; the images generated by the modelling process are viewable. Visual comparisons between the target subject and the model are easy to make without a lot of abstract analysis. This viewability is not so true in more abstract math courses. Tangibility encourages fuller emotional and intellectual participation from students having different skills and interests

8. The **not-like-other courses argument**. Papert warned about the difficulties of inserting Logo philosophy into traditional academic courses, especially math ones. He knew how hard it is to change educational institutions and therefore advocated for doing something totally new instead. In the early 1980s when I first taught formal computational modelling, I deliberately refrained from labelling it either a math or a computer science course. Instead I focused on the design aspect of visual modelling, a non-traditional academic approach. I structured and marketed these courses to appeal to a variety of different student majors and interests: from studio arts, though humanities to the social sciences and physical sciences.

9. The **emotionally comfortable vocational argument**. Visual modelling is a multidimensional introduction to design theory, computational programming, formal reflection on thinking, careful and clear observational techniques, and effective journaling. I think it is important to note that a visual modelling approach also seems to help many students who have suffered unhappy experiences with math courses in the past. With visual modelling they gain math agency because they are able to use mathematical notions to see and find new meaning in their own physical worlds. The freedom and ease of mixing personal math knowledge with other disciplines is an extremely useful vocational outcome.

10. The **power of computational explorations of patterns seen in concrete objects argument**. It is obvious that turtle graphics cannot depict objects like a photo would. All media have this limitation and the “modern” development of abstraction thrives on it. Different media facilitate the showing of different aspects, different themes seen in objects, as well as indicating how the modeller feels about looking at those assemblages of themes.

Visual modelling introduces the notion of looking closely at works of art, as well as other objects around us, by modelling their themes algorithmically. But finding characteristic themes requires learning a new vocabulary appropriate for describing and interpreting art. In my classes we show each other specific examples, often using museum postcards, from different abstractionist schools of painting and learn to describe them. Various methods for deconstructing or reconfiguring objects into themes are portrayed and discussed. Alternative approaches to modelling similar objects taken by different artists are examined. Using cubist, impressionist, fauvist, pointillist, supremacist or abstract expressionist techniques can suggest different and surprising ways to configure and display even the most mundane of objects as well as seeing the surprising complexity of all shapes. (Clayson 1985, 2007, 2008, 2013). I argue that looking closely at works of art, both abstract and realistic, can affect how we look at and relate to objects around us. The reverse is also true.

11. The **tricking the ego argument**. We see what we expect to see and usually that is what we have already seen before. We scan fields of things, but often do not look closely at individual items, especially

if they are thought to be already familiar. We may hesitate from taking a visual arts class because we think “I can’t draw” because we have never learned how to slow down enough to look at things closely and carefully. Yet everybody is capable of drawing, so why are we so hesitant? It’s because we hear the little voice of our ego warning us that the experience might be embarrassing.

Visual modelling offers an alternative and tricky means of settling down and looking at concrete things that is not like other visual arts. Paradoxically, it shows us that in order to slow down we have to work faster. We are drawn-in to the object-subject without realizing it: by talking very quickly about what we see, writing about it, sketching, coding and playing with that code. Fast, without reflection. Something surprising always happens during this experience. Ironically, using such a simple and limited medium as turtle graphics, actually gives us a freedom and willingness to draw that more sophisticated tools often inhibit. It is exciting and empowering, like the child’s uninhibited use of crayons on blank paper. The slowing down trick is accomplished by working fast with what we have at hand. Sketch fast, talk fast; then code it; not the reverse. The trick is to break the pattern of seeing only what we anticipate seeing. Visual modelling is full of tricks and surprises, so the results cannot be anticipated. It’s fun.

12. The looking for new transformational objects argument. In his book “Mindstorms”, Papert concentrated on telling us about his own transformational experiences with model car gears and how he used Logo to seek out other transformational occasions. But he didn’t say why it might be important for each of us to recall our own transformative events and how we also might draw energy from them. (1982) Christopher Bollas (1987,1992) argues convincingly that people are designed to continue searching for transformative objects over their entire lifetime. Unfortunately, the frenzy of our adult lives inhibits us from coming upon them naturally. Visual modelling helps restore students’ ability to find and exploit transformative objects in their local surroundings for personal pleasure and development.

Sherry Turkle is another person who has documented the primal importance of transformative moments in human existence. She has assembled several collections of evocative personal narratives about transformative objects that all constructionists should read (2007, 2011).

13. The different modes and different points of view argument. Good problem solvers know how to structure and then restructure situations in alternative ways. Statisticians know that multiple approaches each based on unique methods is the creative way to explore a data set. Art students know to move around a figure or still life, sketching and evaluating the model at each location before they settle down to work in more detail. But then they go back, change position, and look at other possibilities; nothing is really definitive, and each perspective offers its own rewards. Using different drawing tools – pencil, pen, chalk, charcoal – helps expand their ability to reconceptualize. So can art students’ weird and non-conventional formulations. They paint it yellow; turn it upside down; push the arguments beyond the realistic; put a cloth over their subject and draw that; change the music; turn off the lights, use a flashlight ... and watch what happens!

Meta-texts versus texts

Constructionists talk a lot about how we learn: is knowledge transmitted or is it constructed? Most of the constructionists I know favor a far more nuanced approach to this discussion. Whatever knowledge building/acquisition is, it is certainly not explainable in binary terms. Such reductionism just doesn’t suit human diversity. Thus, a variety of learning modes is always preferable. Throughout my teaching career I have found that a combination of instructionist and constructionist approaches works best. This seems natural to me but is not always natural to my students – especially because of the dominance of instructionist education in the world and resulting aversion to ambiguity and risk. To address this student anxiety, I try to show by example why and how different approaches to a single problem may all be useful, each in its own way. For every course I teach, I write what I call a meta-text – as opposed to a standardized text. My meta-text is designed to show how I have gone about learning the material in our course and suggests that students use it as a starting point to create and record their own experiences too.

The most important text in many subjects is the one written by the students themselves in their own language and style. As a constructionist I know that agency is best gained by having formally constructed a written document so that it can be read and modified over time. The student as author is

free to make additions and changes to their own text, and in my classes, they are required to share this record with others who are also writing their own texts.

This record keeping is what I mean by “journaling”. But how do students get started with such a journaling exercise? Not, I think, by following a fixed set of rules, but rather through examples in the context of the course material being explored. We talk in class about the different ways of doing this, and we check in throughout the semester to compare notes. Therefore, my meta-texts include my own journal entries of how I explored the same task that I set for my students, along with illustrations of other approaches that may also represent student work.

Conclusion

This paper has described my own approach to constructionist practice in the classroom in the form of thirteen arguments. The approach and lessons learned have emerged over 40 years of teaching. The introduction of visual thinking and journaling as essential elements of learning may make me a bit of an outlier here in a community mostly addressing math and science teachers, not those in the humanities and liberal arts. It seems only right, therefore, that I propose some assessment of my radical project and its outcomes.

I offer four criteria:

- 1) **Carry-over agency:** the students who had taken my visual modelling class were far more effective in my traditional applied mathematics courses than those who had not done so.
- 2) **Student testimonies:** students have written to me years later to say they identify my visual modelling course as a major learning event in their undergraduate life.
- 3) **Student feedback:** I won many teaching awards for my work.
- 4) **My own thinking process:** I look at the world far differently because of the work done in visual modelling. I can actually see the changes in the journals that I have kept for those many years.

Looking back, I see that my journey has been surprising, taking unexpected turns and new directions. I certainly wasn't planning to become a teacher. At each new juncture there was a transformative event or object – teacher, book, conversation or conference – that pointed the way forward. And always, it seems, there was a constructionist thread: a need to explore, to learn, to model and to build a common project along with my students. The relationship has been reciprocal: we have all learned from one another, but each in our own way.

In today's world there is a tendency to look without seeing. We are too busy, moving too fast, too overstimulated by digital interfaces and gadgets. This is such a pity. Seeing is not only one of life's great pleasures, but also a powerful instrument for learning and understanding. We need to remind ourselves that in human evolution, the eye, the hand and the brain developed together. This paper has proposed one way that visual literacy can inform, augment and enrich the constructionist enterprise. If you take the trouble to look closely you'll be amazed at what you see. It's magic!

References

- Arnheim, R. (1969). *Visual thinking*, University of California Press, London. 13-36
- Bateson, G. (1972). *Deuterolearning, Steps to an ecology of mind*, U. of Chicago Press, London.
- Bollas, C. (1987). *The shadow of the object: psychoanalysis of the unthought known*, Free Association Books, London, 15ff
- Bollas, C. (1992). *Being a character: psychoanalysis and self-experience*, Hill & Wang, New York.
- Clayson, J. (1985), *Visual modelling with Logo: a structured approach to seeing*, MIT Press, Cambridge.
- Clayson, J. (2007), *Radical bricolage: making the liberal arts coherent*, Plenary, EuroLogo, 2007, Bratislava.

- Clayson, J. (2008), Radical bricolage: building coherence in the liberal arts using art, modelling and language, *International Journal of Education through Art*, 4:2, 141-161
- Clayson, J. (2013), Talking statistics/talking ourselves: some constructionist lessons from the work of George Kelly, *Technology, Knowledge and Learning*. 18, 181-199
- Clayson, J. (2015), A computational eye: visual modelling with Python, Deep Springs College. Bishop, California.
- Dewey, J. (1897), My pedagogic creed, *The School Journal*, Volume LIV, Number 3, 77-80
- Kelly, G. (1955) The psychology of personal constructs. Vol. I, II. Norton, New York.
- Levi-Strauss, C. (1966), The savage mind, Weidenfeld and Nicholson, London. 1-33.
- Papert, S. (1982), *Mindstorms: Children, Computers and Powerful Ideas*, Perseus Books, Jackson.
- Minsky, M. (1988). Chapter 10: Papert's principle, *The Society of Mind*, Simon & Schuster, New York. 98–107. (Some of the most crucial steps in mental growth are based not simply on acquiring new skills, but on acquiring new administrative ways to use what one already knows).
- Turkle, S. (2007). *Evocative objects: things we think with*, MIT Press, London.
- Turkle, S. (2011). *Falling for science: objects in mind*, MIT Press, London
- Wentworth, P., Elkner, J., Downey, A., Meyers, C. (2018), How to think like a computer scientist, <https://media.readthedocs.org/pdf/howtothink/latest/howtothink.pdf>
- Wilde, J. and Wilde, R. (1991). *Visual literacy*, Watson-Guptil, New York

Computational Thinking and Creativity

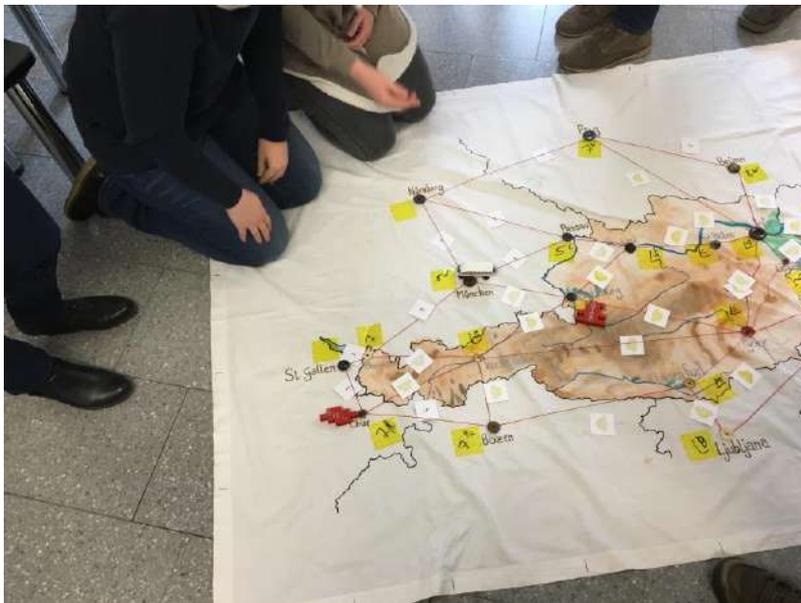
Gerald Futschek, gerald.futschek@tuwien.ac.at
Vienna University of Technology, Austria

Abstract

Creativity is one of the characteristics in constructionist learning. Creativity is frequently connected to learning by doing and needs a high amount of freedom in choice of activities and learning steps. Basically, Computational Thinking denotes thinking processes that are related to problem solving known from computer science. Computational Thinking does not only involve algorithmic thinking skills that are useful in programming and algorithm design but also integrates skills like abstraction, decomposition, generalization and evaluation that are used in problem definition, system modelling and system evaluation.

Since Computational Thinking has become part of informatics school curricula in many countries, the paradigm of competence orientation has led to very detailed curricula that describe a large variety of detailed competences. The high amount of details can lead to a kaleidoscopic teaching practice. Although, from a constructionist viewpoint it seems more promising to learn with learning settings that allow creativity, fun and sense of achievement. The role of creativity in Computational Thinking Learning is not only in creating a creative output but also in finding new ways of thinking to find solutions to problems.

We show several examples of learning settings and didactic projects that allow creativity, fun and sense of achievement in learning Computational Thinking.



*Students creating a new game
for Computational Thinking Learning*

Keywords

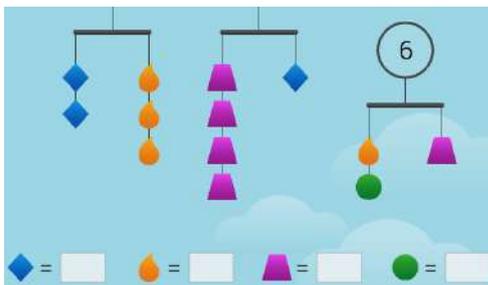
computational thinking education; problem solving; creativity

Teaching Children to be Problem Posers and Puzzle Creators in Mathematics¹

Paul Goldenberg, pgoldenberg@edc.org
 Education Development Center (EDC), Waltham, MA, USA

Abstract

Seymour Papert’s 1972 paper “Teaching Children to be Mathematicians Versus Teaching About Mathematics” started with the summary statement “The important difference between the work of a child in an elementary mathematics class and that of a mathematician is not in the subject matter...but in the fact that the mathematician is creatively engaged...” Along with “creative,” a key term Papert kept using is *project* rather than the common notion of *problem*. A project is not simply a very large problem. It centrally includes a focus on sustained and active engagement. The projects in his illustrations were essentially *research* projects, not just multi-step, fully-prescribed, build-a-thing tasks, no matter how nice the end product might be. A *mathematical* playground with enough attractive destinations in it draws children naturally to pose their own tasks and projects—as they universally do in their other personal and group playgrounds—and to learn to act and think like mathematicians. They even acquire conventionally taught content through that play. Physical construction was always available, and appealed to such thinkers as Dewey, but for Papert computer programming, newly available to school, suggested a more flexible medium and a model for an ideal playground.



Some cells are already filled. Fill in the rest.

The instructions you give.	Pictures	Suri
Think of a number.		
Add 3.		
Double that.		16
Subtract 4.		12
Divide by 2.		
Subtract your original number.		

I can read your mind! You got ____!!!

Figure 1. Two puzzles that introduce algebra’s logic and then its notation. Children can invent their own.

A fact about playgrounds is that children *choose* challenge. In working and playing with children I’ve seen that puzzles tap some of the same personally chosen challenge that a programming-centric playground offers. Children are naturally drawn to intellectual challenges of riddles and puzzles (ones they learn and ones they invent); and adults are so lured by puzzles that even supermarkets sell books of them. So how do real puzzles and school problems differ? What’s useful about *creating* a puzzle or posing a problem? How might puzzles and problem posing support mathematical learning? And what’s constructionist about this? This plenary will try to respond to these questions, invite some of your own responses, let you solve and create some puzzles, and explore how problem posing in programming and puzzling can support mathematics even in an age of rigid content constraints.

Keywords

Problem posing; puzzles; mathematics; algebra

¹ Funding for doing and reporting the work described in this paper was provided in part by the National Science Foundation, grants 1441075, 1543136 and 1741792. Views expressed here are those of the author and do not necessarily reflect the views of the Foundation

Teaching children to be problem posers and puzzle creators in mathematics

Papert's early work and the origin of constructionism largely were outside of the school setting. The current school environment is even more rigidly constrained than it used to be. The question is "Is there any hope for this kind of constructionist thinking and teaching *in* a school setting, not as a pull-out for well-resourced schools and with the best of their students, but as part of the regular program?" I want to share some ideas that, to me, exhibit the essential elements of constructionism and could easily be core to even moderately conservative school practice.

I, too, love playing with kids outside the classroom. There's more freedom and it's easier. But we all know that if we really want to touch many children's lives, we need to find a way to find them where they are. They are in school. I think it's possible.

Children choose challenge

Not all children; not all the time; but mostly. Children are often pretty adventurous on the playground. Tiny ones climb the monkey bars higher than their parents are totally happy with. When climbing gets too easy, they hang upside down. Children walk on five-inch-wide retaining walls two to three feet above sidewalk level when they get a chance; they hop across the street on one foot; when bicycle riding feels easy, they try letting go of the handlebars. Even with games, they up the ante if the game feels too easy, changing rules fluidly to add extra challenge.

For a toddler, there's enough challenge fitting the boat-shaped piece into the boat-shaped hole and the moon-shaped piece into the moon-shaped hole, but when that's no longer a challenge, kids seek more. Kindergarteners like fitting together the two-dozen jigsaw puzzle pieces of a large picture of a dinosaur. And when that gets too easy, some try putting the pieces together face down, some try jigsaw puzzles with smaller and more numerous pieces, and some just move on to totally different activities.

Children also put effort into figuring out how things work. Schulz & Bonawitz (2007) showed pre-schoolers a box with two levers and two different toys that popped up when the levers were pressed. One group of children were shown that each lever caused one toy to pop up. The other group saw only that when both levers were pressed simultaneously, both toys popped up. The first group's information was complete and unambiguous, with nothing left to figure out. The second group's information was incomplete: either lever might have controlled *both* toys, with the other doing the same, or nothing, or raising just one toy if pressed by itself. Or the two levers might be totally independent, one for each toy. When the children were then given the toy to play with (or ignore) on their own, children in that second group played longer, spontaneously exploring to puzzle out the cause and effect relationship. It's tempting to relate the first group's experience to the situation children often experience in school mathematics, where common pedagogy (at least in the U.S. and UK) shows exactly how each thing is done, leaving no evidence that there *is* anything to figure out, and taking little advantage of children's built-in curiosity.

Kids also love riddles, challenges to logic, interpretation or perception. And just as they spontaneously add challenge to their playground activities or jigsaw puzzles, they'll add to the repertoire of riddles by making up their own, sometimes creations that they, at their age, find funny (illogical) because the challenge "works" for them, and that we adults find simply ludicrous (illogical) because the challenge no longer works.

The point is the challenge. When it's not there, children are bored. When they're bored, they *invent* challenge. In school, that inventiveness can be to the dismay of their teachers, whose response may dismay the children, but that won't stop the drive for the challenge.

Why puzzles?

Kittens stalk and pounce to make their hunting skills sharp and they scratch to keep their claws sharp; that's because sharp claws and hunting skills are among the particular adaptations that make their species successful. *Our* species' special adaptation is not sharp claws and pouncing but a mind that lets us adapt to nearly any environment, which is how we wound up populating city and farm, blazing

heat and frigid cold, arid and tropical jungle. Keeping our *minds* sharp is what makes our species successful.

That has implications for learning. In *our* species, it is adaptive for the young to be a bit distractible and *not* to focus too narrowly. Because we live in such varied environments we cannot have “built in knowledge” about which features will matter most for survival. As children, we watch social behaviour (whom should we copy, whom should we stay close to, whom should we stay away from?), animals (food or danger?), artefacts (how do they work?), math lessons (who knows?, maybe they’re important) and everything else.

Some distractibility and lack of focus are assets to a child, but less so for adults who must earn their living, whether by blow-darting the rabbit (while avoiding the tiger) or by generating research papers or by teaching children. But adults still have to keep their minds sharp. Adults argue the merits of ideas—politics, religion, co-workers—even when the *practical* value of the argument is near zero. It’s a mental exercise. Puzzle books for adults are sold not just in academic bookstores but also in supermarkets; puzzles appear in newspapers and in airplane magazines. Boredom is painful; enforced boredom is torture.

Puzzles and surprise in mathematics learning

In 1964, Sawyer (2003) seeded the ideas for a wonderful textbook series for primary school mathematics (Wirtz, et al., 1964) and for our own curriculum materials (see, e.g., Goldenberg, et al., 2008). He took a very algebraic approach to teaching elementary arithmetic, with a major emphasis on play and surprise. On the surface, the content was exactly what one expects for the grade level but with a twist that included research, puzzles for children to figure out, all foreshadowing the algebra that children would learn later.

For example, as a way to give seven-year olds practice with addition and subtraction they start

with a piece of mathematical research. A child is asked to suggest some addition equation like $4 + 1 = 5$ or $1 + 1 = 2$, and the teacher would write it at the board. Another child is asked to suggest a new equation, which the teacher carefully lines up directly underneath the first. Then the teacher has the children add vertically, displaying the results like this.

$$\begin{array}{r} 4 + 2 = 6 \\ 1 + 2 = 3 \\ \hline 5 \quad 4 \quad 9 \end{array}$$

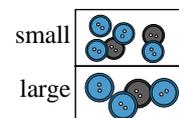
Do these three new numbers make a true addition equation? The teacher completes the bottom row of numbers to read $5 + 4 \stackrel{?}{=} 9$. Surprise! Will this *always* happen, or did they just get lucky? Children are given the challenge of finding a pair of addition sentences that *don't* work. Seven-year-olds are sure they can find some and set off busily, getting lots of practice.

$$\begin{array}{r} \square + \square = \square \\ \square + \square = \square \\ \hline \square + \square \stackrel{?}{=} \square \end{array}$$

Of course, they *will* find some (they think) and report them excitedly, but the preponderance of cases that do work will get even seven-year-olds to doubt the counterexamples and check to see if they’ve made a mistake. This research is hardly a project in Papert’s sense. The problem may not even last more than one classroom period or so. But it *does* generate curiosity, the creative engagement that Papert referred to as the experience of the mathematician.

Their research convinces them of a result, but if we don’t leave it as magic and instead help expose the logic inside the puzzle, children get even more excited. They have a tool they can and do use, first to figure out for themselves why the puzzle works and then to invent new

puzzles for themselves and their friends! Exposing the logic involves reminding children of reasoning they developed in Kindergarten and first grade. Given a collection of buttons differing by two attributes, colour and size, kindergarten children naturally sort, though sometimes their sorting is idiosyncratic—two large buttons and a small one, for example, to make a “family.” They learn to respond to “show me a small button” and “how many small buttons do you have?” And they can learn to respond to “show me a



large grey button” and “how many small blue buttons do you have?” After sorting by a single attribute, they can learn to sort by two attributes.

	blue	gray	
small			
large			

Now, when we ask how many small buttons and how many large, we are summarizing the rows, and we can write that summary.

small			6
large			4

We can similarly summarize the number of buttons by colour (columns).

	blue	gray
small		
large		
	7	3

Once children really have and can use cardinality, it is clear that the number of blue and grey *must* be the same as the number of large and small—either way, it’s all the buttons. Second grade students comfortably replace buttons with numbers and then use that structure as part of their reasoning.

	blue	gray	
small			6
large			4
	7	3	10

	blue	gray	
small	4	2	6
large	3	1	4
	7	3	10

Figure 2a, 2b. Actual buttons replaced by the number of buttons.

Reading across, children see $4 + 2 = 6$ and $3 + 1 = 4$; adding down the columns, they get 7, 3, and 10, which *must* make a true addition statement.

Subtracting down isn’t always possible for 7 year olds—depending on the situation, it might require negative numbers, and the meaning changes, too (it doesn’t yet make sense to subtract the number of large buttons from the number of small ones)—but with numbers that they *can* subtract (as is the case in Fig. 2b), the arithmetic still works and produces a true addition statement. Subtracting to see how many more small buttons than large, we get $1 = 2 - 1$, and that exact same logic will be essential in algebra a few years later!

Subtracting

$$\begin{array}{r} 5x + 3y = 23 \\ 2x + 3y = 11 \\ \hline 3x + 0 = 9 \end{array}$$

The format isn’t just a trick, or a school artefact; it’s the structure of any spreadsheet that subtotals the columns and rows and has a grand total. Wirtz, et al. (1964) used this format as a puzzle (right), and as a route into multi-digit addition and subtraction (fig. 3).

6		11
	3	
		18

Which cell might you fill in first?

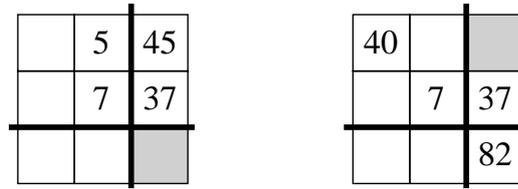


Figure 3. The same arithmetic presented in 3a as an addition puzzle $45 + 37$, with the grey square as the sum, and in 3b as a subtraction puzzle $82 - 37$, with the grey square as the difference.

For 9- or 10-year olds, this structure also models the multiplication algorithm. Instead of colour and size labels, we label the width and height of the columns and rows, and imagine cells filled with unit squares instead of buttons. How many squares are in the four regions? In the most concrete image, everything is to scale (fig. 4a).

With a smaller array, say 3×4 , we can see why multiplication gives the answer and we can count to check. But with numbers like 37×26 , we certainly don't want to count! Instead, we use an abstraction (fig. 4b), ignoring scale, but maintaining a sense of the *logic* of multi-digit multiplication, not a set of memorized steps that often wind up feeling arbitrary. Of course, the steps involved in this logical model map perfectly onto the abbreviated notation often taught in school, and fully explain that notation. In fact, it is worth delaying the abbreviated notation until children are so secure in the logic of the array model that they can easily extend it to three-digit multiplication, because exactly this method—four separate multiplications and only *then* a possible summing up—will be required when the students study algebra.

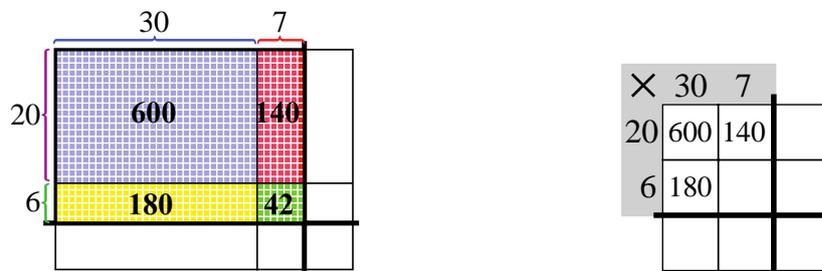


Figure 4. An array model of multiplication true to scale (left, figure 4a) and abstracted (right, figure 4b).

Sawyer suggested other ways that even very elementary content like addition and subtraction of small numbers could be learned or practiced in a puzzle-like context that both builds curiosity and foreshadows later ideas and methods. Figure 5, for example, shows what a standard worksheet might present as 16 unrelated addition/subtraction practice exercises for 7-year olds, but structured in a way adds a bit of intellectual challenge—how-do-I-do-this?—and foreshadows systems of equations that the children will meet several years later.

$\square + \triangle$	7	10			29				20	26
\square	4	8	9	10	20		16			
\triangle	3	2	4	5		7		20		
$\square - \triangle$!	6				0	8	40	6	8

Figure 5. A practice exercise for 2nd grade, foreshadowing systems of equations. (Wirtz, et al., 1964)

Again, it's not a "project" in Papert's sense, and not "creative" in the most familiarly used sense of that word, but especially the last two columns pull for children to be *mathematically* creative.

One of the most powerful introductions to algebra that I've seen is Think-of-a-number tricks, also from Wirtz, et al. (1964): Think of a number. (Yes, you! Please think of a number.) Add 3. Double the result. Subtract 4. Cut that result in half. Subtract your original number. Aha! I can read your mind! You got 1 at the end!!!

For 9- or 10-year-olds, this is wonderful magic. They want to do it over and over, but also want to know how it works. I say that I picture the secret number as that many marbles or whatever, tucked in a bag  or bucket  where we can't see them—only the secret keeper knows the number inside. When I give the instruction “add 3,” I *know* about *those* grapes, so I draw them outside the bag. I ask the children what the next instruction is (they almost always remember) and what the picture should be like (they almost always say “two bags and six marbles”). Then I continue, each time asking the children to describe the next picture. At the end, “subtract your original number” gets rid of the bag. So the number of grapes in it doesn't matter! There's one grape left, and we can see it!

Even after the usual huge smile and the cry “I get it!”, seeing it once isn't enough. The understanding evaporates until children see the generality, not just the way this particular trick worked. To create that abstraction for themselves, children need research time: practice drawing pictures to match instructions, applying instructions to specific numbers, and variations on the trick from which to generalize and learn to invent their own tricks.

They also need chances to study the trick inside out and backwards, starting, for example, with the 16 that Suri had in mind after the instruction “double that” and figuring out what secret number she must have started with. To do that, a child might note that the picture corresponding to Suri's 16 *shows* six grapes, so ten grapes must be hidden in the two bags. Suri's secret number—the grapes in *one* bag—must have been 5.

Some cells are already filled. Fill in the rest.

The instructions you give.	Pictures	Orli	Naomi	Suri	Adam
Think of a number.	 <i>Imagine your number is hidden in the bag.</i>	4			0
Add 3.		7	10		
Double that.				16	
Subtract 4.				12	
Divide by 2.					
Subtract your original number.					

I can read your mind! You got ____!!!

Figure 6. Using bags and grapes to introduce 3rd graders to algebraic notation and solving equations. (Problem from Wirtz, et al., 1964, reworked for 3rd grade based on Mark, et al., 2014.)

I've recently been introducing a new crop of 8- and 9-year olds to algebra this way and told them that they'd soon know how to invent new tricks of their own. After two days of playing with the puzzle, Lucy said “I really get it, but I still don't know how to make up my own.” So we played. I said “OK, I've thought of a number” and I drew . “Just make up one instruction, anything you like, and I'll draw the next picture.” She said “add 5?” I said “OK,” drew , and asked “What next?” She said “double that?,” still with the question in her voice. I said “whatever you'd like me to do... Is that what you want

me to do?" She nodded and I said "you draw the picture." She drew two buckets and 10 dots. She then told me to subtract 2 (no question in her voice, and she drew the picture), then subtract 7 (she drew the picture). That change in tone—no question in her voice—was because she now understood something new, not about *the* mathematics of this trick but about mathematics, itself. She could just make up a rule, *any* rule, and it was then up to her to figure out its implications. That is so like watching a child program, see the effect, decide whether that effect is desired or not, and then decide what to do next.

I asked, "OK, what can you do in order to know my number?" Long pause. Then Lucy commanded "subtract your original number" (and drew the picture). After another pause, she said "Oh!! Subtract your original number *again!*" Her smug smile showed clearly that she knew what she had done but I wanted to check, so I prompted her to "read my mind." Instantly, but with excitement and what also sounded like surprise in her voice, she said "Oh! One! You got one!" as if understanding the trick for the first time all over again. The joy of "getting it" is far more magical than any grade, praise or prize could be.

These are five to fifteen minute events. By the end of a week of them, instead of *drawing* the pictures that the children describe, we write the *words* with which they describe the pictures. "Two bags and six marbles" is a lot to write, so we abbreviate it: $2b + 6$. No discussion of variables; no explaining about letters standing for numbers; $2b + 6$ is brief, but the language the children themselves used, and they fully understand it. For now, that's enough. Later, when they formalize algebra, the bag or bucket image is useful to return to: a variable is a container for a value.

Containing a value (or being a *pointer* to it) is the programmer's image; *representing* a value is the mathematician's image. The underlying idea common to both images is that a value can be referred to by a *name* and that this abstraction is useful. In practice, nearly all children love the think-of-a-number tricks, so they become a natural, appealing and compelling way to acquire that value-naming idea. Part of the power of the "trick" is that it is faithful to the mathematics, even though it is limited.² But part of its power, I'm sure, is what Schulz & Bonawitz (2007) saw: children play longer and more curiously when there's something they don't understand *and they believe that they can figure it out.*³

This was not a classroom assignment. The children didn't have to do this and wouldn't be tested on it. But they put effort and attention into the think-of-a-number trick because they *want* to know how it works. The intensity of Lucy's interest, even readily admitting what she couldn't yet do and asking for help doing it, was because there was a genuine mystery left to solve—one that she saw as *hard*—but she was so tantalizingly close that she was convinced she could reach that goal.

Why have students *invent* puzzles?

Four reasons come immediately to mind; perhaps there are more.

First, the *construction* of a workable puzzle is a creative act, making the student a creator and not just a consumer of mathematics. We who call ourselves constructionists easily accept making as a good thing, but it's useful to say why. What you *make* is yours; creating gives ownership. Mathematics is often perceived—except by mathematicians—as the antithesis of creativity, a subject in which rules rule and we obey. It's very possible to learn mathematical *content* that way, and some people like that order and simplicity. But mathematical *thinking* can't work that way because genuinely new problems could then never be solved. For new problems, one *must* create new ideas and approaches. Young students' mathematical creativity can't be at the leading edge of mathematics, but it can be at *their* leading edge. Puzzles are not the only opportunities for students to be creative in mathematics but they're good ones, especially for younger students.

² This imagery doesn't represent "divide by 2" well unless the numbers of bags and marbles are both even. The imagery is adaptable to "negative grapes," but frankly awkward. So we need to be clear that the imagery is not the goal, not a "new method" for algebra. But it's an extremely effective entry to algebra.

³ This qualification is important. *Nobody*—no corporation, no person—puts time/money/effort into an endeavor that they believe has no chance of success. Students who have been convinced they are "no good at math" often don't put effort into study that we believe would make them better. But *they* don't share that belief, so from *their* perspective, it is wiser to aim their efforts in a direction that seems more likely to pay off. That is an adaptive, economical choice. That is why it is so important to show (not tell) them that they *are* capable by hooking their interest on something they perceive as hard but attainable.

Second, constructing a good sharable puzzle is a balancing act—easy enough to be solvable and hard enough to be fun. To be solvable, a puzzle must also be well specified—enough clues to derive a unique solution (or a limited class of solutions)—without having so many clues that only the arithmetic is left. Determining when one has given enough clues to derive a solution is quite a challenge.⁴

That challenge, and also the act of being a creator, may be part of *why* construction of a sharable puzzle appeals to kids, but the appeal is yet another reason to have kids create.

And fourth, construction of a sharable object helps reveal the child’s thinking to both the child and teacher, supporting refinement of that thinking, and discussion and analysis.

SolveMe.edc.org is a puzzle world with three kinds of puzzles aimed at developing algebraic reasoning. Each puzzle type also lets students create their own puzzles and share them on line. The Mobiles app collection begins with relatively elementary puzzles like the ones in figure 7, and even simpler ones for real beginners.

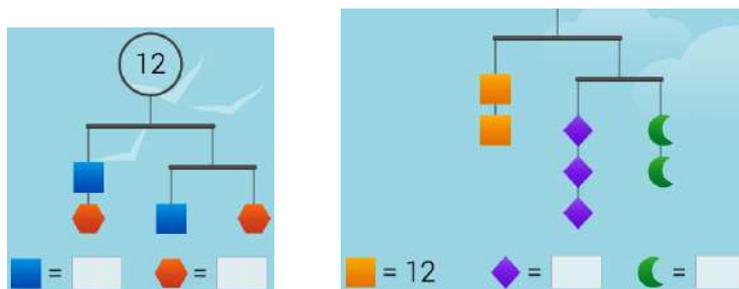
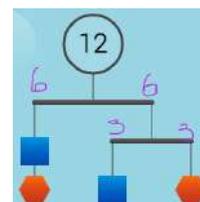


Figure 7. Two relatively simple mobile puzzles.

The mobile’s total weight might be given (above, left) and players must figure out how much the blue red objects must weigh in order for this mobile to balance. Or (above, right), no total weight might be given, but the weight of one of the hanging objects might be specified. Again, the player must puzzle out the weights of the other objects.

Players often just work these out in their heads, but the app offers them other options: they can scrawl annotations on the screen, like this. →

They can also create equations by dragging off a copy of a horizontal beam $\square = \circ$, or the entire mobile $12 = 2\circ + 2\square$, and substitute these into other equations (or the mobile) to derive new information, like $12 = 4\circ$. The app also lets them factor 2 out of equations like $12 = 2\circ + 2\square$ to derive new



equations $6 = \circ + \square$ and to drag a common element out of both sides of an equation like $4\circ = 2\square + \circ$ —to see $4\circ = 2\square + \circ$ —and to get $3\circ = 2\square$. Otten, et al. (2017) describe

how 11-year-olds used explicitly algebraic correct reasoning in the context of informal notation and manipulations of a physical hanging mobile.

The mobile puzzles are essentially systems of equations. Some students are intrigued by the fact that they can get those equations and see what those equations mean. In class, that is an advantage, but informally, even the students who like the fact that they can get equations mostly don’t work with the equations, instead inventing informal methods equivalent to the formal manipulations that algebra classes teach and name. They also see, early on, that the “weights” can be fractional and even negative.

Some of the puzzles are quite challenging, like the ones in figure 8a, without being required to, students really persevere because they’re sure they can solve the puzzles if they keep at it.

⁴ This is especially true in creating a good MysteryGrid puzzle or Who Am I puzzle, not described here, but part of the SolveMe suite of puzzles mentioned below.

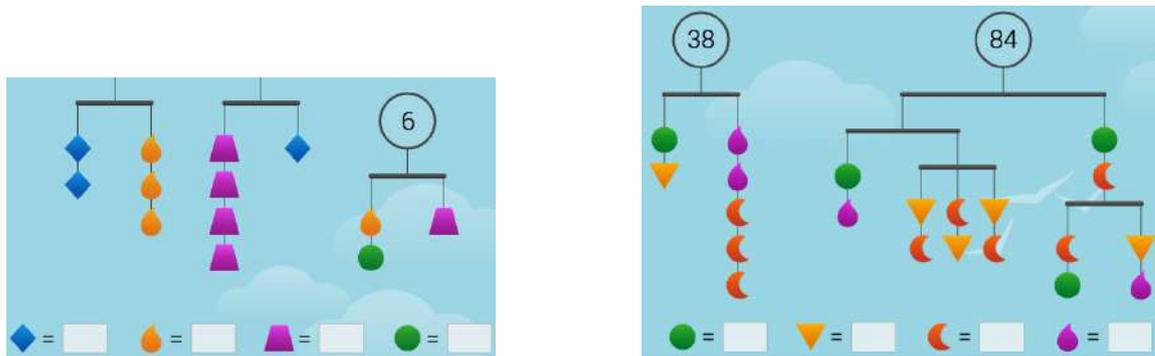


Figure 8a. Two mobile puzzles at a more advanced level.

As I'd said, we felt it important to provide a tool with which students could create their own puzzles and even share them with friends or with the entire SolveMe community. The sheer variety of users' contributions is fascinating. Some are genuine puzzles, like those shown above. Others seem to be intended more as works of art, like these.

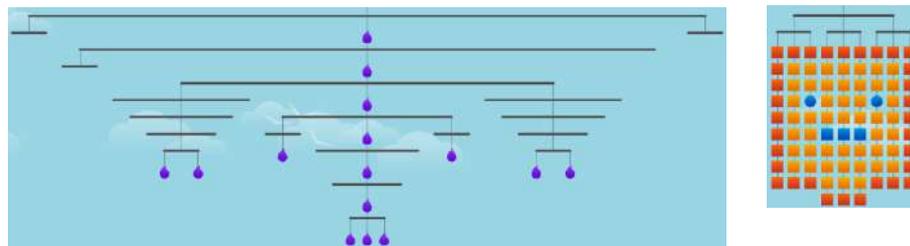


Figure 8b. Two mobile "puzzles" invented by users, apparently intended only as art.

Manousaridis (2018) regularly encourages students in grades 2 and 3 to create their own puzzles as posters after solving some on line. Part of her goal is, of course, the ownership that comes from building a puzzle. But it is also clear that the task naturally leads children to work at the frontier of their ability, partly because they take special pride in pushing (and displaying) what they can do.

The 9-year-old who created the puzzles shown in figure 9 was clearly proud of the arithmetic she did but especially proud of having created a puzzle that *required* such fancy arithmetic. The puzzle, not just the artwork on the poster, is a highly personal and creative act. This child is what Papert (1972) described as "the mathematician... creatively engaged."

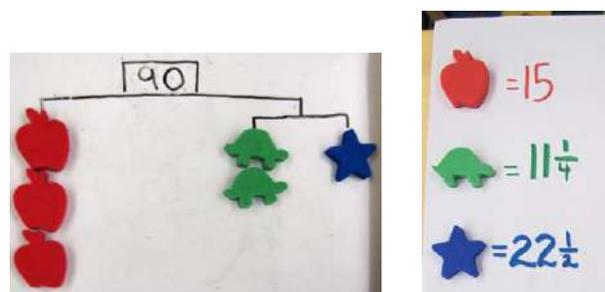


Figure 9. A mobile puzzle invented by a nine-year-old to challenge her classmates to use fractions.

Programming in mathematics

The examples and contexts described above have been very far from the programming-centric proposal that Papert made in (1972) but, as I've tried to show, well in line with the mathematical creativity, exploration, and research projects that he regarded as *doing* mathematics rather than learning about it—creating and solving one's own problems versus learning mathematical facts and solving problems

created by others. While nobody would claim that programming is the only (or even always best) venue for creative expression and exploration in mathematics, I and others believe it can be an *enormous* help if it can become a natural part of learning mathematics. For it to be “a natural part,” it would need to be learned along with the mathematics, growing over time just as the mathematics does, and used in ways that *support* the mathematics and don’t compete with it by seeming to be a separate venture—fun stuff but disconnected—or by creating excessive overhead or distraction. If that can be achieved, then the flexibility and expressive ability of programming can give it a central role in children’s mathematical learning and creativity.

Noss & Hoyles (2018) focused especially on that expressive ability:

Maths is difficult in part because of the language in which it is expressed. Can we find a different language—and set of ideas and approaches—that is more open, more accessible and more learnable. And can we find it without sacrificing what makes mathematics work? Our tentative answer is “yes”—the language of programming might, if we design it right, be just such a language.

Mathematics really needs three languages. Two are already used universally in school: natural language for semantics (context, explanation, and some of the logic) and conventional arithmetic/algebraic notation. Unfortunately, if used inappropriately, both can also get in the way. In particular, mathematical notation is too often used as the entry point to new ideas rather than a concise way to record ideas that are already well understood. Consider that the third graders knew intuitively that doubling  ●●●●● produced  ●●●●●, and six-year-olds, when asked *verbally* (not in writing) what five eighths plus five eighths might be, are happy to respond “ten ayfs,” then ask what an ayf is. They never answer ten sixteenths. The distributive property is *built in* to our logic early. But when the word “distributive property” is introduced in third grade, it is often taught with a written string like $8 \times 7 = 8 \times (5 + 2) = (8 \times 5) + (8 \times 2) = 40 + 16 = 56$ that is opaque and daunting to a beginner. Despite your mathematical literacy and knowledge, probably even you zipped past the string of symbols without reading closely enough to see if it was correctly typed. Processing such a string of symbols takes focus and effort, and therefore can’t be the optimal way to *introduce* the distributive property to an eight-year-old. Too much cognitive space is taken up just decoding the long string; not enough is left for thinking about the idea.

And neither natural language nor mathematical notation is particularly good at expressing process or algorithm. That’s what a good programming language can provide. Also, unlike a string of symbols or words that sits on paper—correct or incorrect—and gives no feedback without the reader (re)reading and (re)processing it mentally, a programming language is a notation that can be *run* and gives direct feedback on what it does.

ScratchMaths (Noss & Hoyles, 2018) is one beautiful example of infusing programming directly into grade-level-required mathematics for 9- to 11-year olds. At EDC, we are building a full-year mathematics+programming curriculum for 7- to 8-year olds as a stepping stone to doing the same for all of the first six grades of school, using the playful and puzzle-centric ideas of Sawyer (2003) and Wirtz, et al. (1964) as redeveloped in our *Think Math* curriculum (Goldenberg & Shteingold, 2007a and 2007b). First programming experiences for young children can be quite open—moving a robot around, or just code-streams of interesting effects—but if we are explicitly intending to show 7-year-olds how they can use programming as a language to experiment with and express the mathematics they are currently learning, the first coding experiences need to be rather simple while, at the same time, leaving plenty of room for puzzling and exploring.

Here is one environment EDC is currently exploring with children. The computer displays a number line, optionally settable for any range, with only one number labeled. The ticks just mark regular intervals, but interval size is completely settable (consecutive integers, consecutive eighths, skip counting by any amount, starting at any arbitrary number). For the youngest children, the ticks indicate consecutive integers, with only 0 labeled (and intentionally chosen not to be the leftmost mark on the line). The 7-year-olds are given a palette of tools from which they can draw. For example, they may have  ,  ,  and, later, a **repeat** block.

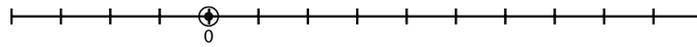


Figure 10. A simple number line with ticks representing consecutive integers.

Clicking a tool performs the indicated arithmetic, shows the corresponding movement on the line, and labels the result. For example, if the sprite is at 3, clicking the **+5** block moves the sprite 5 spaces right (arc optional) and marks 8 there (figure 11).

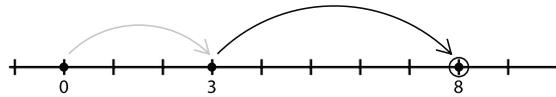


Figure 11. A move of **+5** from 3 to 8.

Children explore the tools with very open puzzles like “Try to label all the numbers 0 to 10.” When they have learned how blocks can be snapped together to create a script, they get more focused puzzles of increasing challenge that require experimenting, planning, mental arithmetic, and prediction of results. Two puzzles are shown in figure 12, as they might appear to children.

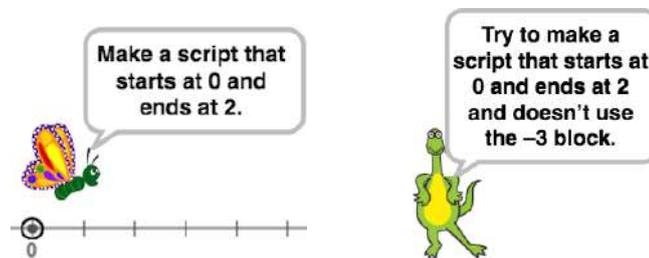
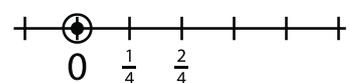


Figure 12. Two programming puzzles.

More challenging problems can follow: Create two different scripts that... Children can also be asked in class discussion to analyze and explain. For example “explain why this script moves **+3** from 0 to 1. If you start at 4, where will this script move you? Is there a way to move from 0 to **+3** 1 in exactly 2 moves? What’s the *shortest* script that...?”

Because activities like this “have legs” mathematically, they can grow with the child and be used in later grades. At the simplest, the very same activity can be used on a “zoomed in” view of the number line, to explore fractions. The children’s tools now include **+ 3/4**, **- 5/4**, etc.; similar tasks would



challenge them to mark $\frac{1}{4}$, $\frac{2}{4}$, etc. Still other variants—like changing the tools to ± 6 and ± 9 and challenging children to label all the numbers they can—show how versatile this format is, capable of addressing later grade-level standards (e.g., factors, multiples, common factors, analyzing patterns, building fluency with multiplication facts) and foreshadowing in grade-appropriate ways ideas children will make explicit later. Other elementary school projects involve functions (e.g., **4 × 10 + 2** ²⁰)

and let students build and compose their own. And creating code that generates squares from repeated moves, rows from repeated squares, and arrays from repeated rows—and the similarities of the algorithms inside **draw row** and **draw array**—illustrates the meaning *and value* of “abstraction.” Abstraction includes both generality, and “hiding complexity”—suppressing details or identifying the important characteristics for a particular purpose—by creating a single *new* command/function to replace a longer collection of instructions that would otherwise have to appear in several places. This example also shows how the activity directly supports *mathematical* content mandated for schools. The **draw array** block depends on two parameters, the dimensions of the array. A **draw rectangle** block is a further (though simpler) abstraction, using the same two inputs but drawing only the border of the array. For either of these, children might invent a playful quiz, having their program draw a random-

sized array and ask about area (how many tiles it has) or perimeter, using their own reasoning about those inputs in order to teach the program how to calculate the correct answer.

And, at the high school level, programming allows students to *build* the mathematical objects and processes that they are studying: relatively easily, they can *build* functions that manipulate polynomials, transform points with matrices, render a set of points in space in a convincing projection on the screen (Lewis, 1990), and study algebraic structures (Cuoco, 1990). And tools such as Geometer's Sketchpad or Cabri—not programming in the usual sense, but construction with the computer rather than just use of the computer to manipulate pre-designed models—allow students studying geometry to build models of mathematical objects and ideas, and to explore the consequences of manipulations of those models.

Programming in general

Secondary students, too, need places to be mathematically creative. The mission of Beauty and Joy of Computing (BJC, 2017) is broadening participation in computer science with a focus on letting students see not just the joy of creation and beauty in the objects they can produce through programming, but also beauty in the programs themselves. With this as one goal, it introduces the elegance of recursion and higher order functions. It manages to make these reputedly “difficult” topics accessible by virtue of the lucid visual imagery of the Snap! visual

programming language, a block-based language that is reasonably characterized as essentially Scheme disguised as Scratch. Two BJC excerpts involving recursion were used in a computer science elective with sixth graders. They wrote recursive code to draw a complex tree, and here they and their teacher are giggling at the result of a gossip-producing program with a randomly invoked recursive step that, in this case, generated quite a long sentence. Other students in this elective created a program to conjugate Spanish verbs properly so that they could generate sentences in Spanish.



They tested the work of their programs by using `map`, a higher order function, to apply their conjugation block to a list of verbs.

Initial funding for BJC required it to be an Advanced Placement course with a framework dictated by the College Board. Even so, except as constrained by AP requirements, BJC is largely project based with experience before formality; the explorations through which programming is learned include projects set in contexts like art and graphics, linguistics, mathematics, and games. While BJC is not a math course, its activities naturally touch—and help teach—many conventional mathematical content topics, and our approach to programming is consistently focused on mathematical/computational thinking.

The point of introducing various contexts—the arts, linguistics, etc.—is partly to meet the varied interests of students, but much more to show how broadly *they* can allow themselves to wander, how much they can tailor their independent projects, for which even the AP framework allocates time, in their own personal direction.

Playgrounds

Giving even very young students a way to think algebraically using bags and grapes lets them invent mathematical tricks they love. It prepares them for algebra but more importantly, it lets them feel smart and pose problems and *play* with their own algebraic ideas. More broadly, treating mathematics as serious intellectual play, puzzling things out by searching and researching, and gaining the intellectual tools for posing one's own challenges teaches children to be mathematicians. Papert suggested programming as a medium for that, but the essential ingredient remains the promotion of serious intellectual play. Programming taught just as a skill or to meet new standards may well not serve that purpose. But if a programming environment lets students explore and create, provides good tools for

doing that, and gives students the “third language of mathematics” so that as their ideas and thinking grow in sophistication they have a language for expressing and honing those ideas, such an environment does add a new playground consistent with Papert’s vision of children being creatively engaged as mathematical thinkers.

A question, based on wild final thoughts, pure opinion that I might disown tomorrow

A few states, including Massachusetts (the one that I live in) have begun to develop frameworks for computational thinking (CT) across the grades (DESE, 2016). CT is variously defined but always includes elements like abstraction, algorithm, modelling and simulation, programming, and data (with an implication, not reflected in all implementations, that “data” means *big* data). Not surprisingly, there has also been a proliferation of on-computer and “unplugged” activities, *not* involving programming, to help develop this thinking.⁵ The difficulty of adding anything else to an already jam-packed school day has led to a lot of talk about *integrating* CT activities into the existing content areas, particularly science and mathematics (e.g., EDC STEM+CT, 2018), but also language. In my opinion, some of the suggestions are shallow, but that should be no surprise at a time when the whole effort is so new.⁶ Still it got me to thinking about why my own inclination has been *toward* programming, not away, and toward abstraction, and algorithm rather than modelling and simulation, whenever the aim is explicitly to integrate with other subjects.

I think it’s largely bias. I tend to think more about elementary and middle school, and more about *mathematics* than about science. At the elementary school level, modelling and simulation are easier to integrate with science than with mathematics; *programming*, along with abstraction and algorithm, is easier to integrate with mathematics than with science.

Modelling, for example, is something that mathematics (and mathematicians) can *do*, and since mathematics can build models of mathematical ideas, modelling is also something that mathematics *uses*. But, at least as far as I see at the elementary school level (especially in the early grades) modelling *with* mathematics—creating mathematical models of phenomena—is very limited. And fairly abstruse, in the following sense. While *every* mathematical statement (like “there are seven cows”) is an example of an abstraction (the cowness is reduced to irrelevancy) and just a model of the reality, no kid in the known universe thinks of such a statement as an abstraction or a model. That level of abstraction is so normal to them that it is totally “invisible”—it’s just what language *does*. By contrast, modelling is a *natural* place to focus in science—the core of experimentation and the form of many scientific claims—and simulation (at least as generally used) is an automation/extension/elaboration of modelling.

Programming is exactly the opposite, easier to integrate into (early) mathematics than into science. (Of course, take this with a cup of salt, as I’ve not given scientific programming nearly as much thought. As I advertised, these are wild final thoughts that I might disown tomorrow.) That may be partly because the kinds of statements one makes in early mathematics tend to be about relationships and about simple processes. “Writing a program” that enacts a function, like doubling its input or adding 10 to its input, is easy programming. In fact, it’s easier to write in a general way as a *program* (a Snap! block) than as a paper-pencil scrawl, because a program is an *active* notation; it will *enact* the action and give feedback, which paper-pencil scrawl do not. It is also a structured notation, imposing a bit of order on what young students typically scatter over a page in a way that, even if totally correct, does not reveal their logic. Similarly, writing a program that pairs elements of two sets, writing a program that draws simple shapes, or creates arrays or paths to study, is mathematically on task and easy programming. By contrast, most scientific phenomena are too complex for young children to model by writing a program (often pretty complex even for adults).

⁵ Just as I was completing this paper, I received a copy of *Bebras* (Dagienė, et al., nd), a set of activities, many puzzle-like, that I found quite appealing, all designed to develop various elements of computational thinking in students.

⁶ And, clearly I, myself, am being a bit shallow in using the vague quantifier “some suggestions.” Of course, in *any* situation, *some* suggestions will be shallow.

I'd love to get reaction to this last, very spur-of-the-moment rumination. What genuine *programming* activities, *at the elementary school level*, can be really well integrated with *science*? And what *modelling* or *simulation* activities, again at the elementary school level, can be really well integrated with mathematics?

Acknowledgments

Funding for doing and reporting the work described in this paper was provided in part by the National Science Foundation, grants 1441075, 1543136 and 1741792. Views expressed here are those of the author and do not necessarily reflect the views of the Foundation.



References

- BJC. (2017) The Beauty and Joy of Computing. <http://bjc.edc.org>
- Cuoco, A. (1990) *Investigations in Algebra*. Cambridge, MA: MIT Press.
- Dagienė, V., Stupurinė, G., Vinikienė, L., and V. Kincius. *Bebras*. Creative Commons Attribution-ShareAlike 3.0 Unported License (CC BY-SA 3.0). <http://www.bebbras.lt>
- DESE. (2016) Massachusetts Department of Elementary and Secondary Education. *2016 Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework*. Accessed at <http://www.doe.mass.edu/frameworks/dlcs.pdf>, 16 September 2016.
- EDC STEM+CT. (2018) <http://go.edc.org/elementary-ct> The Broadening Participation of Elementary Students and Teachers in Computer Science project is directed by Joyce Malyn-Smith. Accessed March 29, 2018.
- Goldenberg, E.P., Mark, J., Kang, J., Fries, M., Carter, C. and Corder, T. (2015) *Making Sense of Algebra: Developing Students' Mathematical Habits of Mind*. Heinemann: Portsmouth, NH, USA.
- Goldenberg, E.P. and Shteingold, N. (2007a) Early Algebra: The MW Perspective. In *Algebra in the Early Grades*, Kaput, J.J., Carragher, D.W. and Blanton, M.L., Eds. Erlbaum: Hillsdale, NJ, USA.
- Goldenberg, E.P. and Shteingold, N. (2007b) The case of *Think Math!*. In *Perspectives on the Design and Development of School Mathematics Curricula*, Hirsch, C., Ed. NCTM: Reston, VA, USA.
- Lewis, P. (1990) *Approaching Precalculus Mathematics Discretely*. Cambridge, MA: MIT Press.
- Manousaridis, T. (2018), personal communication.
- Mark, J., Goldenberg, E.P., Kang, J., Fries, M. and Corder, T. (2014) *Transition to Algebra*. Heinemann: Portsmouth, NH, USA.
- Noss, R. and Hoyles, C. (2018) The ScratchMaths project is directed by Richard Noss and Celia Hoyles, with Ivan Kalaš, Laura Benton, Alison Clark Wilson, & Piers Saunders. See <http://www.ucl.ac.uk/ioe/research/projects/scratchmaths>. Accessed March 29, 2018.
- Otten, M., van den Heuvel-Panhuizen, M, Veldhuis, M., Heinze, A. and Goldenberg, P. (2017) Eliciting algebraic reasoning with hanging mobiles. *Australian Primary Mathematics Classroom*, Vol. 22, No. 3, pp. 14–19.
- Papert, S. (1972) Teaching Children to be Mathematicians Versus Teaching About Mathematics, *Int. J. Math Ed in Science and Tech.*, Vol. 3, No. 3, pp. 249–262.
- Sawyer, W.W. (2003) *Vision in Elementary Mathematics*. Dover: New York, NY, USA.
- Schulz, L.E. and Bonawitz, E. B. (2007) Serious Fun: Preschoolers engage in more exploratory play when evidence is confounded. *Developmental Psychology*, Vol. 43, No. 4, pp. 1045–1050.
- Wirtz, R., Botel, M., Beberman, M. and Sawyer, W.W. (1964) *Math Workshop*. Encyclopaedia Britannica Press: Chicago, IL, USA.

May I Teach an Algorithm?

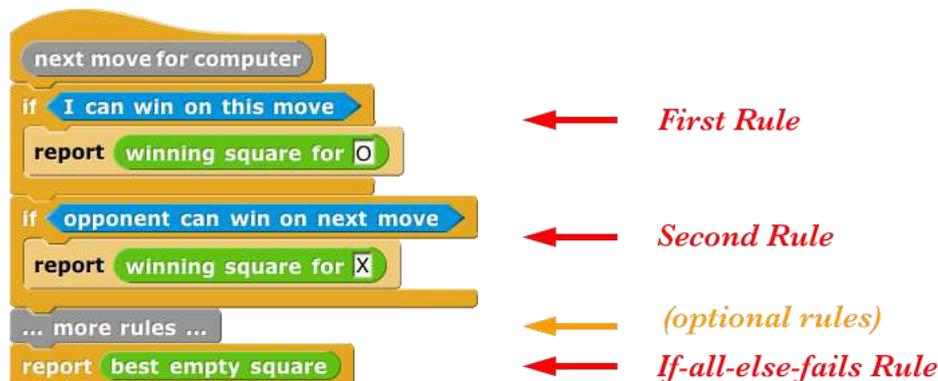
Brain Harvey, bh@berkeley.edu
University of California, Berkeley, USA

May I Teach an Algorithm?

BRIAN HARVEY, bh@berkeley.edu
University of California, Berkeley

Abstract

After all these years, I'm still not sure what Constructionism entails. Here's an example taken from our work on the *Beauty and Joy of Computing* curriculum: We have a yearlong Tic-Tac-Toe project. Early in the year, students draw the board and use mouse clicks on the board to let two human players alternate moves. A month later, they start building data structures that will let the program analyze the board. At that time, they check whether either player has won, or whether the game is tied. Two months after that, they return to the project, letting the program be one of the players, and determining the program's move with rules such as "if I can win on this move, do it." A Tic-Tac-Toe program is an obvious project, one of the things many learners do spontaneously. But I had an ulterior motive in the design of the project: I want students to practice using higher order functions (MAP, KEEP, COMBINE). The data structures in the program are built to accommodate that.



We provide the overall structure of the strategy procedure.

The question is, because I have this motive and I impose this design, am I being hopelessly instructionist? Is it *my* project rather than the kid's project? How much flexibility is required for the kid to "own" the project? And, is the kid owning the project what makes it Constructionist?

Keywords

curriculum; ownership; constructionism; instructionism

What does "Constructionism" mean?

About 10 years later, I was working with Sherry Turkle and John Berlow at the Lamplighter School in Dallas, TX, the first elementary school where there were enough computers for children to have almost free access to them. The first space shuttle was about to go up, and the tension of waiting for it appeared in many representations on screens all over the school. "Even the girls are making space ships," one girl told us. But we noticed that although everyone had space ships they did not make them the same way. Some programmed their space ships as if they had read a book on "structured programming," in the top-down style of work that proceeds through careful

planning to organize the work and by making subprocedures for every part under the hierarchical control of a superprocedure. Others seemed to work more like a painter than like this classical model of an engineer's way of doing things. The painter-programmer would put a red blob on the screen and call over her friends (for it was more often, though not always, a girl) to admire the shuttle. After a while someone might say: "But it's red, the shuttle is white." "Well, that's the fire!"—came the reply—"Now I'll make the white body." And so the shuttle would grow, taking shape through a kind of negotiation between the programmer and the work in progress.

—Seymour Papert and Idit Harel, "Situating Constructionism," in *Constructionism* (Ablex Publishing Corporation, 1991)

The first thing to notice about this story is that no teacher assigned the project of making a space ship. Teachers did give assignments at Lamplighter, but there was also time for kids to do what they wanted. The outside stimulus of the Space Shuttle led to Seymour being able to compare student-initiated projects that happened to be depictions of the same thing in two different styles. My question is, which of the different characteristics of this anecdote make it constructionist?

- If a teacher had assigned drawing a spaceship, would that matter?
- How about if the teacher had said "make it as realistic as you can"?
- What if the common stimulus had been something abstract rather than a physical thing?
- The kids in this story were already familiar with Logo. What happened in the first week of school? Could a constructionist teacher start with the usual "Draw a square"? Or must we just sit the kids down in front of Logo and have them explore? (That would have been hard with Logo, really. But it can definitely be done with Scratch.)
- What if this week's class topic had been subprocedures, and the teacher had said "make whatever project you want, but try to use subprocedures"? Still constructionist?

The minimal definition of constructionism is that people learn best by making something, preferably a physical thing, that is available for public critique. But most people who call themselves constructionists, including Papert, connect the approach with the broader progressive education movement, in which locus of control is a crucial issue. That's why these questions don't have obvious answers.

I wasn't at Lamplighter to ask these questions. But somewhat analogous questions have come up for me in the development of the *Beauty and Joy of Computing* secondary school curriculum (BJC).

The Beauty and Joy of Computing

About a decade ago was the beginning of an ongoing campaign in the United States to increase the number of women and minorities studying computer science. The National Science Foundation (NSF) made a strategic decision to concentrate on high school (ages 14-18) students, reasoning that it's in those years that young people make career choices. My colleague Daniel Garcia and I decided to redesign our CS breadth course for non-majors, with the idea that the same course would serve as a high school course for all students. A pilot version of the course was offered in 2008; since 2009 it has been offered to more than 300 students every semester.

The United States doesn't have a central education authority; curriculum decisions are made by each state or by local school districts. It's very difficult, therefore, to introduce a new topic to the high school curriculum. But there is a non-governmental *de facto* national curriculum organization: the College Board, a consortium of colleges and universities⁷ that creates standardized tests that students take as part of the college application process. One project of the College Board is a set of 38 Advanced Placement exams in college-level curricula. The first 37 AP curricula were set by surveying freshman courses actually taught at colleges and universities, but the NSF partnered with the College Board to create a new course, "AP Computer Science Principles," in a subject that was, at the time, taught at

⁷ In the United States, a "university" is an institution that conducts research as well as teaching undergraduates (ages 18-22) and graduate students (over 22). A "college" is an institution that primarily teaches undergraduates, although there is no hard rule separating the two categories.

very few colleges. (There was already an “AP Computer Science A” exam, a traditional introductory programming course.⁸) AP courses are typically one semester long (about five months) at a college or university, but take a full school year at the high school, so that students are not deterred from AP courses by the density of college-level work.

Because CS Principles wasn’t a well-established college course, the College Board couldn’t use its usual approach of surveying colleges to determine what topics should be tested. Instead, they set up a committee of university and college faculty to invent *ab initio* a curriculum framework for a computer science course appropriate for non-majors. (College Board 2017)

Berkeley’s course, *The Beauty and Joy of Computing* (BJC), was initially designed before the CS Principles framework was written; we designed the course that we thought would best serve Berkeley undergraduates. One consideration was that the course should prepare students for our first course for CS majors, based on the brilliant text *Structure and Interpretation of Computer Programs* by Hal Abelson and Gerald J. Sussman. (Abelson 1996) At the same time, BJC was one of the pilot projects for CS Principles, and once the framework was written, we added content to BJC to satisfy it.

In the early years of BJC, we partnered with individual high school teachers who heard about the curriculum and wanted to try it. These early adopters tended to be either at private schools or in wealthy suburban school districts. But in 2014 the NSF solicited proposals for partnership grants proposed jointly by partnerships including a university and a secondary school district, and we decided it was time to jump into the deep end: New York City, a majority-minority school district (that is, most of its students are nonwhite) with 500 high schools. In New York, many students are learning English as a second language, and even some of the native speakers have below-grade-level reading skills. We made the crucial decision to include in the partnership an NGO, Education Development Center (EDC), with long experience developing curricula for K-12 education, to rewrite the curriculum specifically for high school. The EDC team for this project is headed by Paul Goldenberg and June Mark. Our partnership was funded for four years starting in January, 2015.

Tic-Tac-Toe

A program to play tic-tac-toe is a classic programming project. We decided to use this example as an ongoing project throughout the year, starting early in the curriculum, when students have written graphics programs but have no experience with data structures; this first part of the project was a program to draw the tic-tac-toe board and then display moves by two human players. A month later, when students have experience with list processing, they add code to detect and announce wins and ties. Two months later, they take the big step of writing a strategy procedure so that the computer can be one of the players.

We are now at the point where the connection with Seymour’s story from the Lamplighter School should be clear. We are, I guess, constructionists; but we are writing a project into a curriculum, with specific milestones based on specific programming techniques that we want students to practice. The students *are* making an artifact: a game program. Is this Constructionism?

That question was not just of theoretical interest to us. As it turned out, each step in the project design involved some disagreement among the curriculum team (six people at EDC, and me, spending half of each year at EDC, in Massachusetts, and the other half at home in California but working with EDC remotely). At each of these points of disagreement, we had to make a decision.

Drawing the board

The first point of disagreement came right at the beginning, with the drawing of the board. One of our team members, who had extensive Logo experience, wanted students to use their `square` procedure repeatedly, but instead of building a fixed 3x3 board into the code, parameterize the number of squares so that the same code could be used to draw a chess board or a sudoku board. Another team member,

⁸ CS A is taught using Java, probably the hardest programming language to learn because of its Byzantine syntax. That’s one reason a new course was needed to encourage non-geeks.

with prolific experience creating Scratch projects, wanted to give a sprite a square costume (three costumes, actually: an empty square, a square with an X inside, and a square with an O inside) and clone it nine times.

Each of these proposals was good in one way and less good in another way. Drawing the squares programmatically was a better fit with the rest of what was happening at that point in the curriculum, and in particular gave an opportunity for students to practice the use of Cartesian coordinates to position the squares. But using sprites made the next step, detecting users' moves in the form of mouse clicks inside a square, *much* easier, because the click would be captured by one of the nine sprites. If the squares were just lines drawn on the stage, students would have to use nested `if` commands or do complicated arithmetic on coordinates to determine the square into which the user wants to move.

The use of layered abstraction and modular arithmetic was part of the original intent of the project, but we learned visiting classes that students were having much more trouble figuring out how to handle the mouse coordinates than we'd anticipated. (See Figures 1–3.) We then had two choices: Debug our project instructions by treating only the 3x3 case and adding enough scaffolding so that students could complete the task, or abandon the original curricular goals by using the “when I am clicked” feature of Snap! to solve the coordinate problem below an abstraction barrier. (The silver lining would be adding something new, parallel processing, to the curriculum.)

Perhaps the most constructionist solution would have been for us to describe the desired result without making any suggestion at all about the algorithm to use. But students had never cloned a sprite at this point in the curriculum, so they would be unlikely to discover that solution (see Figure 4); and we already knew that the Logo-like “draw nine squares” solution made capturing mouse clicks too hard for many New York high school students without scaffolding.

2. You won't create the `draw gameboard` block that accepts any number of tiles as input until the end of the next page. The task is complex, so start by building blocks that specialize in each part of the task. One approach is described below. Be sure to test each block by itself before building a block that uses it.

a. Build `draw tile of size` to make a single square tile. You can use the same code from your `draw square` block.

Why abstraction? Why build `draw tile` when you could use `draw square` or just the script inside it? Making a specific block to hide the details does *two* useful things:

1. Less code (if written well) makes the code easier to read and debug.
2. If you later decide to create the tiles differently (maybe with thicker borders, filled with random colors, or by using `stamp`), it makes sense to edit `draw tile` (changing `draw square` would change *all* the squares).

b. Build `draw row of tiles of size` using your `draw tile` block to draw one row of tiles, like this:

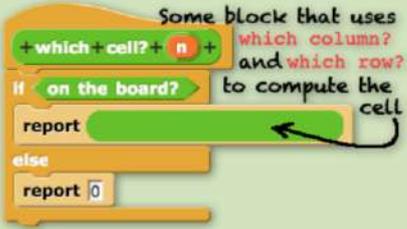


Figure 1. Teaching abstraction was a goal of the original version.

2. Build a predicate `is between and ?` that tests whether its first input is bigger than the second input and less than the third. You'll use this to test where the sprite is.

3. Build a predicate `on the board?` that uses your `between?` block to report `true` if the sprite is on the board and report `false` if the sprite is dragged completely off the board.

4. Build a reporter `which cell?` that tells which cell the sprite has been dragged into. One way to do that is to build blocks that reports `which column?` and `which row?` the sprite is in and then figure out how to use them to report the exact cell.



Remember: Your board isn't *only* for tic-tac-toe. You can draw nine-by-nine Sudoku boards, seven-by-seven Connect Four boards, and so on. So, these blocks will need the input `n` to specify how many rows and columns are in the grid.

One way to do this is to think of specific examples: What if the sprite is at the position (90, 110)? What if the sprite is at the position (-100, 80)?

If you need more help: download this file and import it into your project. It contains completed `which row?` and `which column?` blocks, but try to build them yourself first.

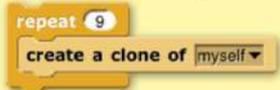
Figure 2. So was teaching Cartesian coordinates and conditionals.

There are many ways to write `which row?` and `which column?` You might think of using lots of `if` statements, but since the number of rows can change, you don't know how many to use. One convenient way is to divide the total size of the game board by the size of each cell. To use that result, you need to round down. The `round` block can do it, but it rounds to the *nearest* integer, up *or* down. The function `floor of` always rounds *down*, making it easier to use. Likewise, the function `ceiling of` always rounds *up*. The `floor` and `ceiling` functions are in the same block as `sqrt`.

Figure 3. We had to provide a lot of hints about converting mouse coordinates to board position.

3. You will be using the clone feature in Snap! to create the 3x3 Tic-Tac-Toe board. A clone is just a copy of an existing sprite, made under control of your program.

Why "under control of your program"? It's possible to make a copy of a sprite by hand; right-click on the sprite's icon in the sprite corral and choose "Duplicate." That's useful if you only want one copy, but in this project you're going to need *nine* copies, one for each square of the Tic-Tac-Toe board. So you'd like to say something along the lines of



The actual code in the project will be slightly more than this, but this is the central idea.

A clone is a complete copy of the sprite you say to copy. It has the same position, costumes, scripts, and so on. But...

- The clone doesn't appear in the sprite corral.
- Once you've made the clone, any changes you make *later* to the original sprite (such as adding or modifying a script) is *not* copied to the clone.
- Clones are *temporary*. Clicking the stop button will delete all clones in the project.

So the idea is that you write your program completely, then make whatever clones you need.

Figure 4. We had to teach what cloning means and how to do it.

We first attempted to solve the dilemma by compromising: We wrote *two versions* of the project instructions and let students (or teachers) choose which to use. This had the virtue that nobody on the team felt rejected, but it turned out to have few other virtues. In particular, the choice in this first part of the project affected later work, so that we would have to maintain two versions of the project throughout its stages. (Consider the problem of determining whether a player has won the game. The “draw nine squares” solution is structured as one program, with a single flow of control, and full access to the history of moves. The “nine sprites” solution, however, has this information distributed among the nine sprites, so the programmer must make extra effort in the second part of the project to communicate the moves to a central database. See Figure 5.)

So we all agreed that we had to make a choice. I preferred the computer science ideas in the “draw nine squares” version, which fit better (as they were designed to) with the overall story line of the curriculum. But I finally changed my vote, breaking a tie, when I convinced myself that many of our students will have used Scratch in elementary school, and will therefore have the same expectation as our Scratch-using colleague that clicks are to be caught by sprites. They would reasonably ask why we are asking them to do it with a much harder technique. We did edit the “nine sprites” curriculum page to call attention to its computer science ideas, although they’re not the ones we had been planning for. (See Figure 6.)

Detecting wins

There are only eight ways to win a tic-tac-toe game: three horizontal, three vertical, and two diagonal. This is a small enough number that it wouldn’t be too painful for students to check for a winning move with eight `if` commands, each checking explicitly for three squares with the same player in them, using square numbers built into the code. (See Figure 7.) This is certainly the solution that most students would invent, left to themselves.

But I had a curricular motive for this project that I was not willing to give up: teaching the use of higher order functions (`map`, `keep`, `combine`). BJC is my third time using this project for this purpose; I’ve done it before in Logo (Harvey 1997) and in Scheme (Harvey 1999). The fundamental idea is to keep the information about eight ways to win in a *list of lists* (Figure 8). Then detecting a win is just mapping the actual board position over this list of wins, and looking for a list of three Xs or three Os in the result. This can be done in a one-liner (Figure 9), but for pedagogic reasons we break the nested `map` calls into separate functions (Figure 10). The resulting curriculum page is shown in Figure 11.

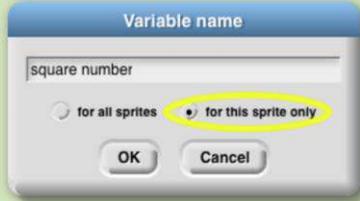
2. It was easy to detect illegal moves (a move in a square that's already occupied) because each sprite could test that without knowing anything about the board as a whole. But for the analysis in this lab, you need a *global* picture of the state of the game. (It's "global" not just in the sense of using a global variable, although that's true too, but rather that you need one single idea of the board, not nine local ideas about one square each.)

- a. Make a global variable `board` and, when the game starts, set it to a list containing the word "Empty" nine times.



As the game progresses, you're going to record each move by modifying this list, so that its value always reflects the actual state of the visible board.

- b. Create a sprite-local variable `square number`:



A **sprite-local variable** is like a global variable in that it doesn't belong to a particular *script*, but it does belong to a particular *sprite*.

When the initial sprite is cloned nine times, each clone will have its own version of this variable. That's how a sprite will "know" which square it is on the board.

- c. As you create the clones, give each the correct `square number`:

1	2	3
4	5	6
7	8	9

[Click here if you want a hint.](#)

- d. When a square is clicked, replace that square's entry in the `board` list with X or O as appropriate.

Figure 5. Creating global board knowledge from distributed-computing sprites.

This project hints at two important modern ideas in computer science. One is **object oriented programming**: Instead of a single sequence of instructions, the program has several *independent* objects (also called "agents" or "actors"), each of which has its own behavior. In this project, you can only click on one sprite at a time, and it then carries out a very quick task (changing costume) before you click another sprite. But a more complicated program might have one sprite send a *message* to another sprite, which would respond with its own behavior, while the first sprite continues with its own script.

The other idea is **parallelism** or **distributed computing**. Your project is running on a single computer, and the apparent ability of sprites to run simultaneously is an illusion provided by Snap!, which runs one script for a little while, then switches to a different script, and so on, then updates the display, then continues the first script. But you *could* have nine separate computers, one for each sprite. That would be overkill for this small project, but large web servers such as Google or Twitter have many thousands of computers all cooperating to handle the large amounts of data they require.

Figure 6. If life gives you lemons, make lemonade.

```

win? player
if
  item 1 of board = player and
  item 2 of board = player and
  item 3 of board = player
  report true
if
  item 4 of board = player and
  item 5 of board = player and
  item 6 of board = player
  report true
...
report false
  
```

Figure 7. Detecting wins the “obvious” or “natural” way.

```

+ TicTacToe wins +
report
list
  list 1 2 3
  list 4 5 6
  list 7 8 9
  list 1 4 7
  list 2 5 8
  list 3 6 9
  list 1 5 9
  list 3 5 7
  
```

Figure 8. Moving the knowledge of eight wins from the code into data.

```

+ win? + player +
report
map map item of board over over TicTacToe wins
contains list player player player
  
```

Figure 9. The one-liner version with nested map calls.

```

+ status of + triple +
report map item of board over triple

+ status of + winning + triples +
report map status of triple over TicTacToe wins

+ win? + player +
report
status of winning triples contains list player player player
  
```

Figure 10. Checking for a win using map.

3. Find out if a winning triple has actually happened:

- Click  to clear your tic-tac-toe board and initialize `board`. Play *one* game, *deliberately letting x win*.
- Try this:

`map` `item`  `of` `board`  `over` `list` `1` `2` `3`  

Note that the first input to `item` is empty.
- What does the result of the above tell you about the state of the game?
- If `{1, 2, 3}` isn't the triple in which x won the game, try part (b) again for that winning triple.

4. Make a block `status of triple`  that takes a list like `{1, 2, 3}` as input and reports a list of which of those squares are occupied by whom.

5. Use `map` and `TicTacToe wins` to make a block (`status of winning triples`) that reports the status of *all possible* winning triples, as a list of lists.

6. Make a block `won?`  that takes the letter X or O as input, and reports `True` if and only if that player has won the game.

Figure 11. Guiding students through the code.

Using this approach was controversial in the curriculum team. Higher order functions are exotic to people who've learned to program in conventional languages, such as Python, Java, or Scratch. It's only people who learned in functional languages (mainly Lisp dialects such as Logo, Scheme, or Snap!) who find this approach more natural. Also, there's nothing about higher order functions in the CS Principles framework; we're not required to teach it, and the biggest complaint teachers have made about BJC is that they don't have enough time to cram it all into the year. But at Berkeley, higher order functions and recursion are the two most important ideas that make BJC special. (A third is that we spend much more time on the social implications of computing than the framework requires.)

The central idea in computer science is abstraction, and the CS Principles framework reflects that. Higher order functions form a potent abstraction over lists; instead of the conventional loop over list items, with an index variable holding the position of the current item in the list, higher order functions allow the programmer to deal with the list as a single entity, without singling out individual items. The `TicTacToe wins` list is best understood not as a list of lists, and certainly not as a two-dimensional array, but rather as a *list of triples*, where "triple" is an abstract data type consisting of three square numbers in the range 1-9. So `TicTacToe wins` is an eight-item list of all the winning triples. `Status of triple` takes a triple as input, reporting a new triple in which the square numbers have been replaced by the player who owns each square. `Status of winning triples` maps `status of triple` over the eight winning possibilities, reporting a new list of triples. Then we're almost done; we ask whether that new list contains the list `{X, X, X}` or `{O, O, O}` depending on who moved last. `Contains` is a primitive block that answers that question.

We seem now to be very far from the space ships at Lamplighter. The project (tic-tac-toe) is assigned by the curriculum writers; students are told to use a particular method that even some of the curriculum team view as "unnatural." It's in a good cause, furthering abstraction, and I can testify that after a while it *becomes* natural. (I learned to program in Fortran, but the solution in Figure 9 is truly the first thing that occurred to me while working on this project.) And the code is beautiful; we have "Beauty" in the name of our curriculum and we take that seriously.

And kids do need to learn skills. Even in maker spaces, the current paradigmatic example of "Do It Yourself" learning, they don't leave kids to discover for themselves how to use a soldering iron or a table saw safely.

Again, it was proposed that we offer students two ways to write this procedure. But why? If we ask students to pick one, they have no basis for choice until they've tried both. So we'd have to have them do it both ways. And, if we're going to have them do it in the way we want, how does it help to have them also do it a different way? I think that proposal was more about trying to avoid conflict among the team than about the needs of students.

But what's important for our story is how much weight the word "project" carries in a curriculum. If something is an "exercise" then everyone easily keeps in mind what purpose the exercise is supposed to satisfy in the overall curriculum. But once something is a "project," completing the project becomes a purpose in itself, and the needs of the curriculum are easily seen as getting in the way of that.

And really, tic-tac-toe in our curriculum barely deserves the name "project." We want students to do it a particular way, for particular reasons, and we carefully guide them along that way. (Review Figures 5 and 11 if you need a reminder of what I mean by that.) At the end of each unit is a collection of about half a dozen optional projects, and those really *are* projects: We don't care how the student does them, or even *whether* the student does them, because they do not bear the weight of teaching the curriculum. They're for the student who finishes early and wants more to do. We try to pick projects that are somewhat relevant to the unit in which they appear, but even that isn't really important. They just have to be beautiful and/or joyous.

Tic-tac-toe strategy

The final stage of the project is allowing a person to play against the computer. This requires a *strategy procedure* to generate the computer's moves. But once we have status of winning triples showing what's where on the board organized by winning triples, this is pretty much just a matter of pattern matching.

We start by asking students to discuss in small groups how *they* play tic-tac-toe. (Figure 12.) (A slight digression: The CS Principles framework asks us to teach about "pseudocode." I don't believe that any programmer really thinks in pseudocode—certainly not the ones who have an expressive programming language to work with. What students do in this exercise may look like what the framework authors have in mind, but it's very different. The students are not thinking about a *computer program* in handwavy English. They're thinking about *themselves*, for which English is an appropriate language. People think in English; computers think in programming languages. Nobody thinks in pseudocode.)

By the way, the claim that people play tic-tac-toe using rules was controversial in our group, as was the claim that the first two rules are the same for all people. Some of us worried that students who had a different way to think about tic-tac-toe might feel excluded. So, take a minute to think about how *you* play tic-tac-toe (or how you played it as a child; you've probably graduated to Sudoku and its variants now). If you use rules, think about what your second rule is. (We give the first rule in Figure 12.)

The first rule is, "If you can win on this move, do so." If the computer can win on this move, it must be by filling one of the eight winning triples. What must be true of a triple for the computer (which is playing O) to be able to win using it on this move? The triple must contain two Os, and no Xs. (The third square of the triple will be empty, which is represented by the square number.) Figure 13 shows an excerpt from the curriculum in which students are asked to answer that question, and then use the answer to write the strategy procedure. Here is an excerpt from the program:



“find first item (of the list of triples) such that the number of Os is 2 and the number of Xs is 0.” Even if you aren’t familiar with the notation for higher order functions (and the students are, by this point in the curriculum), you can see how the structure of the code matches the structure of the problem you’re trying to solve.

Note that this is *not* the same as saying that the language is “like English.” It isn’t, except in the sense that all programming languages are like English: the names of primitives are meant to suggest what they do. But the *arrangement* of those primitives—the syntax of the language—is quite different from English, as it should be. Rather, the point is that the higher order functions, like all abstractions, let you talk in terms of the problem you want to solve (triples, Xs, Os) rather than in terms of computer hardware (index variables, looping, memory addresses).

Figure 13 shows the excerpt from the curriculum in which students are asked to express the first rule in terms of triples, and then use that to guide writing the code. Figure 14 shows the complete solution, apart from a couple of helper procedures.

How Do You Play Tic-Tac-Toe?

Almost certainly, you have a bunch of rules of the form “if such-and-such is true, then play here.” These rules have different degrees of urgency. For example, the most important rule is “if I can win on this move, then play in the winning square.”

Before you read further, answer these questions, discussing them with your partner and with other students if necessary:

For You To Do

3. What’s the next
4. Can you express

The if-all-else-fails rule is

```
script variables ready-to-win triple ▶  
set ready-to-win triple to  
find first item such that  
  number of player in □ = 2 and  
  number of opponent player in □ = 0  
from  
status of winning triples
```

For You To Do

5. Why is the center the best choice? Why is a corner second best? You should have a really short, logical, convincing explanation. Compare your explanation with those of other students.

These three rules (if you can win on this move, do so; the one you answered in question 3 above; and the just-pick-a-square rule) are part of any human-like tic-tac-toe strategy, and they are the minimum acceptable strategy for this assignment. But these three rules alone aren’t enough to play tic-tac-toe really well. Most players develop additional rules that come after the first two but before the last-resort rule. Human players don’t always think alike about those middle rules; we’ll make suggestions later, after you finish the minimal assignment.

Choosing the Best Move

To simplify the project, assume that the human player will always move first (playing X). So the computer is playing O. Make a block `next move for computer` that will report the square number (1-9) into which O should move.

Figure 12. How do human beings play tic-tac-toe?

6. The **won?** block works by looking for a triple in which all three slots are O, or all three slots are X. Describe precisely what the slots of a triple will contain if O can win on this move by filling its last square. (There's more than one correct answer to this question, but not very many of them.)

7. Write the **winning square for** block. If there is no winning square for the input player, it should report 0.

You might find this block in the starter file useful:

find first item such that **from**

It's like **keep**, except that it reports just one matching item, rather than a list of all matching items. (If there are no matching items, it reports an empty list.)

Also, you may find it helpful to invent these blocks:

opponent **number of X in list**

Want a hint?
(Do you see why it's important to represent empty squares with the square number rather than the word "Empty"?)

8. Use the block you just wrote to implement **can win now?**

Figure 13. Guiding the student to convert a rule to a program.

The figure shows a sequence of Scratch code blocks:

- +winning+square+for+ player +** (comment)
- script variables ready-to-win triple** (variable)
- set ready-to-win triple to** (set block)
- find first item such that** (loop block)
 - number of player in = 2 and** (condition)
 - number of opponent player in = 0** (condition)
 - from** (loop block)
 - status of winning triples** (comment)
- if empty? ready-to-win triple** (if block)
 - report 0** (report block)
 - else** (else block)
 - report** (report block)
 - find first item such that** (loop block)
 - is a number?** (condition)
 - from ready-to-win triple** (loop block)
- +can+ player +win+ now?+** (comment)
- report winning square for player > 0** (report block)
- +next+move+for+computer+** (comment)
- if can O win now?** (if block)
 - report winning square for O** (report block)
- if can X win now?** (if block)
 - report winning square for X** (report block)
- report best empty square** (report block)

Figure 14. The complete strategy procedure.

Efficiency

Note that in our solution, `can __ win now?` calls `winning square` for __, and so next move for computer ends up calling `winning square` for __ twice for a rule that succeeds. Of course we could use a local variable to save the value from the first call and avoid the second, but we are adamant that in a first course that aims to attract nontraditional students to computer science, efficiency doesn't matter. What matters is promoting abstraction all the time.

Differentiated instruction

The directions we give students are narrow, because we have specific purposes in mind and also because we are under pressure from teachers to ensure that even ill-prepared students succeed in the activities. We do have "Take It Further" activities, for more advanced students, with much less scaffolding. (Figure 15.)

- C. Here's a completely different approach to tic-tac-toe: Make a permanent list of *key-value pairs* in which the key (the first item of a pair) is a possible state of the board  **board** and the value (the second item) is a number saying where to move next. This would be really messy if you had to account for every possible board position, but you only need to consider the ones in which neither player has won yet. Also, you can make a special case for the computer's first move. If the computer is playing O, there are really only three ways X can have made the first move: the center, a corner, or an edge. If X moved in a corner, *renumber* the board so that the corner where X moved becomes 1, and similarly for an edge move. (This will take some effort in the program; you can't just wish away the fact that X may have moved in square 3, say, rather than 1.) Then the computer can move just by looking for the current board situation in the list.
- D. Here's yet a third approach: Have the same key-value list, but have it start out empty. If the computer moves next and the current board situation isn't in the key-value list, make a *random* move, keep a record of it, and see what happens. If the program loses a game, it should note that its last move wasn't a good one, and next time that same board position comes up, try a different random move. This *learning* version is complicated to write, but it's much more like a serious artificial intelligence program, and it's not limited to tic-tac-toe. You can use the same algorithm to play any strategy game. Once you've written the program, you can let it play *against itself* a few thousand times and save the resulting key-value list.  

Figure 15. Enrichment activities.

We stole the twisty-road signs from Donald Knuth's *T_EXBook* (Knuth 1984). A double-twisty-road exercise could be a month of hard work.

In the other direction, we provide for students who need even more handholding than we expect with the "click for a hint" links.

What does "Constructionism" mean? (reprise)

Or, getting to the heart of the matter, am I a constructionist?

The most pessimistic answer would be that I crossed the line as soon as I decided to be a curriculum writer instead of directly working with kids. Actually, I'm less worried about violating Seymour's philosophy than that of Martin Buber, who writes about the human relationship between student and teacher as the most important part of education. I guess a more positive way to say this is that teachers

and curriculum writers are both important, and curriculum writers do have a big influence, but I shouldn't be surprised that the front lines in education are in classrooms and not in curriculum team meetings.

But that's not a very good answer, because it lets me off the hook. It means that I can write any old curriculum, without worrying about whether it's progressive or not, artifact-creating or not, just whether it contains the right ideas and information.

Another answer that appeals to me is that the duty of a constructionist curriculum writer is *not* to hold students' hands, but rather to leave space for them to create something new. I fear that we haven't done that enough. The original Berkeley version of BJC, aimed at college students, was better about this. If our purpose is to improve access and equity in computer science, we have to succeed with all, or at least most, of real high school teachers and students, in classes in real schools, not all of which will be progressive, let alone constructionist.

A big part of our problem is that we are constrained by a standardized test that we didn't write. Teachers are terrified of getting to the exam in May with unprepared students. To avoid that, students not only have to succeed at our lessons; they have to do it *quickly*. When I was a high school teacher, I had no authority to answer to, no standardized test, and so I could be relaxed about how fast students worked. We have been considering a version of BJC that would *not* be labelled as an AP CS Principles course. This would let us leave out many things in the framework that aren't part of the BJC story line, so, ironically, it could end up having more and deeper computer science than the CS Principles version. But most importantly, teachers could teach their classes at whatever pace seems right for that particular group of students. That would let us leave out some of the scaffolding, or at least bury more of it under click-for-hint links.

By the way, a student's AP score is based not only on the sit-down exam in May, but also partly on a programming project that the student does in class and submits to the College Board for grading. (Teachers are required to set aside 12 hours of class time for this project; it can't just be homework because some students don't have computers at home. If a class period is 45 minutes, that's three weeks of instructional time gone—one more reason the teachers feel rushed. There is also an 8-class-hour "computational artifact" addressing the social implications of computing.) The required programming project makes this class at least slightly constructionist: There's one artifact created entirely by the student.

As I write this, I am leaning toward not feeling guilty about wanting students to build programs in specific ways. Ideally, I think, there would be more variety about this in the curriculum; sometimes we'd insist on them doing it our way, but other times we'd let them work without guidance and then have groups of students compare their solutions, and *then* show them our solution. But we are deterred by time pressure from doing this.

There's a lot more curriculum I want to write. I've mentioned the non-AP BJC. Some schools have asked for a semester-long subset of BJC. The project that most excites me, although maybe nobody would actually use it, is "BJC year 2," a course that would introduce object oriented programming, dive deeper into functional programming, and perhaps write an interpreter for a programming language. We've also talked about mix-and-match modules for schools that have robots, 3D printers, and other programmable toys.

On the other hand, I really miss teaching kids!

Acknowledgments

This work was funded by the National Science Foundation under grant number 1441075. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.



References

- Abelson, H. and Sussman, G. J. (1996) *Structure and Interpretation of Computer Programs 2/e*. MIT Press, Cambridge.
- College Board (2017) *AP Computer Science Principles: Course and Exam Description*. College Board, New York.
- Harvey, B. (1997) *Computer Science Logo Style 2/e*. MIT Press, Cambridge.
- Harvey, B. and Wright, M. (1999) *Simply Scheme: Introducing Computer Science 2/e*. MIT Press, Cambridge.
- Knuth, D. E. (1984) *The T_EXBook*. Addison-Wesley, Menlo Park.
- Papert, S. and Harel, I. (1991) Situating Constructionism. In *Constructionism*. Edited by I. Harel and S. Papert. Ablex Publishing Corporation, Norwood, NJ.

Social Gears – a Constructionist Approach to Social Studies

Arthur Hjorth, arthur.hjorth@u.northwestern.edu

The Center for Connected Learning and Computer-Based Modeling, Northwestern University, USA

Abstract

The Gears of Seymour Papert's childhood have been a persistent analogy for our work in the Constructionist community for decades. It is a productive analogy that has helped concretize many central themes in Constructionism: It frames our view of knowledge as understanding the inner workings of a system; it forefronts the existence of an *external representation* of some domain knowledge; and it emphasizes that learning happens through the manipulation of this external representation by alignment of an internal, mental model with an external, physical model. Gears are powerful, maybe *because* their cause-and-effect is simple: one cog moves another cog which moves another cog which moves another cog. Always at the same ratio, and at a rate pre-determined by whichever cog we apply torque to. Importantly to our work as an educational research community, it makes studying *thinking* about cogs relatively straight forward.

But what if the gears were *social*? What if they have inner lives, mood swings, wants and desires, and work under the constraints of social pressures and modern family life that they must negotiate, collectively and individually, in order to organize their turn ratios? Does it change how we should design Constructionist learning environments and activities? Does it change how we should study students' thinking? I will not claim that the deterministic nature of the Gears-analogy *caused* Constructionism to focus on deterministic subjects. But maybe the focus on STEM and programming have led us to not explore and interrogate potential shortcomings of the analogy?



Students reasoning with different representations of Urban Planning

I address these questions based on my experience with designing, implementing, and studying Constructionist learning in social studies classrooms at the high school level. I present “Complex *Social* Systems Thinking” (CSST) as a guiding framework for designing learning activities and for studying students' thinking. I then present a set of Constructionist learning activities that I have designed on Urban Planning, and present data to exemplify what CSST looks like “in the wild”. Finally, I discuss the relationship between Constructionism and CSST and present my hopes for the future of Constructionism in social sciences education.

Keywords

complex systems thinking; social studies; design; qualitative analysis

Scratchmaths: a Positive Outcome for Constructionism at Scale

Richard Noss , r.noss@ucl.ac.uk
University College London, UK

Celia Hoyles, c.hoyles@ucl.ac.uk
University College London, UK

Abstract

From September 2014, all primary schools in England have been required to teach the national computing curriculum, which includes designing and building programs. This presentation will discuss some of the challenges in implementation as part of describing the two-year ScratchMaths (SM) project. SM designed a comprehensive curriculum for Year 5 and 6 pupils (aged 9-11 years) that maps directly to the computing curriculum, seeks to develop pupils' programming skills as well as exploit these skills to explore key mathematical concepts. The SM curriculum comprises detailed student activities for about 20 hours per year over two years, with teacher support materials and professional development days. SM has been implemented in over 100 schools across England. The project was evaluated qualitatively through design, survey and observational research. In addition, SM was evaluated by an independent research team employing a randomised control trial.

We will report a surprising result on the constructionist path from the RCT, namely that:

ScratchMaths led to a small increase in computational thinking as measured by the test used at the end of the first year of the trial. (The independent report on SM project).

This is surely good news for the constructionist community, not least as it was shown that this increase was significantly greater among 'disadvantaged students' as measured by their eligibility for free school meals (a standard proxy measure in UK). We will have plenty to say about the substantive issues concerning this finding and the quantitative methods employed. While it is relatively straightforward to identify performance in clearly pre-defined skills, it is much more difficult to do the same for necessarily open questions: How does programming (in Scratch) engage students in ways that supports them see learning as worthwhile? How do learners express themselves using Scratch? What can a Scratch-aware learner do that he/she couldn't have done without Scratch?'. And of course, what was meant by computational thinking.

We will also report the second major finding of the SM project, namely that:

...even though a positive correlation was found between computational thinking and mathematics, ScratchMaths did not increase mathematics attainment during the trial period, as measured by Key Stage 2 tests. (The independent report on SM project)

Again, there is much to say to seek to explain this outcome: it raises questions again of the validity of the measures, but also of teacher confidence and the 'fidelity' of the intervention as implemented in the second year. this is a problem for researchers in the scientific as well as social domains. For example, medicine – routinely regarded as the arena in which the most rigorous research is carried out, the 'gold standard' for research methodology – is facing much the same challenge. For example, the Medical Research Council in the UK, acknowledges that when dealing with complex interventions, one must be aware that complex interventions are built up from a number of components, which may act both independently and inter-dependently. The components usually include behaviours, parameters of behaviours (e.g. frequency, timing), and methods of organising and delivering those behaviours (e.g. type(s) of practitioner, setting and location). It is not easy precisely to define the "active ingredients" of a complex intervention. (Medical Research Council, UK April 2000.)

Educational interventions are certainly no less complex than this. So even before we reach the problem of knowledge and change, we have to admit that it is difficult to be clear what are the key components

of a given complex intervention. In conclusion, we admit to be quite surprised at the above outcomes, they give us all we think food for thought and we note we would have missed them if we had closed the door on methodologies that have, up to now, been relatively taboo in constructionist circles.

Keywords

Scratch; computational thinking; mathematics; methodology; randomised control trial

References

Benton, L. Hoyles, C., Kalas, I & Noss, R. (2017) Bridging Primary Programming and Mathematics: preliminary findings of design research in England Digital Experiences in Mathematics Education, pp 1- 24

Benton, L. Kalas, I; Saunders, P; Hoyles, C; Noss, R. (in press) Beyond Jam Sandwiches and Cups of Tea: An Exploration of Primary Pupils' Algorithm-Evaluation Strategies" J of Computer Assisted Learning

Programming in Lower Primary Years: Design Principles and Powerful Ideas

Ivan Kalaš, kalas@fmph.uniba.sk
Comenius University, Bratislava, Slovakia

Abstract

Latest national computing education strategies – often installing the beginning of the new mandatory subject into the lowest years of primary education – embrace programming as a key instrument of computational thinking. National curricula usually set ambitious requirements for primary computing education, listing essential *computational constructs* and *practices* to be mastered by primary pupils (e.g. *use sequence, selection, and repetition in programs; or design, write and debug programs that accomplish specific goals...*). While the research findings within our recent ScratchMaths project suggest that the intervention which we developed is a viable strategy to meet the expectations of national curricula in years 5 and 6, the question remains how to implement them in years 1 to 4 (i.e. with the age group of around 5 to 10, depending on educational system).

There are numerous portals and on-line resources claiming to have the answer to that question. Our main concern, however, is that those resources and the expertise behind them often originate from after-school experience, secondary or higher education practise or individualised home “edutainment” – focusing on isolated flashes of learning, often neglecting complexity of important basic computational constructs and practices, plus failing of advantages of primary education.

In our on-going research and development, we strive to better understand what distinguishes after-school programming environments and approaches (in the *code.org* style) from systematic and appropriate pedagogies for lower primary computing. In my plenary I will present our emerging approach for transforming so called “basic” computational constructs into thoroughly constructed and iteratively verified gradation of short units of tasks which the pairs of pupils – and then the entire class – try to explore and solve, envisage and discuss, compare, share and explain, exploiting the 5Es pedagogical framework of the ScratchMaths project. I will give reasons for our decision to develop new set of programming environments and I will formulate our key design principles and explain which powerful ideas we want pupils to experience and explore so that they get the opportunity to gradually build deep understanding of essential *computational constructs* and *practices* in appropriate progression.

Keywords

primary programming; programming environments for primary pupils; developmental appropriateness

1 Background

Latest national computing education strategies – often installing the beginning of new mandatory subject into the lowest years of primary stages⁹ – embrace programming as a key instrument of computational thinking. Renewed national curricula usually set ambitious requirements for primary computing education, listing essential computational concepts, procedures and processes to be mastered by primary pupils (e.g. *understand the concept of abstraction, logic, algorithms and data representation..., use sequence, selection, and repetition in programs;... design, write and debug programs that accomplish specific goals... etc.*). Besides that, new national strategies also emphasise building productive connections between computing and other subjects, mathematics in particular. Naturally, new situation increases demand for new learning content and interventions – systematic, complex and consistent, and new programming tools – inspiring and powerful yet developmentally appropriate for

⁹ in some countries starting at the age of 6 or 7, in some others at 5 or even 4

pupils and the needs of formal education. For years 5 and 6¹⁰, Scratch (Maloney et al., 2010) is considered to be such environment, with functionality and affordances which inspire educators, researchers and developers to exploit it as a new means for cultivating computational thinking and contributing to the development of mathematical thinking as well.

This is exactly the trajectory we have recently followed in the UCL IoE ScratchMaths project, however, while our research findings in the ScratchMaths project (see Benton et al, 2016; 2017, 2018a; 2018b; Kalaš, Benton, 2017) suggest that the intervention we developed might be a viable answer to meet the requirements of national curricula for years 5 and 6, the question remains open how to implement these requirements in years 1 to 4 (i.e. approximately with the age group of 5 to 10, depending on different educational systems).

2 Analysis

Indeed, years 1 to 4 of the primary education are a real challenge for researchers and educators in the field of computing. While there are numerous portals and on-line resources addressing that challenge, our main concern is that those resources – and the expertise behind them – often originate from after-school experience, secondary or higher education practise or individualised home “edutainment”, focusing on isolated flashes of learning, exploring bits and pieces of computer science essentials in unspecified order, often neglecting complexity of important basic *computational constructs*, failing most of advantages of primary learning environment and lacking consistent curriculum design and complex pedagogical framework.

That is why we have recently focused on lower primary years and set out for studying *cognitive difficulties of computational constructs*¹¹ which are traditionally considered essential, easy, and thus appropriate for that age group. In doing so we are encouraged by the authorities of 70s, 80s and 90s, referring here e.g. to Papert (1980), Pea (1985) or Perlman (1976) who observed ... *children becoming overwhelmed when introduced to multiple new concepts...* through her Tortis Slot Machine system. Inspired by those observations she started considering the cognitive difficulties behind some aspects of programming (see Morgado et al., 2006, p. 4). More recent research literature – if focused at the lower primary age group – often studies the potential of programming in the context of teaching mathematics (see e.g. Clements, 2002; or Lewis and Shah, 2012). Those that explicitly study the development of programming and computational thinking in the lower primary age group, like (Futschek and Moschitz, 2011), (Touloupaki et al., 2018), (Wilson et al., 2013) or our own earlier research (Moravčík and Kalaš, 2012), are still very rare.

Let us note however that it is not possible to study cognitive demand and appropriateness of any *computational constructs* without selecting a programming tool or designing and developing one, and in parallel with studying the *constructs* also study the relevance of that tool for the age group of our interest. Therefore we conducted an analysis of existing programming tools and environments¹² in which we decided to identify and exclude from the consideration:

- tools which we regard as ‘closed programming quizzes and puzzles’¹³,
- tools which support *computational constructs* that we a priori consider too complex or inconsistent for the lower primary age group,

¹⁰ that is, for pupils of the age group around 9 to 11 – in some countries last years of primary (ISCED 1), in others first years of lower secondary (ISCED 2)

¹¹ As explained in (Kalas, Benton, 2018), we prefer to think about *computational constructs* rather than *computational concepts*. In our understanding, computational constructs comprise *computational concepts* (like *procedural abstraction*, *variable* or *iteration*) and *computational procedures* associated with exploiting that concept, for example interpreting a sequence of commands, modifying it, transforming it by reducing some of its steps etc.

¹² we will report on that aspect of our work in depth elsewhere

¹³ as aptly nicknamed in (Grizioti, Kynigos, 2018)

- tools which do not accommodate with our conception of appropriate constructivist pedagogy of programming¹⁴ for the lower primary years, nor allow to serve the whole and systematic learning process,
- tools which do not comply with our technical requirements¹⁵.

As this selection process excluded all tools presently known to us, we decided to design and develop a new programming environment (in fact, a compound interface with several closely connected environments or microworlds arranged in a gradation) for primary years 3 and 4 (i.e. for the age group of app. 8 to 10).

3 Method

In summer 2017 we started the development of Emil, a virtual robot-like character, positioning the intervention into the 'middle' of the lower primary years – to win more flexibility for later possible alterations and fine-tuning. At present (summer 2018) we have completed the development of the content and programming tools for year 3. It is designed for about 12 lessons; however, several activities may easily be expanded, based on the learning goals, erudition and experience of the teacher¹⁶. Also, we have already launched parallel development for all other years of lower primary computing education – addressing pupils aged 5 to 10.

The interventions for years 1, 2 and 3 are being inspired by our previous development of Thomas the Clown, see Moravčík, Kalaš (2012), enriched with our recent experience from ScratchMaths project, Emil development and all other valuable experience gained in the recent years. The plan is to develop and support around 10+ programming lessons per year, supported by three or four new programming environments (to be designed and developed by our team in the next 18 months). All lessons (and programming environments) will acknowledge our understanding of the role of programming as an instrument for learning by exploring and solving problems. Thus, the whole intervention will offer rich connections (bridges) to other subjects and areas.

In parallel with this we continue the development of Emil for year 4: The plan is to build year 4 (i.e. pre-Scratch) programming intervention of 16 to 20 lessons, where most of the lessons will address multiple components of the new computing curricula¹⁷. In Emil year 4 intervention we plan to create more opportunities for open constructionist learning by designing one of the environments as open (although restricted) programming tool.

Emil intervention for year 3 includes:

- three programming applications or microworlds,
- gradations of thoroughly designed units of tasks, sometimes with one solution, with no solution or several 'possibly good' solutions – so that important discussions are being ignited to consider arguments for or against pupils' strategies. All tasks themselves are open for informed content developer and thus may be either rebuilt or redesigned¹⁸ to fit the specific needs,
- a workbook for pupils with series of worksheets to use in the lesson and other tasks to solve as extra activities or as homework,
- teacher materials of (a) basic and (b) advanced levels (where basic materials are designed for teachers implementing our interventions for the first time),

¹⁴ for example, with the possibility to first build the need for a computational construct, and then discover it by oneself, thus constructing its understanding

¹⁵ which are multi-platform application, with offline and/or online versions

¹⁶ Recently one of our design teachers commented one of the tasks as being rich enough to be extended to a full lesson – of mathematics – by itself. This matches our plans and we expect teachers to make similar transformations by themselves – into mathematics, traffic education, art and design, technology etc. We also want to stimulate natural and self-motivated extensions to after school activities and explorations at home.

¹⁷ speaking from the broad international point of view, see e.g. Webb et al. (2018)

¹⁸ although we do not expect teachers to develop their own tasks or modify the ones we designed and trialed for them

- strategy and content for the professional development, and
- some supplement tools and materials to support additional activities.

In the project we apply the design research strategy (as presented e.g. in Nieveen, Folmer, E. 2013), proceeding in numerous short iterations. We closely collaborate with three design schools and five of their classes. These schools represent wide range of schools in Slovakia, including a modern big state school in a small town, a school belonging to a network of schools governed by a church, and a small and new private modern constructivist school. Regular visits to the classes inform our development by (a) *observations* – after several initial months when each lesson was conducted by a member of our team we completely handed the teaching process over to the generalist class teachers themselves, (b) *talking to pupils* (in the lesson) and *teachers* (after the lesson), and (c) by *collecting and analysing pupils' worksheets* – as we use one or two worksheets in every lesson as we want pupils to:

- (I) record their work, sometimes developing their own notations for that,
- (II) read the assignments from the paper (sometimes these are given in the screen, sometimes only in the worksheets),
- (III) design and share similar tasks for their classmates etc.

We also use some extra worksheets to be solved by the pupils after the lesson to assess their achievement. Figure 1 illustrates a task from such worksheet. The scene on the left shows the situation and a plan (a program) for Emil's next journey. He will fly over the coins and buttons, on his way collecting everything into a box (as if he was 'buying' buttons). Pupils are to choose one of three possible outcomes of that journey. Note that solving this problem means reading a plan and interpreting it, considering dynamically changing situation in the scene: when flying from position marked with 3 to 4 Emil will collect two coins and only two buttons as one of the three in the line will have been collected already.

In our visits to schools we pay special attention to the wording of the assignments, as we often observe that pupils are not sure what to do even if we think we managed to post the task in a very simple and straightforward way.

Pupils are always working in pairs, which for the most of our schools is rather uncommon scheme of work. Traditional „lecturing“ is still a frequent form of teaching – even in primary school, therefore teachers were – at the beginning of our collaboration – often uncertain about the pedagogy, in which there is no space for lecturing, pupils work in pairs (but in parallel individually fill in their worksheets) and every now and then are invited to meet on the carpet with the worksheets in their hands to discuss the tasks (at least once for each unit of tasks). The focus of our teacher materials is on what to discuss with the pupils and when. These common discussions play several important roles in our interventions:

- pupils learn from each other by explaining and comparing their solutions, and listening to others,
- pupils discover by themselves how important the worksheets are and the way they have recorded their solutions as they later use those histories to argue and discuss their results and strategies¹⁹,
- by asking the questions and organizing the discussion with demonstrations (usually by the pupils themselves) using the teacher laptop projected on the screen, teacher has important opportunity to assess the group and everyone's achievement.

¹⁹ They started writing down the movements (commands) of Emil even before we proceeded from the *direct drive* style of control to *computational control* (programming). We study different level of control in (Kalaš, Blaho, Moravčík, 2018).

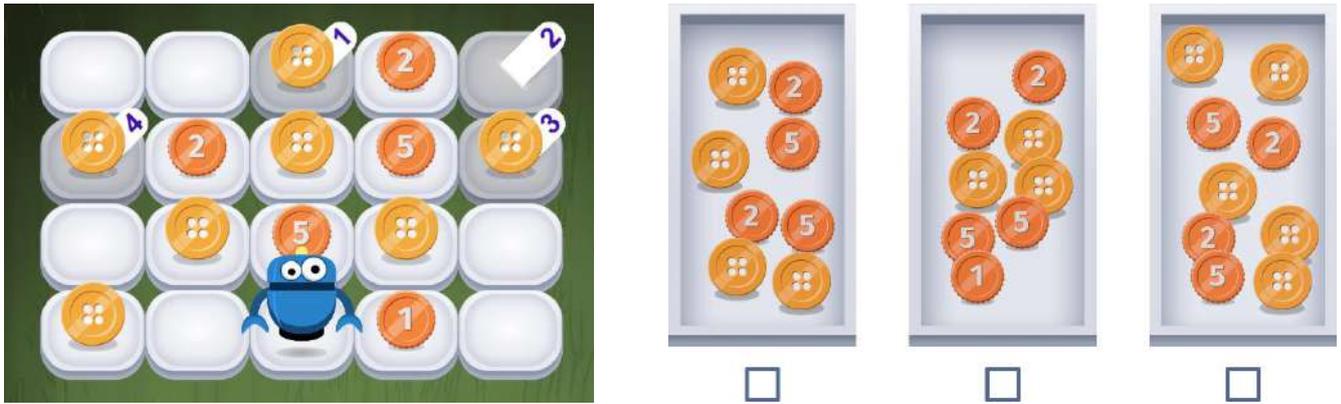


Figure 1. Emil is going to 'buy' buttons again. Check the box to show which shopping Emil will get if he follows the plan as indicated on the left?

Note that our programming environments do not give any feedback to pupils – a big surprise for them and their teachers at the beginning. Instead, we encourage the pairs – and then the whole group – to discuss each solution and decide whether they deem their solution correct. Note also that some of the tasks have several solutions, sometimes disputable 'could be solutions', sometimes no solution at all – situations which are surprising and very rare in our schools. Our observations prove however that after one or two lessons this approach is smoothly accepted both by the pupils and the teachers.

4 Findings so far

The strategy of short and frequent iterations in our design research has helped us move from mostly *experience- and intuition-based* development to more systematic design and better understanding of what the basics of primary programming comprise. While we agreed on several *design principles* from the beginning of the development for year 3, some others have appeared and crystallised later in the process. At present we can briefly formulate our design principles as follows:

- i. the intervention must be an opportunity and formative experience for every pupil in the class, clearly illustrating the perception of programming as an instrument for exploring the world, experiencing powerful ideas²⁰, and controlling an agent, in harmony with what Ackermann (2012) identified as the most appealing role of programming for very young children,
- ii. the learning process must be structured into gradations of units of a small number of tasks (4 to 6, as we see it now). The lead principle of this organisation must be a coherent arrangement of *computational constructs* structured by the increasing cognitive demand (as advocated e.g. by Perlman, 1976). These steps of increasing complexity must be small and intuitive²¹,
- iii. the tasks should exploit natural intrinsic motivation of pupils, the tasks and programming tools must be developmentally appropriate,
- iv. the pupils work in pairs, but individually fill in their worksheets,
- v. there is no space for lecturing, the whole learning process is implemented by pairs of pupils collaboratively solving the tasks, then the whole group discussions (explaining, reading, demonstrating, arguing, listening...) about the solutions and problems²² – scaffolded by the teacher,

²⁰ For the origin of the concept of powerful ideas in learning, see (Papert, 1980) but consider also Kay's recent definition from the *Thinking about Thinking about Seymour* symposium in 2017: *An idea is powerful if it changes the context in which we think.*

²¹ not requesting any additional explanations

²² Beside our own observations of this principle in the design schools, we are also supported by literature: Harel and Papert (1990) highlight the cognitive benefit of generating verbal explanations, which helps to clarify ideas. Noss and Hoyles (1996) assume that the process of reflection and explicit articulation required to generate these explanations is a key part of the constructionist learning.

- vi. the programming environments give no explicit feedback other than – in the *computational control* stage of the work – running the program and showing each step of Emil in the screen, thus displaying the final state of the process²³,
- vii. the intervention should be delivered by the generalist class teacher, rather than engaging a computing specialist. Why is this principle so important for us in the primary education? We believe that the intervention must be accepted by the teacher and integrated in the whole learning process of the pupils, making explicit connections and bridges to all other subjects whenever possible and productive. We find this – at this stage of school – more important than having a formal computing education.

Recent revisions of several national curricula set ambitious requirements for primary computing education, listing essential *constructs* and *practices* to be mastered by primary pupils. Whether those *practices* and *constructs* are developmentally appropriate for lower primary years, how to interpret them and how to arrange the intervention so that it helps pupils ‘discover’ them by themselves in a true constructivist way remains the hardest and most crucial question.

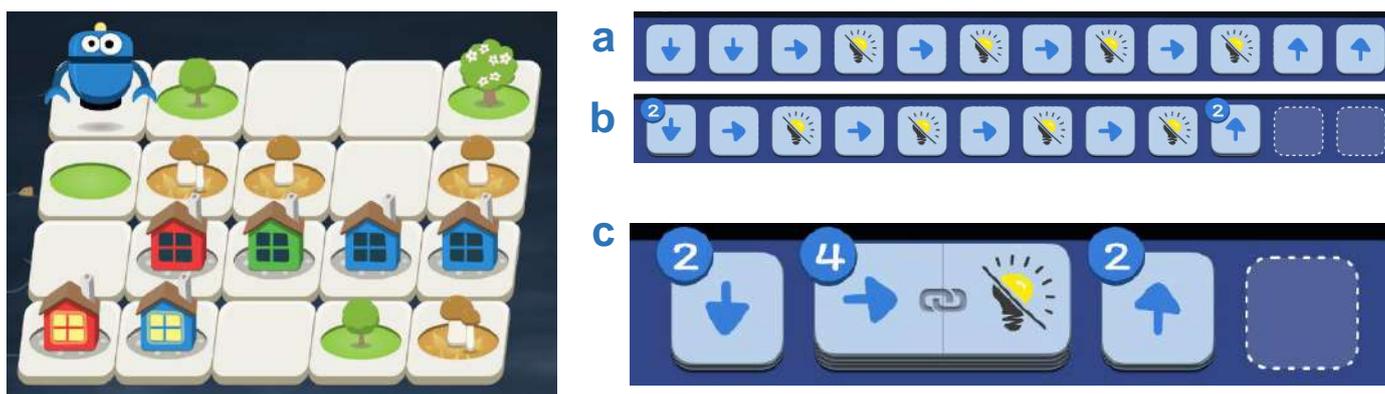


Figure 2. When pupils directly control Emil in the scene to solve a problem (the task here is: Light all houses and come to the blooming cherry tree) by clicking or touching the commands, a record of the steps is automatically collected in the panel above the scene (illustrated by a). However, the panel has limited number of positions thus the length is a real issue. Later in the succession, identical cards are cumulated into ‘piles’ (see b above), thus “saving” more positions in the panel. Then pupils start to notice repeating pairs of cards and start connecting and cumulating them up into piles of ‘double cards’ or ‘triple cards’ (see c above).

In our experience, the complexity of so called ‘basic’ *computational constructs*²⁴ is often trivialised, or even not recognized nor properly understood. That is the reason why the most important design principle for us is to discover and respect their complexity and identify possible gradations of ‘micro steps’ leading to their deep and sustainable understanding. In Emil year 3 intervention we mostly focus on three such constructs, namely *sequence* (or more broadly, *order*), pre-constructs²⁵ of *repetition* and pre-constructs of *abstraction*. Figure 2 illustrates (in a simplistic way) some of the steps leading to discovering the *construct* of *simple repetition* (note however that the process of full understanding and adopting of this *construct* will continue into higher years of primary and lower secondary education).

Beside the design principles we are also constantly trying to clarify and reformulate the powerful ideas of computing which pupils are experiencing while solving the tasks. Towards the end of the first year of our iterative design and assessment of the environments in the design schools we identify the key powerful ideas of the Emil year 3 intervention as follows:

²³ explicit feedback, however, would mean that the application itself evaluated the final state and communicated to the user whether the solution was correct or not

²⁴ i.e. concepts and related procedures

²⁵ by pre-concepts or pre-constructs, we mean thoroughly identified simple ideas or an ‘atomic’ operation which in small steps cumulatively and gradually lead to deep understanding of the concept or the procedure connected with that concept

Experiencing order

Pupils start discovering the *order in things* by collecting the items from the scene either into a *box* (i.e. a container with no internal order) or in a *shelf* (i.e. a container with explicit order from left to right) – by *indirectly manipulating*²⁶ Emil. Later, when the level of control gets to higher levels, every command is recorded in the panel above the scene²⁷. Thus, pupils start perceiving and discussing the *order in steps* (of a process) as well. The other form of order present in many tasks is the inherent order of the items themselves, as with numbers or letters. Throughout the whole year 3 intervention we then cultivate the sense of order by recognizing, building, reading and discussing different patterns *in data* and *in programs*, i.e. in external records of steps or in external plans for future steps.

Coping with constraints

Pupils encounter several kinds of constraints, static and dynamic, implicit and explicit, such as (a) *constraints in driving Emil* – where to click, which command can be applied, how many clicks or commands can be run etc.; (b) *constraints in the stage* where objects serve as static or dynamic obstacles²⁸ (*collect all coins but nothing else...*) or later in the succession when one position cannot be clicked twice; (c) *constraints in the box* (*collect only ..., how many buttons will Emil get if he runs a given program* etc.); (d) *constraints in the shelf* where some positions have predefined content, or when the order of items to be collected is bound by a rule (*collect some 'good' words..., collect a repeating pattern of apples and pears...*); or (e) *constraints in the panel with program* when pupils are limited by a restricted number of steps (or piles of steps as illustrated in Figure 2).

Learning to control

In (Kalas, 2016) we referred to three levels of control in educational primary programming: *direct manipulation*, *direct control* and *computational control* (i.e. when the steps to be taken by an actor are all planned in advance), in a way related to increasing “distance” between an actor to be controlled and a pupil to control it. In our current design research, however, we have identified one more level – *indirect manipulation*, sitting between *direct manipulation* and *direct control* in the hierarchy. Pupils exploit this level of control when they cannot drag or click Emil but can click somewhere else in the scene to make him move there – if such move is possible.

Besides, we have also realized that the levels of control must be studied in relation with the way the record of these steps or the plan of the steps is represented, see Figure 3. This two-dimensional categorisation of control provides more complex instrument for designing and analysing tasks and progressions of pupils. We discuss these aspects of control in more detail in (Kalas, Blaho, Moravčík, 2018).

Learning to think with program, learning to think about program

Programs – first as **records** of the steps being taken, later also as **plans** for future steps (future behaviours) – become the objects to think with and think about in the course of Emil year 3 intervention. Pupils learn to read them, analyse and compare, modify, match and simplify, they learn to discuss and think about their properties and structure – in activities of Emil, in common discussions or in the unplugged tasks in the workbook. Our goal is to build pupils’ (and teachers’) perception of a program as an object which represents certain process and is an interesting object to work with, an instrument to explore a problem, experience and share an idea, or create a product.

Learning to create and abstract

In Emil the Artist, the last of the three environments in year 3 intervention, pupils create different visual patterns, repeating textile motifs or other regular geometric structures. It serves as an opportunity for them to build several elementary compositions, then abstract from the details and plan massive

²⁶ the reader will find the explanation of this category of control later in this paper

²⁷ see **a**, **b** and **c** in Figure 2

²⁸ Think of pupils interpreting (i.e. reading and envisaging) Emil’s move from mark 3 to 4 in Figure 1: will the button in the middle column be still there?

repetitions and combinations of those bits. We consider this to be a workshop for preliminary steps that will later lead to the construct of user defined procedures and infinite and conditional iteration.

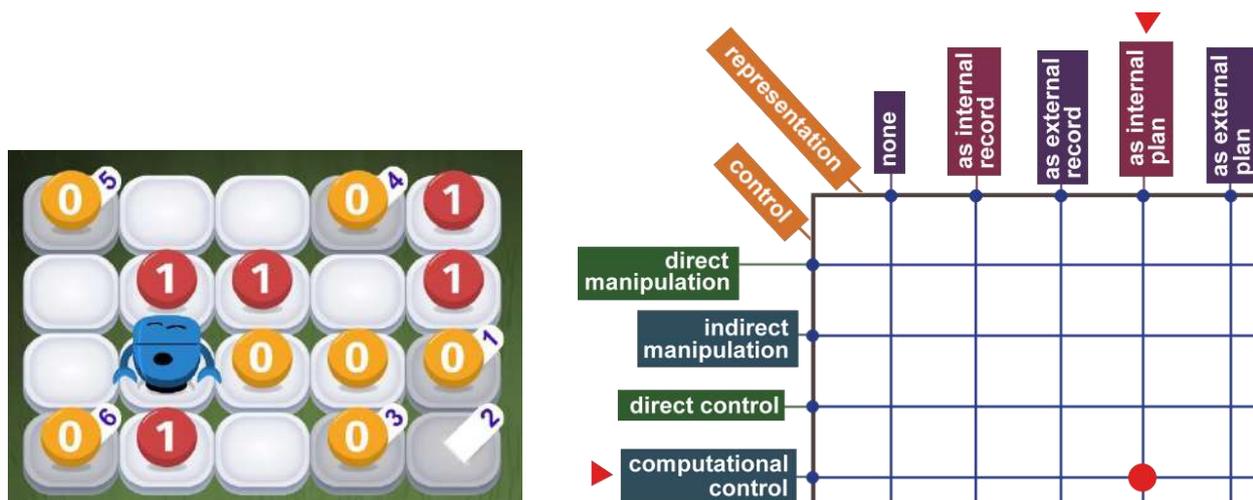


Figure 3. The task on the left: For sleeping Emil plan a journey to collect all 0's (and no 1's) in no more than six clicks. Then wake Emil up to run your plan. (In the picture the solution – a plan – is already shown as well.) When this task is analysed within our two-dimensional categorisation of control we arrive at computational control where the program is represented as an internal plan (i.e. inside the scene, not externally as a sequence of explicit commands).

5 Discussion and conclusion

Increased interest in primary programming creates enormous need for better understanding primary programming as a complex phenomenon comprised of properly selected educational objectives, *computational constructs* (or sometimes *pre-constructs*) and *practices*, programming environment(s), and systematic learning content with properly developed pedagogies – suitably delivered to primary teachers. All of these must be studied in parallel as they are interconnected in multifold ways.

As we decided in our project to contribute to deeper understanding of the complexity of elementary *computational constructs* at the level of lower primary, we need to be in possession of an adequate programming tool. As we failed to identify such tool we set out for designing and developing one. This demanding but productive constellation is giving us unique opportunity to get deeper in studying the *computational constructs* by designing and developing new programming environments and, in parallel, design and implement appropriate programming environments by studying the learning processes of pupils when building their understanding of the elementary *computational constructs*, exemplary circumstances for adopting design research strategy. We believe that this situation provides us also with valuable opportunity to design modern *computing programme of study* for lower primary education and thus support systematic and sustainable learning processes in primary programming²⁹ – something rather new and much needed in the primary school.

Our design and development is continuously informed by frequent small assessment probes composed of a selection of tasks. These are the tasks included in the learning gradation, but also some additional assessment tasks we prepare for teachers and pupils. The assessment task of Figure 1, for example, contains a program represented as an *internal plan of moves* (i.e. moves indicated inside the scene). Pupils were asked (working individually this time) to identify the box of buttons and coins which will result from Emil running given program. In the class of 21, four pupils wrongly marked the third alternative, probably failing to notice that the button in the second row, third column will have already been collected when Emil later moves from mark 3 to 4. All other 17 pupils correctly identified the first alternative as the correct result. Such small probes – regularly inserted into our design and assessment

²⁹ inspired e.g. by complexity and consistency of the mathematics programmes of study

iterations – help us observe and understand the progression of each pupil in the group and about the developmental appropriateness of the units of tasks as well.

Naturally, this is the point where deeper research must start. We believe that programming environments which are already resulting from the project, will provide interesting instrument for a follow-up systematic research of the complexity of primary programming.

Acknowledgements

First, I would like to thank other core members of our Emil project team, Andrej Blaho and Milan Moravčík. I am also grateful to Indicia, npo. for funding this work. Finally, I want to thank the pupils and teachers from our Slovak design schools for their dedication, hard work, enthusiasm for programming with Emil and continued engagement with our team throughout the project.

References

- Ackermann, E. K. (2012) *Programming for the natives: What is it? What's in it for the kids?* Child Research Net, Japan
- Benton, L., Hoyles, C., Noss, R., Kalas, I. (2016) *Building Mathematical Knowledge with programming: Insights from the ScratchMaths project*. In: Proceedings of the Constructionism in Action: Constructionism, 2016. Suksapattana Foundation, Bangkok, pp. 26-33
- Benton, L., Hoyles, C., Kalas, I., Noss, R. (2017) Bridging primary programming and mathematics: Some Findings of Design Research in England. *Digital Experiences in Mathematics Education*, August 2017, Vol 3(2), doi: 10.1007/s40751-017-0028-x, pp. 115-138
- Benton, L., Saunders, P., Kalas, I., Hoyles, C., Noss, R. (2018a) Designing for learning mathematics through programming: A case study of pupils engaging with place value. *Int. J. of Child-Computer Interaction*. Vol 16, June 2018, doi: 10.1016/j.ijcci.2017.12.004, pp. 68-76
- Benton, L., Kalas, I., Saunders, P., Hoyles, C., Noss, R. (2018b) Beyond Jam Sandwiches and Cups of Tea: An Exploration of Primary Pupils' Algorithm-Evaluation Strategies. *Journal of Computer Assisted Learning*. doi: 10.1111/jcal.12266.
- Blackwell, A. F. (2002) *What is Programming?* In 14th Workshop of the Psychology of Programming Interest Group, pp. 204-218
- Clements, D. H. (2002) Computers in early childhood mathematics. *Contemporary issues in early childhood*, 3, 2 (2002), pp. 160-181
- Futschek, G., Moschitz, J. (2011) *Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming*. In: Kalaš I., Mittermeir R.T. (eds) *Informatics in Schools. Contributing to 21st Century Education*. ISSEP 2011. Lecture Notes in Computer Science, vol 7013. Springer, Berlin, Heidelberg, pp. 155-164.
- Grizioti, M., Kynigos, Ch. (2018) Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. In: Proceedings of the Constructionism 2018. Vilnius
- Harel, I., Papert, S. (1990) *Software design as a learning environment*. *Interactive Learning Environments*, 1, pp. 132
- Kalaš, I. (2016) *On the road to sustainable primary programming*. In: Proceedings of the Constructionism in Action: Constructionism, 2016. Suksapattana Foundation, Bangkok, pp. 184-191
- Kalaš I., Benton L. (2017) *Defining Procedures in Early Computing Education*. In: Tatnall A., Webb M. (eds) *Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing*. WCCE 2017. IFIP Advances in Information and Communication Technology, vol 515. Springer, Cham. doi: 10.1007/978-3-319-74310-3_57
- Kalaš, I., Blaho, A., Moravčík, M. (2018) *Exploring Control in Early Computing Education*. Submitted to ISSEP 2018, Saint-Petersburg

Lewis, C. M., Shah, N. (2012) *Building upon and enriching grade four mathematics standards with programming curriculum*. In: Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM, Raleigh, North Carolina

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010) The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* 10(4)(2010), pp. 137-146

Meerbaum-Salant, O., Armoni, M., Ben-Ari, M. M. (2013) *Learning computer science concepts with Scratch*. *Computer Science Education*, Vol 23 (3), pp. 239-264

Moravčík, M., Kalaš, I. (2012) *Developing software for early childhood education*. In Addressing educational challenges: the role of ICT. IFIP Working Conference, Manchester, MMU, 12 p. [CD-ROM]

Morgado, L., Cruz, M., Kahn, K. (2006) *Radia Perlman – A pioneer of young children computer programming*. *Current Developments in Technology-Assisted Education*. Formatex, pp.1903-1908.

Nieveen, N., Folmer, E. (2013) *Formative Evaluation in Educational Design Research*. In: Plomp, T., Nieveen, N.: *Educational design research*. SLO – Netherlands institute for curriculum development: pp. 152-169

Noss, R., Hoyles, C. (1996) *Windows on Mathematical Meanings: Learning Cultures and Computers*. Kluwer Academic Publishers, 275 p

Pea, R.D. et al. (1985). *Logo and the Development of Thinking Skills*. In M. Chen and W. Paisley (Eds.) *Children and Microcomputers: Research on the Newest Medium*. Sage, pp. 193-212.

Papert, S. (1980). *Mindstorms. Children, Computers, and Powerful Ideas*. Basic Books, New York, 230 p

Perlman, R. (1976). *Using Computer Technology To Provide A Creative Learning Environment For Preschool Children*. AI Memo 360, MIT, 32 p

Touloupaki, S., Baron, G.-L., Komis, V. (2018). *Un apprentissage de la programmation des l'école primaire: le concept de message sur ScratchJr*. In Parriaux, G., Pellet, J.-P., Baron, G.-L., Komis, V. (eds.): *De 0 a 1 ou l'informatique a l'école*. Actes du colloque Didapro 7 – DidaSTIC, Lausanne 2018. Peter Lang, Bern, pp. 303-323

Webb, M. E., Bell, T., Davis, N., Katz, Y. J., Fluck, A., Sysło, M. M., Kalaš, I., Cox, M., Angeli, C., Malyn-Smith, J., Brinda, T., Micheuz, P., Brodnik, A. (2018). *Tensions in specifying computing curricula for K-12: Towards a principled approach for objectives*. *IT – Information Technology*, Vol 60 (2), pp. 59-68, doi: 10.1515/itit-2017-0017

Wilson, A., Hainey, T., Connolly, T.M. (2013) Using Scratch with Primary School Children: An Evaluation of Games Constructed to Gauge Understanding of Programming Concepts. *Int J of Game-Based Learning*, Vol 3(1), doi: 10.4018/ijgbl.2013010107

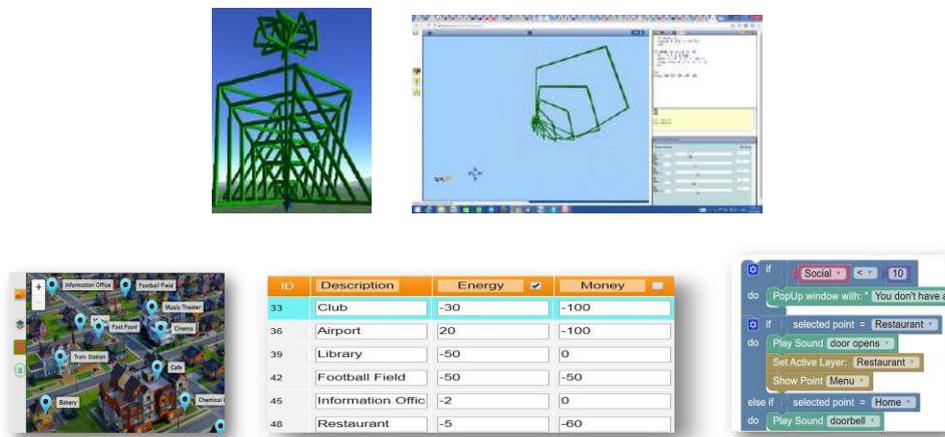
In Support of Integrated Approaches to Constructionist Designs and Interventions: The Case of ChoiCo and MaLT

Chronis Kynigos, Kynigos@ppp.uoa.gr

National and Kapodistrian University of Athens, Educational Technology Lab School of Philosophy, P.P.P. Dept, Greece

Abstract

Constructionism is now a 50 year-old theory of learning, a theory of educational design and a framework for pedagogical action. In this time, society, educational challenges and the abundance of digital media have brought a diversity of frames, focal points, viewpoints and interventions. In many places silo constructionist perspectives are seen as obsolete or at best ultimate frames for meaning-making through individual and social bricolage. In my talk I will argue for perspectives integrating constructionism in a wider landscape of educational paradigms, theories, affordances and intervention strategies. I will do this by showing what students and teachers have built with two web-based constructionist expressive media, one very different to the other, MaLT – turtle sphere and 'Choices with consequences' games (ChoiCo). In my context these are proving to be powerful means for a proximal approach to infusing constructionist perspectives in wide scale initiatives.



MALT2 - programming, 3D, Dynamic Manipulation, <http://etl.ppp.uoa.gr/malt2>. ChoiCo: GIS, Programming, BD, <http://etl.ppp.uoa.gr/choico>

MALT is a Logo programmable 3D simulator including Turtle Graphics and most importantly dynamic manipulation of variable values with DGS - like effects on the graphical output from variable procedures. ChoiCo is a tool for game modding, supporting a socio-scientific paradigm of diverse consequences games involving complex issues like dietary choices or environmental issues and embedding powerful ideas in various ways. The examples will show how avenues for powerful ideas uniquely made available and embedded in multifaceted issues present exciting challenges for design.

Keywords

Integrations; affordances; educational paradigms; programming

Bones, Gears and Witchcraft

Jens Mönig, *jens@moenig.org*
Principle Investigator SAP, Germany

Abstract

Who doesn't like a good story? I never get tired playing with stories kids animate with Scratch or Snap! To me, storytelling is at the heart of the current digital literacy movement. But it's not just about kids. I've been surprised by the culture of storytelling nurtured in big industrial companies. Designing the right story can be crucial for a project, a program or a promotion. The opposite is also true: The wrong story has the ability to compromise funding and even thwart a career. And then there are stories that convince for all the wrong reasons, and success that feels like defeat.

I will share a few of my stories for children, corporate management and government officials. Among them, how Katharina Kepler's witchcraft trial has been a turning point for computing, and how machine learning can be used to illustrate a business proposal, before examining a particular terrible instance of constructionism gone wrong in the German state of Baden-Württemberg. Expecting this to spark some controversy I will close by opening up a discussion with the audience about favorite stories, inviting examples of "good" and "bad" specimens..

Keywords

Scratch; Snap!; Idigital literacy

Constructionist Experiences for Mathematics Across Educational Levels

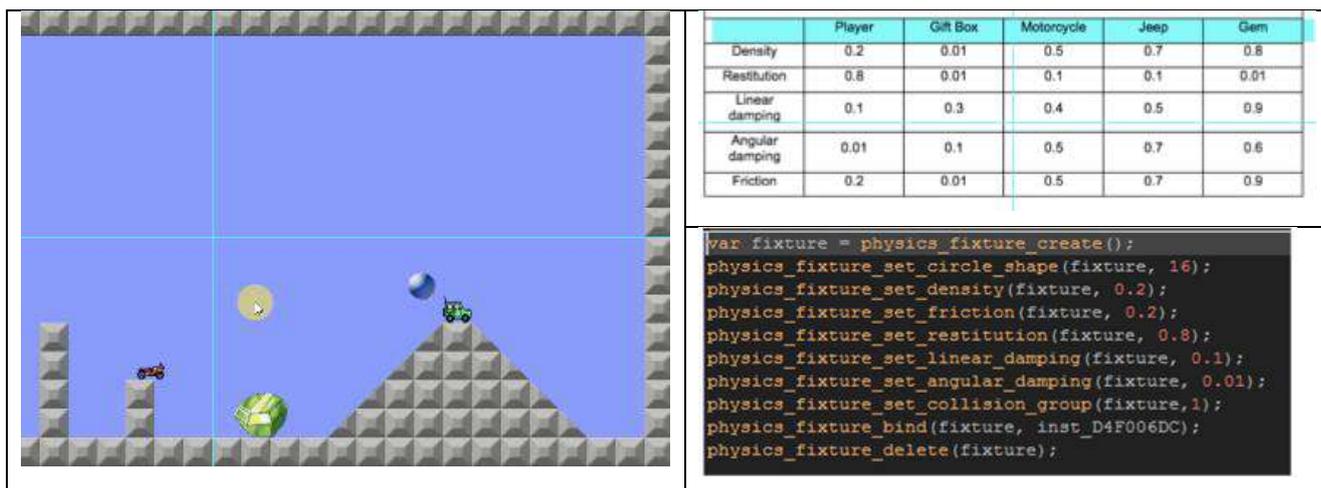
Ana Isabel Sacristán, asacrist@cinvestav.mx

Centre for Research and Advanced Studies, Cinvestav, Mexico

Abstract

Constructionism, in the research literature, is seen predominately in contexts where it is used as a paradigm for promoting learning at K-12 levels. However, it can apply at all levels, as well as connect levels. In this paper I deal with the issue of classroom implementations for mathematical learning of constructionist experiences at different educational levels, but with particular focus on higher education implementations. I begin by revisiting some of the fundamental principles of constructionism. Then, I look briefly at how constructionism can provide early access to powerful ideas; that is, where younger learners can access mathematical ideas perceived as more advanced. For that, I mention my work in the design of two different microworlds that enabled young students to explore and engage with mathematical infinity-related ideas.

I then share some of my experiences in several attempts of constructionist implementations and microworlds for mathematical learning at university level. I present two examples that involved building and exploring computer models and simulations of real phenomena: the first in a distance-learning course conducted as a virtual mathematics laboratory; the second, of videogame construction by engineering students (see figure below). A third example is of computer programming R-based tasks for the learning of statistics in environmental science students. I finish by presenting a fourth example from a university in Canada where mathematics university students are required to program digital mathematical exploratory objects or microworlds. In all the examples presented, students engage in constructing models or programs, and in doing so, engage in doing mathematics.



A student's videogame (left), with his table of physical characteristics of the objects involved (top-right) that he would need to program into the game engine (bottom-right).

Keywords

Constructionism; computer programming and expressive media; collaborative learning; modelling; higher education.

Introduction

Since the 1980s, there have been many projects attempting to implement the constructionist paradigm in schools, in order to enhance mathematical learning. Most of the first ones were based on Logo

programming, and were at primary or middle-school levels. There have been, however, fewer constructionist implementations for mathematical learning at university level reported in the literature, relative to those reported at the K-12 levels, despite: several excellent proposals for college of quite advanced mathematical discovery ideas with Logo –e.g. as in Abelson & diSessa's (1986), *Turtle Geometry*–; some of the work of members of the Center for Connected Learning and Computer-Based Modeling (<http://ccl.northwestern.edu>), particularly related to concepts such as proof or probability (e.g. Wilensky, 1993 & 1995) or for STEM learning using NetLogo; and some advocates for rethinking university education through technology-enhanced learning and collaboration (e.g. Laurillard, 2002). Why the lack of constructionist implementations in higher education, is beyond the scope of this paper, although it is clear that implementing constructionist exploratory learning environments in school cultures has been problematic and complex, as has been discussed elsewhere (e.g. by Laurillard, 2002).

In this paper, I would like to present some examples of constructionist experiences and implementations in which I have been involved. The first examples are meant to illustrate how constructionism can help bridge educational levels, by giving younger learners access to powerful and advanced mathematical ideas. In the second part, I focus on examples of constructionist implementations for mathematical learning in higher education.

But first, I would like to begin by revisiting some of the main ideas of constructionism.

Some fundamental ideas of constructionism, and the value of computer-based expressive activities

The fundamental premise of the constructionist paradigm, as stated by Papert and Harel (1991), is that it shares “constructivism's connotation of learning as ‘building knowledge structures’ [...] then [adds] the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity” (p. 1), that is, something shareable.

In Papert's (1980) vision, one particularly valuable means of achieving the above is in programming the computer because, in doing that, the student “establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building” (p. 5); and “in teaching the computer how to think, [students] embark on an exploration about how they themselves think” (p.19). Computer programming also involves debugging and, as Papert (1980) explained, errors are of benefit because they lead to the need to understand what went wrong, and through that understanding, to fix them.

Papert's theories of computer programming as a way for learners to engage in “Mathland”, was confirmed in a relatively recent study by Rich, Bly and Leatham (2014) which showed that programming and solving programming problems, can: provide a context for many abstract concepts; illustrate the distinction between understanding the application of mathematics in a specific situation, and the execution of a procedure; help divide complex problems into more manageable tasks; provide motivation and eliminate apprehension; and give context, application, structure and motivation for the study of mathematics. In fact, programming can be an engaging problem-solving activity where students can explore mathematics in different representations and generate and articulate mathematical relationships. Nevertheless, as has been learned, these programming activities need to be carefully designed and structured within a learning environment.

In terms of learning environments, Papert (1980) first proposed that with the versatility of computers, one could create microworlds: “self-contained world[s], [...] each with its own set of assumptions and constraints”, where learners “get to know what it is like to explore the properties of a chosen microworld undisturbed by extraneous questions [...] and] learn to transfer habits of exploration from their personal lives to the formal domain of scientific theory construction” (p.117). He called these incubators for knowledge and powerful ideas; “places” where these ideas (including certain kinds of mathematical thinking) can easily hatch and grow. Moreover, Papert (1980) placed emphasis on the entire learning culture (a different kind of culture) conceiving educators as support for learners to build their own intellectual structures; creating conditions for construction and invention (rather than providing ready-made knowledge), giving students objects-to-think and materials drawn from the surrounding culture,

including “emotionally supportive working conditions [that] encourage them to keep going despite mathematical reticence” (p. 197), with students having creative and personally defined end-products that they can genuinely be excited about. Hoyles and Noss (1987) considered that microworlds had to be designed taking into account the characteristics of the specific learners; having a careful pedagogical design with materials (e.g. worksheets) and appropriate teacher interventions; and a social environment fostering collaboration and where products can be shared and discussed in small and whole groups.

A way to summarise a constructionist learning environment (where the italicised words describe most of the main elements of constructionism) could be as: *student-centred learning situations* where students *consciously engage* in *constructing* (e.g., program) *shareable, tangible* objects, through *creative personally meaningful projects* (e.g. computer-based) where they have *objects-to-think-with*, access/develop *powerful ideas*, and have opportunities for *explorations*, and *thinking about their own thinking* (analysing) through *debugging* (fixing).

I consider that constructionist implementations ideally should have the following characteristics:

- There has to be a medium (e.g. digital tools) for an exploratory and expressive activity (such as computer programming/coding, or building/describing models or structures using an expressive medium or software).
- Students need to be actively involved and mostly in charge of their constructions and explorations.
- Activities should take place within a structured microworld and collaborative learning environment with the characteristics described above.

In the following sections, I present examples of projects in which I have been involved, that have fulfilled those characteristics.

Constructionism across educational levels: Early access to powerful ideas

Although my main focus, in this paper, is on tertiary educational level constructionist experiences, in the title of the paper I refer to experiences *across educational levels* because an important aspect of the constructionist philosophy is that the experiences and learning can transcend pre-established (institutional) and assumed educational hierarchies and categories, by enabling learners to access and/or develop powerful ideas. So I would like, therefore, to delve briefly into the idea of how learners can have early access to powerful ideas.

Connectedness and webbing for enhancing mathematical meaning-making

There are many examples of how, through constructionist approaches, advanced mathematical and scientific ideas can be made accessible to younger learners. As described above, Papert talked in terms of microworlds as incubators for powerful ideas. For example, many of the mathematical ideas presented in Abelson and diSessa’s (1986) *Turtle Geometry* book, although some are very advanced, are powerful ideas that become more accessible through their Logo-based explorations.

Wilensky (1993) talked of “connected mathematics” where mathematical meanings are enriched by connecting ideas and representations; thus mathematical knowledge becomes more concrete (i.e. accessible) through the built relationships. His ideas were precursors to what Noss and Hoyles (1996) later called “webbing” where, from webs of familiar connections, further connections are built outwards along lines of one’s own interests, with “the presence of a structure that learners can draw upon and reconstruct for support – in ways that they choose as appropriate for their struggle to construct meaning for some mathematics” (p.108). It is through that scaffolding of microworlds or constructionist digital environments, that learners can more readily build connections and access powerful and advanced ideas. It is worth noting that Wilensky later expanded the “connected mathematics” to “connected learning”, in order to include all disciplines and, as I understand it, more forms of connectedness, such as connecting people in collaborative learning and through collaborative and networking technologies.

In my past work, I have been involved in several projects where constructionism was used for early access to powerful ideas. Some were related to trigonometry and called the Painless Trigonometry

Projects, where students explored and used trigonometry in personally meaningful year-long school projects, in grades below those where most of the trigonometric ideas they used are first formally introduced (see Sacristán & Jiménez-Molotla (2012). Others projects, which I present next, related to mathematical infinity.

Programming-based microworlds for creating meanings of the infinite

The examples that I present here involve a couple of microworlds designed to make mathematical infinity-related ideas accessible to younger students. The first example is a Logo-based microworld (see Sacristán & Noss, 2008), that enabled students to explore and develop concepts of infinite processes and objects. In this microworld, students of different ages (some as young as 14), by programming in Logo, constructed and investigated graphical models of infinite sequences of the type $\{1/k^n\}$ (such as spirals and bar-graphs), as well as fractal figures (such as the Koch curve), conceived as "limit-objects" of infinite graphical sequences. Students gave meaning to the processes under study by coordinating the visual and numeric outputs with the symbolic code contained in the procedures that they themselves had written. In this way, the microworld supported students in the coordination of hitherto unconnected or conflicting intuitions concerning infinity, based on a constructive articulation of the different representations.

Another example was the design and implementation of computer programming activities using the innovative ToonTalk infrastructure (<http://www.toontalk.com>), aimed at introducing young students (9–13 years old) to the idea of infinity, and in particular, to the cardinality of infinite sets (see Kahn, Sendova, Sacristán & Noss, 2011). Via carefully designed computational explorations within an appropriately constructed medium, students in several European countries built computational models and programs (for example, through programming robots in the ToonTalk animated world – see 0), which they shared and discussed through distance communication (through web reports): students in each country challenged the constructions of the others, thus fostering deep mathematical inquiry and introducing the real spirit of mathematics to school classrooms. In this way infinity was approached in a learnable way without sacrificing the rigour necessary for mathematical understanding of the concept.



Figure 1. Part of a student's web report showing ToonTalk robots generating the set of integer numbers

Constructionist implementations for mathematics in higher education

In terms of higher education, I have been involved in several constructionist projects where university students engage in computer programming and/or expressive activities for mathematical exploration or learning. I present some of these below. These follow the constructionist principles outlined in the first part of this paper, including providing students with objects-to-think and materials from the real world or surrounding culture; as well as having students building shareable objects, discussing and collaborating. In that sense, I follow Laurillard (2002) who advocates for constructionist and collaborative technology-based learning environments in higher education and considers that "the aim of university teaching is to make student learning possible [...] not simply impart decontextualised knowledge, but must emulate the success of everyday learning by situating knowledge in real-world activity" (p. 42) helping students reflect on their experience of the world and ways of representing it. Thus, in all the projects presented below, students were or are involved with topics and data related to real-life phenomena and that can be meaningful for their area of study.

A distance-learning virtual mathematics laboratory

A first project (see Olivera, Sacristán & Pretelín-Ricárdez, 2013) involved a distance-learning environment (a virtual laboratory) where university students (mostly adults pursuing continuing education) engaged in exploratory modelling activities of various types of real-life mathematical problems (e.g. related to linear motion; gravity and free-fall; population growth; cryptography; and involving statistical analysis). This project used Lesh's *et al.* (2000) idea of model-eliciting activities (MEAs), where tasks centre on building models, cycling through models and sharing these. MEAs share some of the conceptions of constructionism in that "the products that students produce [...] involve sharable, manipulatable, modifiable, and reusable conceptual tools (e.g., models) for constructing, describing, explaining, manipulating, predicting, or controlling mathematically significant systems" (Lesh and Doerr, 2003, p. 3).

In our virtual laboratory, the tasks required exploring the proposed situations; building models through collaboration; sharing, discussing and reflecting; and proposing new explorations. Some of the most interesting activities involved the analysis of videos (which led to extensive discussions on determining the scale of the videos), developing models to reproduce the behaviours and phenomena shown on the video, and analysing and discussing which proposed models best fit the real data (see 0 further below).

The explorations, modelling and collaborations were carried out through various digital means and interactive tools for learning through exploration (following the definition of a virtual laboratory proposed by Jeschke, Richter & Seiler, 2005). The various digital tools included different materials (e.g. real-life videos) and complementary expressive software, such as: frame-by-frame video analysis software; a virtual ruler for on-screen measurements (JR Screen Ruler, <http://www.spadixbd.com/freetools/jruler.htm>); tools for finding mathematical equations to fit the data, such as spreadsheets or CurveExpert (<http://www.curveexpert.net>); and modelling software (Modellus, <http://modellus.pt>) for building mathematical models and comparing them to the real data. Since students were at a distance, they needed to collaborate and share their conjectures and findings online; for this, we mainly used a web-based discussion forum. The online exchanges constituted additional means for learning, since they forced students to express their ideas, constructions and conjectures as clearly as possible to others in written form (aided by screen captures and even photos of their handwritten work), thus helping them clarify their own understandings (while at the same time acting as windows, for teachers and researchers, into their meaning-making, as described by Noss & Hoyles, 1996).

In most activities, as explained above, students collectively worked on a problem (or part of it), and later proposed new problems. For example, one activity involved analysing and modelling free-falling dropping objects on Earth; after the initial explorations, some students proposed analysing the gravity on the Moon by analysing a NASA video of an astronaut jumping on the Moon (0). Because that video is not of a free-falling object, it generated discussions on what kind of movement it is (with students concluding it is a type of parabolic shoot with a nearly vertical angle). They then proposed mathematical equations that they implemented in Modellus and compared the resulting model to their real data (0). The construction of models and simulations helped students identify and discern the important mathematical elements in the situation under study and that help model it. Furthermore, as described above, the construction of meanings was also supported by the social structure created by the online community.



Figure 2. A student proposes, in the online forum, to analyse the gravity on the Moon through the video of an astronaut jumping on the Moon.

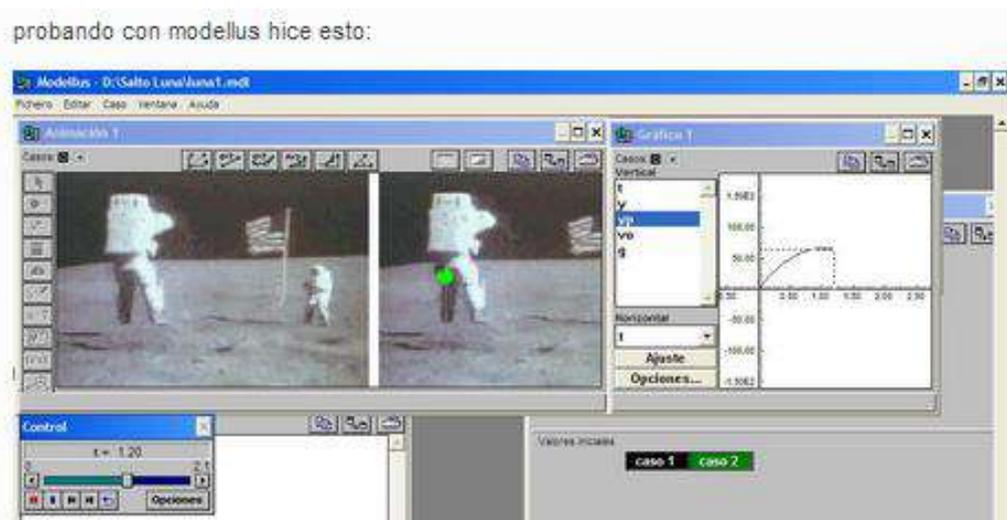


Figure 3. The student shares her comparison of the real data video (left), with the model she constructed using Modellus (right), where the astronaut is represented by the green dot (centre).

Videogame construction by engineering students

Another higher educational constructionist project in which I was involved in recent years, was one where senior university engineering students created videogames (see <https://tesis-apretelinr.blogspot.mx/>) which required using and adapting physico-mathematical models. This was a project inspired by the work of Kafai (1995). The programming of videogames is a motivating activity that engages students in producing working models of certain real-life behaviours but in a context that is meaningful to them. Some of the videogame topics involved: physical phenomena (e.g. modelling and simulating water behaviour; see Pretelín-Ricárdez & Sacristán, 2015; or bouncing objects, e.g. balls, such as in basketball or tennis games – see 0 and 0); virtual robots navigating mazes (which required using and designing digital systems, i.e. combinational logic circuits; see Sacristán & Pretelín-Ricárdez, 2017); or simulated mechanical systems (e.g. robotic arms). One of the aims of this project was for students to develop know-how for their future profession as engineers on how to apply mathematical knowledge and modelling.

The videogame constructions (using GameMaker Studio – <http://www.yoyogames.com>) were structured through sequences of model-building tasks that involved several stages combining or alternating paper-and-pencil work; individual and/or collaborative programming work; and whole class discussions. As in the project described above, this project used Lesh et al.'s (2000) Model-Eliciting Activities (MEAs) theory; in particular, the design of the model-building tasks took into consideration the six principles of MEAs: reality, model construction, model documentation, self evaluation, model generalisation, and simple prototype.

Having to build models of phenomena and then program these models into the GameMaker videogame engine, helped students gain a deep understanding of all the elements involved in each model (e.g. see the Figure in the abstract; and 0 and 0, which show some of the elements that needed to be programmed into each videogame depicted in the figures). For example, for the videogames involving water-behaviour (Pretelín-Ricárdez & Sacristán, 2015), students needed to produce first a mathematical model for that behaviour: they usually came up with complex models of fluid mechanics, addressing the water model either as a molecular model or as a continuous model. They then realised that these models could not be programmed as such into the videogame engine, so they were forced to analyse and discern the most important elements present, in order to produce, and program, simplified models into the videogame engine. They did this through collaboration and discussion.

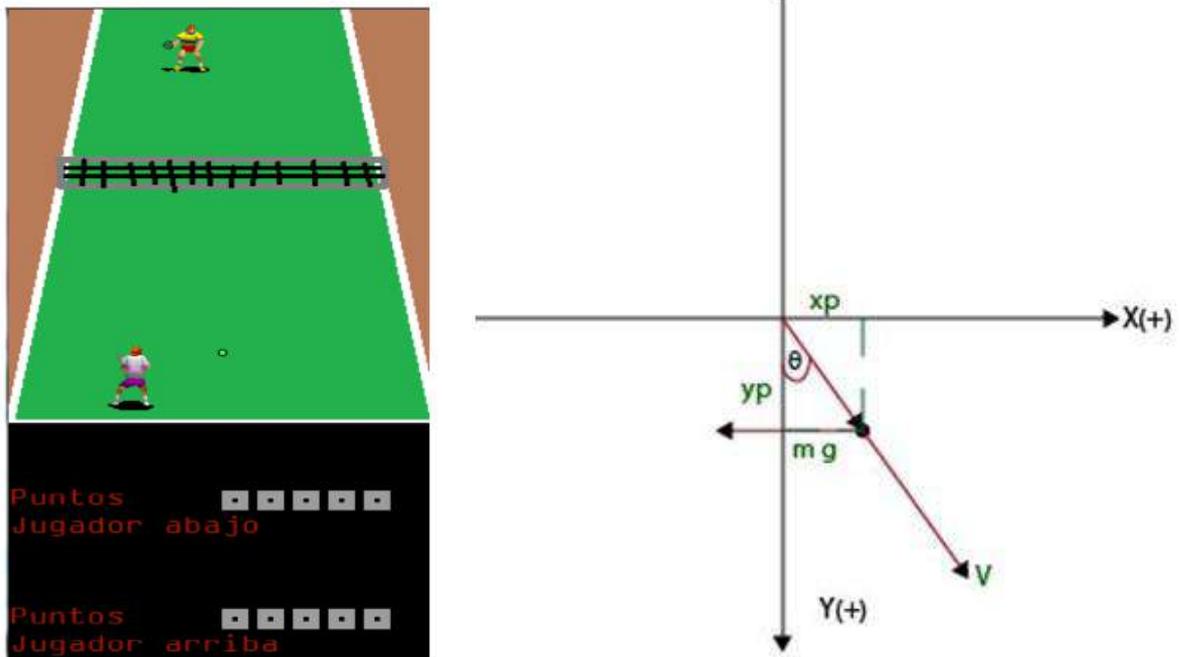


Figure 4. A couple of students' Tennis videogame (left). In order to program the ball's movement (see 0) the students' analysed, and mathematically depicted its behaviour, such as its up-and-down movement when hit from the right, through a diagram (right).

```
//ecuación del movimiento parabólico de arriba hacia abajo lado derecho
if (derecha == 1)
{
    //x_final    = 150;
    tf          = 0.5;
    t           += 1/100;
    t2          = t*t;
    argumento   = (x_final + var_grav*(sqr(tf)/2) - x_inicial)/(y_final - y_inicial);
    argumento2  = degtorad(argumento);
    var_alpha   = arctan(argumento);
    var_dos     = (1/2)*var_grav*t2;
    var_v0      = (y_final - y_inicial)/(tf*cos(var_alpha));
    x           = x_inicial + (var_v0*sin(var_alpha)*t) - var_dos;
    y           = y_inicial + (var_v0*cos(var_alpha)*t);
}
else derecha = 0;
}
else instance_destroy();
```

Figure 5. Script of the tennis ball's up-and-down movement when hit from the right.

As recognised by the students, the videogame construction activities provided an opportunity to apply their theoretical knowledge in real-life projects and for experiencing how such real-life projects could be carried out. In this way, students gained insights and expertise on how to apply their knowledge in different realistic projects of contexts related to their engineering profession.

Another important result, is that students appropriated the proposed videogame construction tasks as their own, turning them into meaningful personal projects, which motivated them highly. This was evidenced by the deep commitment that they manifested in the way they integrated the physics and mathematical models with game-playing mechanics and aesthetic aspects (such *sprites*, backgrounds and sounds), thus turning the games unto attractive ones (e.g. 0).

R-based tasks for the learning of statistics by environmental sciences students

In another project (see Mascaró, Sacristán & Rufino, 2014, 2016), directly inspired by Logo programming microworlds, we have designed sequences of constructionist and collaborative, computer-programming tasks in the R programming language (see the R Project for Statistical Computing – <http://www.r-project.org>) for the learning of probability, statistics and experimental analysis concepts. These tasks are the core content of courses for college and graduate environmental sciences and biology students –who tend to have strong aversions to mathematics and statistics. The aim is for students to develop statistical reasoning, rather than applying blindly statistical tests; build statistical models for research; apply and understand statistical computing software (in this case, R) to carry out calculations in experiments; and learn how to interpret the results given by the software. All the tasks are presented through R-code “worksheets” with instructions, guidelines, examples (using data adapted from real research situations), programming tasks, questions for reflection and comments (see 0).

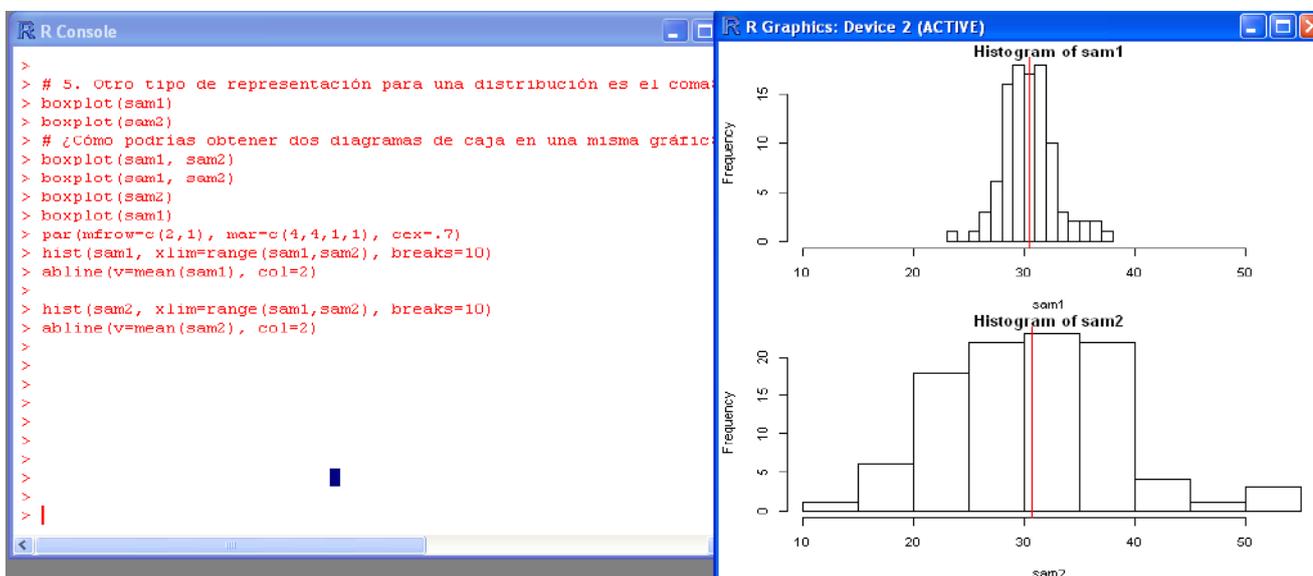


Figure 6. Part of an R-based task, including on the left-hand side some worksheet questions, with typed commands for generating graphs (i.e. the histograms presented at the right).

The understanding of statistical models is facilitated by creating objects in R, to represent them (e.g. graphs, lists of data, statistical values, etc.). Tasks are carried out through collaborative work leading to reflective interactions, explanations and evaluations. By typing R-commands, students draw and interpret graphs (i.e. visualise models) relating numerical data to graphical representations, as well as to mathematical formulae. They need to predict what a change in the programming code would produce. In this way, students go back and forth in the analysis of the data, and suggest changes for obtaining different representations. In this way, they can develop deeper understandings of how statistical functions and graphical representations can help create richer meanings for the data.

Over the course of eight years of design research and iterative processes, we have refined the R-based tasks, and implemented these in over a dozen courses at university and post-graduate levels, More

recently (Mascaró & Sacristán, 2018, in these conference proceedings), we have also developed constructionist-compatible ways to assess students' learning, in order to grade them for the university courses in which they are enrolled. Through both course observations and assessments, we have obtained encouraging results, particularly in the affective dimension (see Mascaró, Sacristán & Rufino, 2016): many students lose their fear of statistics, with most of them actively engaging in the activities; furthermore, several students have appropriated themselves of the software (e.g. building their own R scripts) for their own research with an apparent clearer understanding of statistical concepts.

Brock University's MICA program

A final example of a higher education constructionist implementations that is worth mentioning, is the MICA (Mathematics Integrated with Computers and Applications) program at Brock University in Canada (Muller, Buteau, & Sacristán, 2015; Buteau, Sacristán & Muller, 2018). That program was promoted by Eric Muller, who was a participant and author of the very first ICMI study held in 1985, researching the influence of computers and informatics on mathematics and its teaching (see Cornu & Ralston, 1992). The MICA program stands out as a complete curricular implementation that has been functioning since 2001 –rather than being just a limited-scope project– and that integrates computer programming activities in the pure and applied mathematics syllabi: In the MICA program, first- and second-year university students, in addition to traditional mathematics courses, have non-traditional courses where they engage in programming their own interactive mathematical digital environments or microworlds, also called Exploratory Objects (or EOs). (The roles of the MICA instructors are also non-traditional, as well as demanding, as discussed in Buteau, Sacristán and Muller, 2018; in these conference proceedings.) Through the construction of their EOs, students explore their own stated mathematical conjectures, theorems, or real-world situations, and need to engage in a combination of modelling, simulation, optimization and visualization of the mathematical ideas they explore and program. As such, they learn mathematics by *doing* and creating mathematics. The premise of the program is that students' abilities and potential to do mathematics are both enhanced as they program their own interactive computer environments (Muller, Buteau, & Sacristán, 2015). Although I was not involved in its design nor implementation, I have been fortunate to collaborate for the past 4 years with Muller and some of his colleagues at Brock University in researching the impact of that program on students' learning and appropriation of computational thinking and programming.

Concluding remarks

In this paper I have summarised what I consider the main characteristics of constructionism. I then presented examples of either microworlds that helped bridge educational levels by giving young learners access to advanced mathematics; or examples of an area that is scarce in the research literature: that of constructionist implementations for mathematical learning, in higher education.

Some of the higher education projects were experimental, while others are programs implemented in real courses: The virtual laboratory, although implemented in experimental courses during several years, could not be continued due to the differences with the established curricula (confirming the complexity of implementing constructionist approaches in institutional settings). It is an example of how implementing such projects in a sustained and non-experimental way, is challenging. In contrast, the videogame project was so successful that it is now being considered as a regular course in the engineering program where it was first given as an experimental course. The statistics program has been integrated into regular courses for several years, with much success, even though, in those courses, students are initially more guided than in other constructionist implementations. And the MICA program has been held and implemented successfully for over 15 years.

In any case, the latter examples illustrate the possibilities for integrating constructionism in higher education. All of those projects meet the constructionism characteristics that I outlined at the beginning of the paper: at the core of each project are tasks that give students a central active role for exploration and construction, where they have to engage in some type of expressive activity (programming and/or modelling) using technology; and most also involve collaborative work and group discussions, where products are shared and analysed. I consider the latter social aspects to be fundamental for reflecting on the knowledge put into practice, and generating more stable meanings for that knowledge.

Acknowledgements

I would like to thank and acknowledge the work of all my co-authors in the projects described in this paper. I also acknowledge the financing from DGAPA-UNAM with the PAPIIME PE204614 and PE207416 grants for developing the tasks in the statistics project. I note, also, that parts of this paper were previously published in Sacristán (2017).

References

- Abelson, H., & diSessa, A. (1986). *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. Cambridge, MA: MIT Press.
- Buteau, Sacristán & Muller (2018). Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics. In *Proceedings Constructionism 2018*, Vilnius, Lithuania.
- Cornu, B., & Ralston, A. (1992). *The Influence of Computers and Informatics on Mathematics and Its Teaching* (2nd ed.). Science and Technology Education Series, 44. Paris: UNESCO. Retrieved from <https://eric.ed.gov/?id=ED359073>
- Hoyles, C., & Noss, R. (1987). Synthesizing mathematical conceptions and their formalization through the construction of a Logo-based school mathematics curriculum. *International Journal of Mathematical Education in Science and Technology*, 18, 581–595. doi:[10.1080/0020739870180411](https://doi.org/10.1080/0020739870180411)
- Jeschke, S., Richter, T. & Seiler, R. (2005). VIDEOEASEL: Architecture of virtual laboratories on mathematics and natural sciences. *Proc. 3rd International Conference on Multimedia and ICTs in Education* (pp. 874-878). Cáceres/Badajoz: FORMATEX.
- Kafai, Y.B. (1995). *Minds in play. Computer game design as a context for children's learning*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.
- Kahn, K., Sendova, E., Sacristán, A. I., & Noss, R. (2011). Young Students Exploring Cardinality by Constructing Infinite Processes. *Technology, Knowledge and Learning*, 16(1), 3–34. doi:[10.1007/s10758-011-9175-0](https://doi.org/10.1007/s10758-011-9175-0)
- Laurillard, D. (2002). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*. 2nd Edition. Routledge.
- Lesh, R., & Doerr, R.H. (2003) Foundations of a models and modelling perspective on mathematics teaching, learning, and problem solving. In: Lesh, R., Doerr, R.H. (Eds.), *Beyond constructivism. Models and modelling perspectives on math. problem solving, learning and teaching*. Lawrence Erlbaum Associates, Mahwah, New Jersey, 3-33.
- Lesh, R., Hoover, M., Hole, B., Kelly, A., Post, T. (2000) Principles for Developing Thought-Revealing Activities for Students and Teachers. In A. Kelly, R. Lesh (Eds.), *Research Design in Mathematics and Science Education*. (pp. 591-646). Lawrence Erlbaum Associates, Mahwah, New Jersey.
- Mascaró, M., Sacristán, A. I. & Rufino M. (2014). Teaching and learning statistics and experimental analysis for environmental science students, through programming activities in R. In G. Futschek & C. Kynigos (Eds.), *Constructionism and Creativity - Proceedings 3rd Intl. Constructionism Conf. 2014* (pp. 407-416). Vienna, Austria: OCG.
- Mascaró, M., Sacristán, A. I., & Rufino, M. M. (2016). For the love of statistics: appreciating and learning to apply experimental analysis and statistics through computer programming activities. *Teaching Mathematics and Its Applications*, 35(2), 74–87. doi:10.1093/teamat/hrw006
- Mascaró, M. & Sacristán, A. I., (2018). Assessing learning through exploratory projects in constructionist R based statistics courses for environmental science students. In *Proceedings Constructionism 2018*, Vilnius, Lithuania.
- Muller, E., Buteau, C., & Sacristán, A. I. (2015). Through the Looking-Glass: Programming Interactive Environments for Advanced Mathematics. *Mathematics Today*, 51(6), 212–217.

- Noss, R. & Hoyles, C. (1996) *Windows on mathematical meanings. Learning cultures and computers*. Dordrecht, the Netherlands: Kluwer Academic Publishers.
- Olivera, M.A., Sacristán, A.I. & Pretelín-Ricárdez, A. (2013). Mathematical learning derived from virtual collaboration, exploration and discussion of free-fall videos, amongst continuing education students. In E. Faggiano & A. Montone (Eds), *Proceedings of the 11th International Conference on Technology in Mathematics Teaching (ICTMT11)* (pp. 232-237). Bari, Italia: University of Bari.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NY: Basic Books.
- Papert, S. & Harel, I. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>
- Pretelín-Ricárdez, A. & Sacristán, A.I. (2015). Videogame Construction by Engineering Students for Understanding Modelling Processes: The Case of Simulating Water Behaviour. *Informatics in Education*, 14(2): 265–277. DOI: 10.15388/infedu.2015.15
- Rich, P.J., Bly, N. & Leatham, K.R. (2014). Beyond Cognitive Increase: Investigating the Influence of Computer Programming on Perception and Application of Mathematical skills. *Journal of Computers in Mathematics and Science Teaching*, 33(1), 103-128.
- Sacristán, A. I. & Jiménez-Molotla, J. (2012) The Continuing Story of the Painless Trigonometry Projects: Eratosthenes' method and the Parthenon. In C. Kynigos, J. Clayson & N. Yiannoutsou (Eds). *Constructionism: Theory, Practice and Impact. Constructionism 2012 Conference Proceedings* (pp. 126-135). Athens, Greece: ETL, National and Kapodistrian Univ. of Athens.
- Sacristán, A. I., & Noss, R. (2008). Computational Construction as a Means to Coordinate Representations of Infinity. *International Journal of Computers for Mathematical Learning*, 13(1), 47–70. doi:[10.1007/s10758-008-9127-5](https://doi.org/10.1007/s10758-008-9127-5)
- Sacristán, A. I., & Pretelín-Ricárdez, A. (2017). Gaining modelling and mathematical experience by constructing virtual sensory systems in maze-videogames. *Teaching Mathematics and Its Applications: An International Journal of the IMA*, 36(3), 151–166. doi:[10.1093/teamat/hrw019](https://doi.org/10.1093/teamat/hrw019)
- Sacristán, A. I. (2017). Constructionist computer programming for the teaching and learning of mathematical ideas at university level. In Göller, R., Biehler, R., Hochmuth, R., Rück, H.- G. *Didactics of Mathematics in Higher Education as a Scientific Discipline*. khdm-Report 17-05 (pp. 124–131). Kassel, Germany: Universitätsbibliothek Kassel.
- Wilensky, U. J. (1993). *Connected Mathematics: Building Concrete Relationships with Mathematical Knowledge* (Doctoral dissertation). Massachusetts Institute of Technology. Retrieved from http://ccl.northwestern.edu/1997_prior/Wilensky-thesis.pdf
- Wilensky, U. (1995). Learning Probability Through Building Computational Models. In L. Meira & D. Carraher (Eds.), *Proceedings of the 19th Annual PME Conference* (Vol. 3, pp. 152–159). Recife, Brazil: International Group for the Psychology of Mathematics Education. Retrieved from <https://ccl.northwestern.edu/papers/pme19>

Back 100 000₍₂₎

Evgenia Sendova, jenny.sendova@gmail.com

Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences, Bulgaria

Abstract

I am sharing memories of the first conference *Children in Information Age*, held in Varna (Bulgaria) 32 years ago – both personal as well as extracted from the proceedings and two documentaries featuring the vision of eminent scientists and educators on computers in education from different continents. Based on discussions with researchers, teachers, parents, representatives of industry, discussed herein are their questions and visions in today's context. Below are some of the questions posed then, still relevant today:

In what way should the students be trained as to the future problems of rational use of computers? How can we make the best use of the computer in the future?

- *Shall we use computers to make the educational process more technical or more human?*
- *What does "computer literacy" mean in fact? How should we define literacy?*
- *Which statements made today would be obsolete in a few years, if not months?*
- *Should the computer be seen as an attraction in itself holding a child indoors with its artificial simulations, however realistic they might be?*
- *What are the new roles of the educators?*
- *What might we really like to see?*

After considering the views of the pioneers of the information age I am sharing some concerns based on a recent experience in observing uses of technology dictated mainly by the market models in the educational institutions.

The views of representatives of various stakeholders of education are presented as shared in a recent panel discussion on the future education we would like to see happen.

After considering some finding and recommendation published in a document of the European Parliament on the teaching and learning in the Digital era I conclude with an optimistic perspective based on the work with teachers and students in the frames of national and European educational projects in which I have been involved.

Keywords

teaching and learning in the information age; digital era

Introduction – How close is 32 years ago?

This paper reflects some of the ups and downs of my professional life, the length of which as far as my involvement in education is concerned, coincides with the number in the title.

In 2017 we celebrated 32 years of the first issue of *Children in Information Age* – a series of 3 international conferences held in Bulgaria (Varna – 1985, Sofia, 1987 and 1989) dealing with the problems of introducing computers in education. (The number 32 is more impressive in binary system, hence the title of the paper.) The subtitle of the Varna conference was *Tomorrow's Problems Today*.

The foremost objective of the Conference was to enable an exchange of opinions and concrete results from research and applied work of scientists and experts in using computers in the education of children at school, in extracurricular activities and at home. In order to outline a more complete picture of the problems to be faced by the teachers, psychologists, programmers and hardware designers, as well as by the strategists of educational policies, the organizers invited both supporters and specialists with reservations about the massive introduction of computers from the earliest years of education (Sendov, Stanchev, 1986, v).

The large number of outstanding researchers and specialists from the East and West (350 participants from 40 countries and representatives of UNESCO, IASA, WHO, IFIP and UNICEF) contributed to summarizing the experience of the most advanced countries in implementing computers in education and taking into account the positive and negative effects when planning the future work.

I would slightly rephrase a remark of André Gide: *Everything has been said before but since people prefer to start from scratch we have to keep re-reading what the pioneers were wondering about...*

In this case I decided to browse through the proceedings of the first two issues of *Children in Information Age* and was surprised and at the same time not very surprised to find wonderfully formulated ideas and questions which appear to be still relevant and challenging for the educators and the policy makers. Let me share some of them with you and challenge you to think which of them have already been solved and which (if any) sound obsolete. (During the late 40s it was considered that the first computers designed exclusively for high speed calculations, because of their price and difficulty to use, had no commercial future and their potential use was only for a few large military or industrial laboratories.) To speak of educating children with computers at the time would have been *pure nonsense* (Hebenstreit, 1986, p. 29). Even the first implementation of Logo was on a machine that, in today's terms, cost about \$1,000,000, and many thought the project would never be brought to any practical consequences (diSessa, 1986, p. 97).

Tomorrow's Problems Today (as seen in 1985)

A general question considered by most of the speakers was as follows:

- ***In what way should the students be trained as to the future problems of rational use of computers?***

In his plenary talk (Velikhov, 1986) the speaker (then Vice-president of the Academy of Sciences of the Soviet Union) shared the idea of introducing computers on a large-scale basis gained from the experience of the Siberian Department of the Academy of Sciences of the Soviet Union, headed by Academician Ershov. He argued that the training of the student should be based on the principle of unity in model building for the task to be solved, creating of algorithm for its solution and coding this algorithm for a computer. *Thus – he claimed - it is necessary to teach not only hardware and programming languages, but also **techniques for the practical problem solving**...* This process embraces three stages: (i) **comprehension stage** (covering the age of 7-11), during which the children conceive the computers as a new assistant and a new friend. A Logo-type language (called by Ershov ROBIK) can be used for about 2-3 hours weekly; (ii) **second stage** (covering the age 11-14), during which the students get acquainted with more professional use of the computer and learn: creating the model of the problem, the problem solving algorithm and finally, the program itself; (iii) **third stage** (the last two grades) – using the computers for **productive** work, based on acquainting the students with application program packages, so that they could use them in typical practical situations.

Related to this question is *how to make the best use of the computer in the future*. Richardson suggests that *as this attractive, useful and co-operative machine improves while the child grows, so must his attitude about how to use it be guided by his parents, teachers and his playmates* (Richardson, 1985).

A variation on the same question is considered in (Hebenstreit, 1985):

- ***Which scenario of introducing computers in education is the most promising: (i) computer awareness/literacy; (ii) computer as an intelligence amplifier, or (iii) computer assisted activities?***

Sub-questions he reflects on include: *Which aspects of programming could be useful in education? In what programming language (if at all) should we train the children today when in 10 years we might program in natural language and it might be Japanese? Does the concept of problem solving have semantic content and if so, is it a method?*

Hebenstreit claims that the general act of problem solving cannot be reduced to the specific task of automating procedures and the teaching of programming and/or algorithms is thus of no use except to specialists in informatics. The real challenge for the educators according to him should be to prepare the children to solve problems which we don't even know today, firstly by developing insight, intuition

and imagination based on a solid understanding of the basic paradigm of science, and secondly – by helping them to identify which part of the problem requires their own intellectual effort and which part can be assigned to the computer as an assistant.

One of the most interesting open questions he poses (open till today to the best of my knowledge) are the following:

- *What is the impact of a text-processing package on the speed of learning to write and speak?*
- *According to Piaget the child becomes progressively conscious that s/he is a performer compared to the outside objects. A computer with a program can become a performer while still being an object. Does this change the way a child understands the outside world and his/her relationship to the world? If it does, what does it change?*
- *How are we going to change our whole system of education to take into account the existence of tools, so powerful that they would have been unthinkable 10 or 20 years ago? Furthermore, how are we going to integrate these in a new set of coherent curricula at all levels, so as to educate everyone from elementary school through university to make the most efficient use of these tools?*

The ultimate purpose of computing as formulated by Hebenstreit *should not be to turn people into servants of the computer but to develop in people those qualities which are unique in men, i.e. **to help people to become more human*** (ibid, p. 45) – a wonderful paraphrase of Richard Hamming's statement that *the purpose of computing is insight and not numbers*.

Talking about *becoming more human* let me bring an interesting analysis of the impact of the emerging technologies on the educational system in Japan as presented in (Shiba, 1985). The author of this analysis claims that the real problem in introducing computers in school does not lie in the hardware or the software but in what is called the *human ware* and *heart ware*, since the human element requires a long period of time to accept any recognizable changes. Shiba expresses his concerns about the competition among the children founded on intellectual ability, student violence both at school and at home, bullying among students. His pessimistic view at the time concerning a rapid progress in the Japanese educational system was due to the *increasing number of teachers who regard the work simply as a means of earning money in exchange for a given number of hours' work*. An essential reason for the lack of vitality in the Japanese educational system according to him had been the limited freedom of teachers to change the content of what is taught influenced by the emphasis on assessing both schools and students by means of standard deviation.

Shiba concludes that for the use of computers to be truly effective it is necessary (i) to create a new type of educational institution outside of the scope of the traditional schools, and (ii) employ a new management system which will enliven this new type of institution and promote the educational reforms necessary for the coming society.

The need of educational reforms expressed by practically all the participants in the Conference comes along with the need of answering the question:

- ***How should we define literacy?***

The limited interpretation of literacy as the condition of being literate, i.e. *able to read and write* should be broadened to the *condition of being educated* (closer but not too precise).

It was at the IFIP congress in 1981 when A. P. Ershov talked about programming as the *second literacy* but serious reservations existed even then as to whether the content of the new literacy should be based solely on a knowledge of programming (Ershov, 1981).

The discussion about what literacy will be needed in future society and what cultural change would have to be the consequence of this need was reflected in a number of papers in both – the 1985- and 1987-issues of *Children in Information Age*.

Several speakers in the first one shared the opinion that the definition of computer literacy is still not clear and that *in the US "computer literacy" could mean anything – from learning to use the keyboard, to a short lecture on the history of computers, to an introduction into a programming language such as Basic...* (White, 1986, p. 48). Sylvia Charp notes in her paper (Charp, 1986, p. 189) certain trends

though, viz. *Programming is becoming decreasingly important. The emphasis is how to best use computers in a variety of applications... Though courses in Computer Literacy are still being given across the US, questions are being asked: What happens after Computer Literacy? What technological and conceptual tools are needed to process a continuing flow of new information?*

The metaphor of computer literacy as second literacy and the analogy with printing was used in (Sendov, 1986, p. 197) to give an idea of the level of the information technology at the time and how it stood in relation to the printed word. As Bl. Sendov had seen it, *these were the times of Gutenberg.*

The theme of defining literacy was continued by Kurt Kreith (Kreith, 1987), who related the search for new meaning of literacy to problem solving. His concerns are that *mathematics, improperly applied, can serve to distort or interfere with rational thought. Thus, in the future, mathematical skills and knowledge may be needed not only to search for societal problems, but also to protect our civilization against inappropriate attempts to represent it in mathematical terms.*

Interesting questions concerning the new literacy are raised by Pamela Fiddy in (Fiddy, 1987). *How do we recognize the New Literacy and the New Non-literacy? What is the difference between New literacy for children, and New literacy for their teachers, parents and other adults? What New Literacy skills do adults need? What are the implications for the development of New Literacy Skills in schools?* Her interpretation of literacy was that *people who have achieved New Literacy use IT with confidence, appropriately and effectively and realize the potential of IT to apply it in innovative and creative ways.*

The software designers were mainly concerned with the principles behind an integrated computational environment for education relevant to the needs of Tomorrow's society. The main question considered in (diSessa, 1986) is formulated as follows:

- ***What should we do beyond Logo and other present educational uses of computers with the increment of power that will come affordable within the next 5 years or so?***

Although aware that it is difficult to deal with disruption of the routine and expected practice diSessa shares the importance of experiencing the *having of a wonderful idea of your very own* during the learning process. His message is that *feelings of achievement and personal satisfaction of the learner (even when they flow from rare events) can influence the whole educational process. But designing such events in the educational system is exceedingly hard (ibid, 1986, p. 98).*

One of the most important questions considered at the Conference was:

- ***What are the new roles of the educators?***

A pilot study in Egypt (Owais, 1985) brings the attention to some related questions which had been overlooked concerning the use of computers in education: *(i) What is the objective of training pupils – to prepare them as future producers or to help them increase their knowledge; (ii) what is the objective of training teachers – to ease their job or to help them to become more innovative and creative; (iii) to make the teacher able to contribute to the adaptation of packages written in foreign languages or to get him able to design and perform programs to alleviate the illiteracy...*

The need of a more relevant teacher education was expressed in (Wibe, 1985, p. 856) who states that during *the last years a lot of teachers have been qualified to teach informatics but very few have learned how to use it as an educational tool in the different school subjects.*

There was a general consensus (Sendov, 1986) that (i) *no computer can replace a teacher*; (ii) *the computer is only a tool*; (iii) *students should be taught how to learn and teachers should encourage and develop the desire and the need for continuing study.*

Are we passing the Constructionism torch smoothly to the next generations of educators?

A number of recent events dealing with demonstrations of new educational software and hardware provoke disturbing feelings not only for the constructionism community.

At an international exhibition marketing innovations in education technology attended by more than 40 thousand visitors there were (to me at least) more examples of how NOT to use technology than of how to use it.

A program generating a sentence of randomly chosen words was called "Poet". Similarly, a program generating a sequence of randomly chosen notes was called "Composer". The authors had not implemented any concepts related to the structure of a poem or a musical composition. Not surprisingly, they had not heard of books such as *Exploring Language with Logo* (Goldenberg, Feurzeig, 1987).

At a very recent conference embracing researchers, educators and teachers (held in Bulgaria) there was a workshop led by young representatives of firms promoting programming courses for young children. They were demonstrating a couple of robots moving in a maze on the floor. After showing the commands needed to move the robots (not RIGHT and FORWARD as you might expect) a mother from the audience asked: *What would you do after a child who has already learned these commands gets bored?* The answer was: *No, problem at all! We have five other programming environments to offer...*

There are teachers who believe that just because something is modern and attractive (e.g. the computer video games), we should try to implement it in the classroom without being sure about the effect, about what we gain as teachers/learners...

In a chapter on Virtual reality in education, I had to review a project on Dante for 4th-graders in which the authors claimed that a simplified and filtered text has been adapted for the use of kids in primary school, allowing a better and clearer understanding of Dante's Inferno. To the best of my knowledge the mysterious tercet of Dante in his Inferno 9, 61-63 *urges the reader to consider the doctrine concealed under these strange verses* (Fowlie, 1981). The video-clip demonstrating the class atmosphere showed kids reading, nobody was looking at the screen showing a fire (possibly to represent the hell). The authors of the proposed Chapter did not discuss what the kids had gained in terms of knowledge which would not be doable without a virtual reality. What if they were just drawing their ideas about the hell, discussing the pictures of Gustave Dore (who was not even mentioned), and if they would have just organized a play themselves instead of using an avatar with the name of Dante on his back...

Just for a comparison with much older "new technology" I reread a paper (Tabov, Muirhead and Vassileva, 1999) in which the authors present a course for interdisciplinary and integrated education dealing with Dante and his world. In particular, the birth and death dates of one of Dante's ancestors is calculated by means of the Geomland software. Astronomy, mathematics, history, and literature (including the difficulties of translating Dante) were highlighted in teaching. Finally, it is suggested that the dates normally presented by editors of Dante's Divine Comedy may be incorrect. The authors discuss the problems and questions relevant to this sort of education and especially those involved when teaching mathematics and the humanities together.

Today, it is troubling to see how often educational software designers and producers trivialize the notion of *creativity* by claiming that children can easily become composers, poets, artists, filmmakers - just by using their fantasy and rearranging the elements of a story, a poem, a famous painting or the bars of a musical piece.

In a nutshell, when I hear people telling me: *O-o-oh, Logo, the little turtle drawing squares... But isn't it old fashioned?* or even worse: *Constructionism? How many papers dealing with this notion have you read recently which are not part of conference proceedings?* I feel bitterness. This is not people's fault though. We owe to the educators, teachers and children the *powerful educational ideas* we have been nurtured with thanks to the constructionism and we should not remain indifferent if these ideas are often darkened by shiny new technology used per se.

And to quote the pioneer of constructionism, Papert, this claim is not based on an arrogant belief that the inventors of this educational philosophy are smarter than the rest. It is based on the belief that the *constructionism was not invented at all, that it is rather the expression of liberation from the artificial constraints of pre-digital knowledge technologies* (Papert, 1999).

What education do we need for the future (seen as of today)?

In a recent panel discussion on what education we need for the future (I had the honor to moderate) the panelists were colleagues and former students of mine representing various stakeholders of education. What follows are fragments of their statements transcribed from an audio recording.

M.B. (parent): *Today's culture in a number of schools is for the children to stay quiet, not to touch anything so as to avoid damages, to be disciplined. It is difficult for the teachers to cultivate and develop the soft skills of their students since they follow the model by which they themselves have been educated. A modern pre-service education should include learning to learn so that it could be demonstrated later in class setting.*

V.T. (teacher): *Teachers motivation is crucial. It is not sufficient though for a school to have a single teacher ready to spark the curiosity for science, the love for knowledge. Communities of teachers who act like researchers are needed.*

D.P. (teacher) – *Few teachers are ready to leave their comfort zone. We are time-poor and even when I manage to achieve an enthusiastic atmosphere I feel like single swallow. The need of more time and efforts discourage even teachers who have been hooked at the beginning.*

T.A. (teacher) *I was professional programmer and musician. My goal is to participate in the foundation of schools of new type. Today's economy does not need just STEM specialists; people who are built personalities are needed. The style of teaching should be constructionism, but the key question is how to prepare teachers who act as masters, designers of experience.*

S.H. (researcher and mentor) *The main problem of today's society is the lack of patience – people want immediate results. And the innovations require time and risk-taking with no guarantee for ultimate results. The IBL in its highest level, open inquiry, is very important idea to be implemented in school. Experiments, observations, explorations should be in the core of education so that the students feel they are participating in the creation of something new. The students' mentors at that level are rarely among the teachers – there is gap between the traditional teacher education at the university and what the modern teaching strategies require. A lot of competences could be acquired in any subject provided the students are active participants in the learning process.*

T.K. (software company founder): *We have to learn to work not for the others but with the others, to build connections and become stronger. We have to have the will of formulating our goals – no matter how powerful, the machine cannot set the direction. We have to become very good in setting goals. Dreyfus Model of Skill Acquisition could be used in any profession. The role of the teacher should be the one of the master: I'll teach you how to become master vs. I'll show to you what master I am. As mentors we can help students raise their intuition, learn how to learn. Recently I used my high school mentee to teach me about block-chain technology – by teaching me he became better.*

K. D. (researcher, involved in gifted education) *Since we can't predict the future professions, we are talking about key competences and transfer of knowledge. In its kernel the education should be oriented to wide range fields. The competence of selecting and analyzing information is crucial.*

O.K. (University Professor) *The more important skills required in the workplace will increasingly be communication skills. I do not think computer training can make much of an improvement in personal communication skills. If it is in the form of training with games - yes, it would evolve thinking towards empathy. On the other hand, analytical skills can be acquired through computerized learning. Perhaps the "truth" is in the balance. In the near future, the notion of "computer literacy" will disappear completely. It is already out of use - today we are talking about "digital competences", which includes communication and basic problem solving in all aspects of life. It is believed that digital skills today help to master other key competences such as communicative, language skills or basic math and science skills. The future role of the educators will be increasingly in the aspect of mentoring and supervising.*

B.S. (researcher involved in educational reforms) *The real question for me is not so much what education but rather what upbringing we need for the democratic society. Education, especially in the public schools and universities, must be engaged with the formation (upbringing) of responsible and loyal citizens... We could declare that the digital technology is an excellent tool for education; the*

problem is how to use this tool for the upbringing of the pupils... This problem is especially difficult in the so called *new democracies*. *The use of market models in the educational institutes is not a consequence of the democracy but rather the result of the weakness of the democracy and the aggression of the market. After decades of successful use of digital technology for education, it is clear that the quality of education depends mostly of the quality of the teacher. All teachers have to be potential researchers.*

Finally, the panelists agreed that we all as teachers have to be guides for our students' development, but at the same time we have to be ready to learn with them and learn from them

What to do (and not to do) in education in the digital era

Let us cast a glance at a recent document dealing with the education in digital era (Lonka, Cho, 2015), which states that so far the digital revolution has not transformed most schools or most teaching and learning process in classrooms. The research of the authors indicates that the students with the best skills in technology are also the ones who are most bored and disengaged at school. Thus finding meaningful ways of using technology not only for learning but also for collaborative knowledge creation is needed.

Some of their key findings and recommendations include: (i) *It is important to base our conclusions on perceiving learning as knowledge creation, rather than emphasizing mere knowledge acquisition;* (ii) Well-being and Social and Emotional Learning (SEL) are at least as important as other 21st century skills (such as media literacy, cultural awareness, and complex problem solving); (iii) Instead of computer-supported learning, it would be advisable to talk about new forms of Socio-Digital Participation (including media literacy, such as using social media and search engines); (iv) Systematic development of flipped and inquiry-based learning programs with meaningful use of technologies would be advisable; (v) We need constant reforms in schools and teacher education so as to follow the important developments of society. Perhaps too much time has been spent looking at test results, such as PISA.

A recent *Call to Action* concerning (i) students and learning with ICT; (ii) professional development of integrating technology, and (iii) educational system policies for infusing technologies, appeared as a result of the work within EDUsummit 2017 *Rethinking Learning in a Digital Age* (Voogt, Knezek, Lai, 2017, pp. 16-17). EDUsummit (International Summit on ICT in Education) is a global knowledge building community of researchers, educational practitioners and policy makers committed to supporting the effective integration of research and practice in the field of ICT. Approximately 90 leading researchers, policy makers and practitioners spanning all continents, attended EDUsummit in Borovets, Bulgaria, September 18-20, 2017 to discuss the research, policy, and practice challenges faced in 9 thematic groups and come up with recommendations for the researchers, teacher educators and policy makers. The summary report of the work of the thematic groups together with cross-analysis of their recommendations was prepared by Voogt, Knezek, and Lai (*ibid*).

Instead of conclusion: So...what do we do now?

Baring in mind the importance of the continuous professional development of teachers expected to implement inquiry based learning approach in their practice a research team at the Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences took the initiative to provide scientific, methodological and technical support in inquiry based mathematics, informatics and IT education (IBMIE) at all levels and forms in a national context (Chehlarova, Kenderov, Sendova, 2017). This support is carried out by organising various types of PD courses and by developing open access learning environments that enhance IBMIE with a focus on the acquirement of key competences.

The inquiry based learning supported and enhanced by digital technologies in my country has its roots in an experiment of the Research Group on Education (RGE) founded by the Bulgarian Academy of Sciences and the Ministry of Education) in 1978 (Sendov, 1987).

RGE - a model for a technology-prompted curriculum

The guiding principles of RGE were *learning by doing and making* and *integration of the school subjects* (Nikolov, 1987). The educational resources developed for the 29 experimental schools included textbooks, teacher guide-books, and unified (Logo-based) computer microworlds tuned to specific subject domains. The newly developed curriculum enabled students to pass gradually from constructing controllable models, through various problem-oriented microworlds for explorations in natural languages, music and art to a fully programmable microworld for explorations in Euclidean geometry. The tasks dealing with modeling pieces of art, music, poetry, and writing programs to generate such works were meant to motivate students to learn about structures and use their programs as *materialized hypotheses* of their observations (Sendova, 2001). The experiment lasted for 12 years starting from 1979. Although it did not lead to essential changes in the Bulgarian educational system as a whole, the RGE model set a good ground for a series of European educational projects involving ICT enhanced inquiry based mathematics and science education (e.g. DALEST, InnoMathEd, Math2Earth, DynaMAT, Fibonacci, Mascil, KeyCoMath, STEM PD Net, Scientix).

Some recent promising educational practices in Bulgarian setting

Thanks to the system for PD of teachers in mathematics, IT and informatics developed by IMI-BAS, a network of such teachers acting as multipliers of the IBL ideas has been established. The current activities of IMI-BAS include various types of PD courses and events, as well as open access learning environments related to STEAM (science, technology, engineering, arts and mathematics).

- **Professional development (PD) courses for teachers in mathematics and IT**

The main goal of the courses is in harmony with the most recent educational strategies for updating the math and science education in the EC countries: the development of key-competences by implementing the inquiry based learning in integration with the world of work. These PD courses are based on a team work (of the lecturers and the participants alike) and implement educational models adaptable to various school settings. The teachers work on pedagogical problems related with: formulating their own math problems reflecting real-life situations, not solvable with the current math knowledge of the students but allowing for explorations leading to a good enough approximation of the solution; studying and proposing methods for tackling problems which are unstructured, or whose solutions are insufficient or redundant; solving “traditional problems” with “non-traditional” data, for which the use of a computing device is necessary; formulating more relevant evaluation criteria for the students’ achievements; assessment of learning resources in terms of formation and development of IBL skills; project-based work with presentation of the results (Zehetmeier, Piok, Holler et al., 2015).

Professional development courses in which teachers and students work collaboratively on real-life problems have turned out to be especially fruitful. The format of such courses is typical for the Summer Research School organized annually by the High School Student Institute (HSSI) of Mathematics and Informatics one of the goals being to help teachers improve their mentoring skills (Sendova, 2014).

- **Learning scenarios in support of cross-curricular integration**

A good repository of digital resources is the Virtual School Mathematics Laboratory (VirMathLab) being developed by IMI-BAS (Kenderov, Chehlarova, Sendova, 2015 a), which contains about thousand scenarios with dynamic files transparent for the users (<http://www.math.bas.bg/omi/cabinet>). The design and the implementation of these scenarios is just an element of a more ambitious goal – we expect our students to look for manifestations of geometric congruences, discover them and use them in various activities, and thus – to be able to find patterns and relationships deepening their knowledge and understanding of the surrounding world.

- **New types of mathematics contests**

Mathematics with a computer and Theme of the month are new type of contests based on the VivaCognita computer platform (Kenderov, Chehlarova, Sendova, 2015 b; Chehlarova, Kenderov, 2015; Kenderov, Chehlarova, 2016). Students (3-K12) are invited to work on a chain of problems in increasing difficulty. Some of the problems in both competitions are accompanied by GeoGebra files which facilitate the exploration of the mathematical essence of the problem.

Conclusions

With all our efforts we are trying to help teachers create an atmosphere where the students would not tell themselves: *I am a good student, because I got so many points on the test, I'll take the exam, I'll enter the university...* But they would rather think about the excitements the genuine learning offers: *How interesting, I wonder what will happen if... I feel like a real scientist! I am not afraid to try something nobody has tried before...* And even to express certain disappointment that Archimedes had preceded them with his discoveries... (a genuine remark by a 6th grader in the context of constructing Archimedean solids by means of plastic straws). As for the teachers they should not feel embarrassed when they don't know the answer but would demonstrate how they are looking for it (possibly together with their students).

For such an attitude to education to be cultivated however the whole society should be adapted to the digital era and learn how to learn throughout life. This is the hope for the society to become *even more human* in the digital era...

Can we agree now with Paul Valéry *that the future is not what it used to be...?*

References

- Chehlarova, T., Kenderov, P., Sendova, E. (2017) A European network for professional development of teachers (and the role of IMI-BAS as a center for inquiry based mathematics and informatics education), In Proceedings: *Mathematics and Education in Mathematics*, 46th Spring Conference of the Union of Bulgarian Mathematicians, Borovets, 9-13 April, pp. 328-338
- Chehlarova T., Kenderov, P. (2015): Mathematics with a computer—a contest enhancing the digital and mathematical competences of the students. In: Kovatcheva, E., Sendova, E. (Eds.). UNESCO International Workshop: *Quality of Education and Challenges in a Digitally Networked World*. Za Bukvite, O'Pismeneh. Sofia, 50–62.
- Charp, S. (1986) Issues and Trends on the Use of Computers by Children in an Information Age. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 189-194
- diSessa, A. (1986) Principles for the Design of an Integrated Computational Environment for Education. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 97-109
- Ershov, A. P. (1981) Programming, the second literacy. In Proceedings: 3rd IFIP World Conference on Computers in Education (WCCE81), Lausanne
- Fiddy, P. (1987) Welcoming people to the Age of Information Technology: Teaching experiences leading to a definition of New Literacy. In Preprints: *Children in the Information Age: Opportunities for Creativity, Innovations and New Activities*, Sofia, Bulgaria, 19-23 May, pp. 182-195
- Fowlie, W. (1981) *A Reading of Dante's Inferno*, University of Chicago Press
- Goldenberg, E. P., Feurzeig, W. (1987) *Exploring Language with Logo*. The MIT Press, Cambridge, MA
- Hebenstreit, H. (1986) Children and Computers. Myths and Limits. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 29-45
- Kenderov, P., Chehlarova, T. (2016): Extending the class of mathematical problems solvable in school, *Serdica Journal of Computing*, Volume 9, No. 3-4, 2015, pp. 191-206, ISSN 1312-6555
- Kenderov, P., Chehlarova, T., Sendova, E. (2015, a): A Virtual Math Laboratory in support of educating educators in IBL style, in: Maaß, K., Barzel, B., Törner, G., Wernisch, D., Schäfer, E., Reitz-Koncebovski, K. (Eds.): *International approaches to scaling-up professional development in mathematics and science education*, pp. 167-176
- Kenderov, P., Chehlarova, T., Sendova, E. (2015, b): A Web-based Mathematical Theme of the Month, *Mathematics Today*, vol. 51, no. 6, pp. 305-309 ISSN 1361-2042

- Kreith, K. 1987 Problem Solving and the search for a new meaning of literacy. In Preprints: *Children in the Information Age: Opportunities for Creativity, Innovations and New Activities*, Sofia, Bulgaria, 19-23 May, pp. 370-386
- Lonka K., Cho, V. (2015) Report for EU Parliament 2015: Innovative Schools: Teaching & Learning in the Digital Era: Workshop Documentation, [http://www.europarl.europa.eu/RegData/etudes/STUD/2015/563389/IPOL_STU\(2015\)563389_EN.pdf](http://www.europarl.europa.eu/RegData/etudes/STUD/2015/563389/IPOL_STU(2015)563389_EN.pdf) (April 5, 2018)
- Nikolov, R. (1987): Integrating Informatics into the Curriculum, in *Education & Computing*, North-Holland, 1987, pp. 369-374
- Owais, A. (1985) Computers and Children's Creativity in Egypt: A Pilot Study. In Proceedings: *Children in an Information Age: Tomorrow's Problems Today*, 6-9 May, Varna, vol. II, pp. 630-657
- Papert, S. (1999) What is Logo? Who needs it? In *Logo Philosophy and Implementation*, LCSl, V—XVI, <http://www.microworlds.com/support/logo-philosophy-implementation.html> (5.01.2018)
- Richardson, J. (1985) *The computer: tool, helper and friend of young people*. In Proceedings: *Children in an Information Age: Tomorrow's Problems Today*, vol. II, Ministry of Education, Sofia, pp. 457 – 476
- Sendov, Bl., Stanchev, I. (1986) Forward of *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, p. v-vi
- Sendov, Bl. (1986) *Children in information age* (a concluding talk). In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 195-200
- Sendov, Bl. (1987): Education for an Information Age, in *Impact of Science on Society*, v37, n2, pp.193-201
- Sendova, E. (2001) Modeling Creative Processes in Abstract Art, Poetry and Music, *International Journal Information on Theories and Applications*, vol. 8, N 3, FOI-Commerce Sofia, pp. 122-132
- Sendova, E. (2014): You do – you understand, you explore – you invent: the fourth level of the inquiry-based learning, in Futschek, G., Kynigos, C. (Eds.) *Constructionism and Creativity*, Proceedings of the 3d International Constructionism Conference, August 19-23, Vienna, Austria, pp. 103 – 112
- Shiba, S. (1986) Information society and education. Past experiences and new Trends in Japan. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 11-28
- Tabov, J., Muirhead, J., Vassileva, A. (1999) Dante and the humanities. *The Teaching of Mathematics*. Vol. II, 1, pp. 31- 40 <http://elib.mi.sanu.ac.rs/files/journals/tm/2/tm212.pdf> (April, 2018)
- Velikhov, E. P. (1986) On the Start-up Course in Informatics and Computer Technology in the Curriculum for Soviet Schools. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 5-10
- Voogt, J. Knezek, G., Lai, K.-W (Eds.) (2017) *Rethinking Learning in a Digital Age*, EDUsumMIT Summary Reports (<http://www.punyamishra.com/2017/11/30/edusummit-2017-summary-report-released/>, April 5, 2018)
- White, M. A. (1986) The future of Electronic Learning for Children. In Sendov, Bl., Stanchev, I. (Eds.): *Children in an Information Age: Tomorrow's Problems Today*, Selected Papers from the International Conference, Varna, Bulgaria, 6-9 May, 1985, Pergamon Press, pp. 47-56
- Wibe, J. (1985) Computer Technology in the Norwegian Educational System. In Proceedings: *Children in an Information Age: Tomorrow's Problems Today*, vol. II, pp. 853 - 871
- Zehetmeier, S., Piok, M., Holler, K., Kenderov, P., Chehlarova, T., Sendova, E., Gehring, C., Ulm, V. (2015) Concepts for In-Service Mathematics Teacher Education: Examples from Europe. In: Gehring, C., Ulm, F. (Eds.) *Developing Key Competences by Mathematics Education*. ISBN 978-3-00-051067-0 University of Bayreuth, Germany pp. 23-33

Rock Bottom, the World, the Sky: Catrobat, an Extremely Large-scale and Long-term Visual Coding Project Relying Purely on Smartphones

Wolfgang Slany, wolfgang.slany@tugraz.at

Kirshan Kumar Luhana, kirshan.luhana@student.tugraz.at

Matthias Mueller, mueller@ist.tugraz.at

Christian Schindler, cschindler@ist.tugraz.at

Bernadette Spieler, bernadette.spieler@ist.tugraz.at

Institute of Software Technology, Graz University of Technology, Austria

Abstract

Most of the 700 million teenagers everywhere in the world already have their own smartphones, but comparatively few of them have access to PCs, laptops, OLPCs, Chromebooks, or tablets. The free open source non-profit project Catrobat allows users to create and publish their own apps using only their smartphones. Initiated in 2010, with first public versions of our free apps since 2014 and 47 releases of the main coding app as of July 2018, Catrobat currently has more than 700,000 users from 180 countries, is available in 50+ languages, and has been developed so far by almost 1,000 volunteers from around the world (“the world”). Catrobat is strongly inspired by Scratch and indeed allows to import most Scratch projects, thus giving access to more than 30 million projects on our users’ phones as of July 2018. Our apps are very intuitive (“rock bottom”), have many accessibility settings, e.g., for kids with visual or cognitive impairments, and there are tons of constructionist tutorials and courses in many languages. We also have created a plethora of extensions, e.g., for various educational robots, including Lego Mindstorms and flying Parrot quadcopters (“the sky”), as well as for controlling arbitrary external devices through Arduino or Raspberry Pi boards, going up to the stratosphere and even beyond to interplanetary space (“the sky”). A TurtleStitch extension allowing to code one’s own embroidery patterns for clothes is currently being developed. Catrobat among others intensely focuses on including female teenagers. While a dedicated version for schools is being developed, our apps are meant to be primarily used outside of class rooms, anywhere and in particular outdoors (“rock bottom”, “the world”). Catrobat is discovered by our users through various app stores such as Google Play and via social media channels such as YouTube as well as via our presence on Code.org. Sharing, remixing, and collaboration is actively encouraged and supported. Catrobat has a very long term perspective in that it is independent of continuous funding and actively developed in a test-driven way by hundreds of pro-bono volunteers from around the world. Our aim is to grow by a factor of thousand and reach a billion users by 2030. We warmly welcome new contributors in every imaginable field and way with open arms. Please join us and contact me via wolfgang@catrobat.org today!

Keywords

Pocket Code, Game Design, Gaming, Gender Inclusion, Coding, Mobile Learning, Social Inclusion, Constructionism, Girls, Teenagers, Apps, Smartphones, Tinkering

Introduction: Background, mission, and history

Knowledge in Computer Science (CS) is essential, and industries have increased their demand for professionals that have technical experience. The next generation of jobs will be characterized by new standards requiring employees with computational and problem solving skills in all areas, even if they are not actual technicians (Balanskat and Engelhardt, 2015). However, the number of young people, and women in particular, choosing to study and work in Information and Communication Technology (ICT) fields is decreasing dramatically (NCWIT, 2015; NCWIT, 2017; European Commission, 2016a). In the last decade, European technology employment has grown three times faster than all employment

in total. The continuous improvements of technology and the numerous advancements in industrial processes made it possible to develop autonomous



Figure 1: Pocket Code's UI

vehicles, robotics, 3D printing, genetic diagnostics, or Internet of Things (IoT) technologies. These technologies are already part of everyday life, and there is a corresponding growing worldwide need for qualified scientists, engineers, and technicians. For these reasons, society and governments have mandated that teenagers should acquire computing and coding skills (European Commission, 2016b), or even a new way of (critical) thinking and problem solving skills (Wing, 2006; Kahn, 2017; Tedre and Denning, 2016; Mannila *et al.*, 2014). Presenting coding as a range of diverse skills which can be learned by adapting ideas from games is a generally applicable concept. Thus, a gamified and constructionist concept should hold teenagers' focus to actively participate by activating intrinsic and extrinsic motivators (Ryan and Deci, 2000). Games can be played everywhere, including on smartphones, tablets, and other digital devices. Moreover, the mobile game market continues to grow faster than other game industries, e.g., the number of game apps on Google Play grew by 28% in 2017 (Jingli, 2017; Takahashi, 2017).

Catrobat's approach is inspired by Piaget's Constructivism theory 1948 (Piaget and Inhelder, 1967), starting with first computer programming courses at the MIT in 1962 (Greenberger, 1962), and refined with Papert's Constructionism concept in 1980 (Papert, 1985). Since then, different approaches were used to motivate kids for coding. With our free open source non-profit project Catrobat our goal is to provide computational thinking skills for everyone, especially teens from less developed areas where other computational devices such as PCs are almost non-existent.

Catrobat's apps and services have been immensely influenced by MIT's Scratch³⁰ project, and we consider Catrobat to be Scratch's little sister project for smartphones. Scratch itself has been strongly shaped by Papert's powerful ideas, and extends Papert's "Low Floor" but "High Ceiling" for the Logo programming language for kids (i.e., easy to start, but allowing to develop more complex projects as well) by adding "Wide Walls", emphasizing that Scratch supports a wide variety of projects as well as ways to learn and play, according to the needs and interests of its users (Resnick 2017). Going beyond Logo and Scratch's metaphor of the room, Catrobat's smartphone based approach allows to literally break down the walls of the room and move outdoors, thus inspiring Catrobat's mission statement, which is "Rock Bottom, the World, the Sky". "Rock Bottom" because on the one hand we aim at building upon and going beyond Scratch's focus on making the first experiences in coding as easy and satisfying as absolutely possible for our teenage user group, e.g., through a physics engine that is much easier and intuitive to use that similar concepts in other game making environments. On the other hand, because of our reliance on smartphones, it also is meant to evocate that our users can and, to a large percentage, do leave the "room" to code outside or create outdoor projects, in some cases literally "on the rock". Many of our users are indeed developing their projects while on the go, far from classrooms and their homes, and have developed apps that take advantage of the various sensors such as the

³⁰ Scratch MIT: <https://scratch.mit.edu/>

cameras, GPS, compass, or acceleration sensors that are built into smartphones and that allow to create, e.g., augmented reality, geocaching, or dynamic outdoor sports games. Additionally, today's teenagers all over the globe have, to an already very large degree and also increasingly, their own smartphones readily available and permanently connected to the Internet, even in rural areas in Africa and other regions in the world where there may be no central power supply at home and kids charge their phones via solar panels in a central community facility of their village. Catrobat also has begun to become available in languages that are not supported by the phones makers themselves, such as Swahili, Gujarati, or Sindhi. Catrobat thus strives at reaching out to all corners of "the World" in an effective and efficient way that is indefinitely sustainable. The reliance on already existing smartphones, as well as the free open source character of Catrobat, which lets it thrive with little to no funding, also are significant aspects that allow Catrobat to scale up by avoiding the high costs and logistics that have hampered the success of similarly motivated projects in the past. Regarding the third part of our mission statement, Catrobat allows already since 2017 to program drones flying autonomously in "the Sky" (in particular, the popular Parrot Augmented Reality Drone 2.0), with a real-time video being transmitted to Pocket Code's screen, under full programming control by the user. While coding with Pocket Code has been done in airplanes, for the future we plan on going even higher. Because phones are small, lightweight, and portable, off-the-shelf Android phones have already been used to control balloons up to the stratosphere, and our extensions via Bluetooth and local WiFi connections to battery powered Arduino and Raspberry Pi allow to control any hardware of these flying computational systems, as long as the isolation keeps the harsh environment at bay and there's enough energy. On a further note, PTScientists' private enterprise Moon rover mission project³¹, scheduled to lift off in 2019 and sponsored by, among others, Vodafone, Nokia Bell, Audi, and Red Bull, has several experiments on-board that rely on regular Android phones to lower the costs of otherwise extremely expensive "rocket science" hardware, with Vodafone and Nokia Bell sponsoring the installation of a 4G data network based on standard phone transmission technology from the rovers to the base station on the Moon and from there back to Earth. One of our dreams is that we will empower kids to use Pocket Code to design experiments and to program autonomous robots on the Moon, on Mars, and possibly even farther away. There is a thriving worldwide PhoneSat³² community led by NASA that has launched a large number of nanosatellites based on 10x10x10cm cubesats using unmodified consumer-grade off-the-shelf Android smartphones and standard Arduino boards, which both would immediately work with Catrobat's apps. The sky has no limit, both in the concrete as well as in the metaphorical sense. Regarding the latter, our goal is to allow every kid to create complex, high resolution, and high performance apps using our tools, which they will be able to offer to other users, even commercially. Indeed, we allow our users to compile their apps into real Android apps, sign them with their own developer key, optionally add ads via their own AdMobs account, and sell them on Google Play for real money, thus directly empowering them to leave the limitations of the metaphorical "room" and bring their creations to the outside world.

On a historical note, one of the initial motivations for starting the Catrobat stems from Neal Stephenson's science fiction book "The Diamond Age: Or, A Young Lady's Illustrated Primer: a Propædeutic Enchiridion in which is told the tale of Princess Nell and her various friends, kin, associates, &c.". In the book, human tutors are hired anonymously on demand by a very affluent industrialist and aristocrat, to remotely educate a young girl, initially a toddler, living under gruesome conditions, by mistaking her, because of circumstances described in the book, for a princess for whom the primer was actually created. The illustrated primer, which is highly portable and has a voice- and touch sensitive interface that allows her to communicate with her tutors, accompanies Nell throughout her adolescence up to young adulthood, when she becomes a leading force and changes the fate of millions of the most underprivileged kids on Earth. One of the main story lines spanning a large part of the book is the acquisition of computational thinking skills by Nell through the Illustrated Primer. The book has won the Hugo and Locus science fiction awards, and was also cited by the developers of the One Note per Child Project as well as of Amazon's Kindle as a motivational inspiration. In 2017, Catrobat won the "Closing the Gender Gap" prize for a new subproject called "Remote Mentor", in which we have started to implement the necessary technical infrastructure and study the required social aspects to realize

³¹ <http://ptscientists.com/>

³² <https://www.nasa.gov/phonesat/>

Stephenson's Illustrated Primer. We have partnered with sociology scientists focusing on gender aspects for the research part. In November 2017 we have begun to conduct initial remote mentoring experiments under real conditions, first with sponsoring from Google as a Google Code-in mentoring organization at the end of 2017, with several hundreds of teenagers from all over the world being remotely mentored by a pool of 38 Catrobat mentors over a period of seven weeks, followed by the "Remote Mentor" project itself, with initial funding from the Internet Foundation Austria until the end of 2018. Because of Stephenson's book, Catrobat has focused from its very beginnings to aim at empowering female teenagers in less affluent regions such as rural areas in India, Tanzania, or Brazil. Teens have reacted very positively to these first remote mentoring experiments, and we plan to eventually integrate the remote mentoring features into our tools and services so that mentors and mentees will be anonymously and automatically matched on demand, internationally in all languages, in a large scale, long term, and continuously further developed way.

This paper is organized as follows: First, we emphasize two trends that have emerged in the last decade (both were important in developing our app Pocket Code): block-based and visual coding, and an increased use of mobile devices among our main target group (teenagers from 13 to 19 years). Second, the focus lies on the Catrobat project and the educational app Pocket Code. Third, we describe our planned next steps and subprojects that are being developed, followed by a summary and conclusion.

Computational Thinking Skills for All

Computational Thinking promotes the importance of coding and computer science activities, thus delivering concepts that are more applicable and highly essential to prepare teenagers for the future (Barnett *et al.*, 2017; Tetre *et al.* 2016). After 2006, there was a rapid increase in the number of published articles about learning to code (Wu and Wang, 2012). The ongoing movement of promoting coding through visual programming languages has its origin at that time.

Trend 1: Block-based and Visual Coding

In the last decade, a number of block-based visual programming tools, e.g., Scratch, have been introduced which should help teenagers to have an easier time when first practicing programming. These tools have all had very similar goals: they focus on younger learners, support novices in their first programming steps, they can be used in informal learning situations, and provide a visual/block-based programming language which allows teenagers to recognize blocks instead of recalling syntax (Tumlin, 2017). Unlike traditional programming languages, which require code statements and complex syntax rules, here graphical programming blocks are used that automatically snap together like Lego blocks when they make syntactical sense (Ford, 2009).

Another critical improvement of visual programming systems over classic text based programming languages is the fact that all elements of the programming environment and also the programming language itself, including the formula elements, are translated to the human language of the young users. Especially for human languages that are not written with the Latin alphabet, this is a huge advantage for users, as they are not used to think in English and very often have difficulties to even read Latin scripts. In case of developing countries, usually only a small percentage of the population understands English, in which most user interfaces are exclusively available, thus implicitly excluding a large part of the world's population. Localization of a software can revolutionize E-learning, resulting in more educated workforce and improved economy (Ghuman, 2017). Pocket Code supports localization and internationalization on the application level. The app's language and locale can be changed without changing the smartphone's interface language on the system level. Languages such as Sindhi and Pashto, which are yet to be supported by operating systems, can thus be seamlessly used by our users (Awwad, 2017). Catrobat shares this feature, which improves accessibility and inclusiveness to users from all regions of the world, e.g., with Scratch and Snap!, and this certainly contributes in a major way to the positive worldwide reception of these visual programming environments.

Thus, visual programming languages provide an easier start and a more engaging experience for teenagers. The ease of use, and simplicity of such programming environments enables young people

to become game makers, and by the seamless translation of their user interfaces, to collaborate with other users on a worldwide scale.

Trend 2: Smartphone Usage

With mobile games, more people can engage who were previously limited to use other platforms such as PCs or consoles. Further, children nowadays grow up with mobile devices and feel comfortable using them. Considering current prices, and the forecast of the user penetration of smartphones in Austria, France, Germany, and the United Kingdom from 2014 to 2021 (Statista Market Analytics, 2016), we can conclude that smartphones will be used significantly more by teenagers in the future than tablets, laptops, and desktop PCs. Smartphones and the use of apps are already a part of our culture and are changing the way in which many people, particularly teenagers, act in social situations. For most adolescents the smartphone performs several functions in their daily lives, and it contributes, e.g., to identity formation through self-presentation on the Internet. In addition, the smartphone is used a lot during spare time (most games are played in the evening (Verto Analytics, 2015)) or for just killing some time, e.g., when commuting. In addition, online games and mobile games play an important role in the daily lives of teenagers (Bevans, 2017). A recent study which examined American female players' experiences found that 65% of the Android users who play mobile games are women (Google and NewZoo, 2017).

Teenagers increasingly have mobile devices on their own, which enables them to creatively express themselves at any time and to use apps that bring their ideas and creations to life. With a more meaningful use of mobile devices, teenagers worldwide can acquire powerful knowledge that will make them into better problem solvers, thinkers, and learners.

Catrobat and the Pocket Code App

The Free Open Source Software (FOSS) non-profit project Catrobat³³ was initiated 2010 in Austria at Graz University of Technology. The multidisciplinary team develops free educational apps for teenagers and programming novices. The aim is to introduce young people to the world of coding (Slany, 2014). With a playful approach, teenagers of all genders can be engaged, and game development can be promoted with a focus on design and creativity. A first public version of our free app was published in 2014, with 47 releases of the main coding app as of July 2018. Our app currently has more than 700,000 users in 180 countries, is natively available in 50+ languages (including several languages not directly supported by the underlying operating system), and has been developed so far by almost 1,000 volunteers from around the world.

These volunteers are implementing software, designing educational resources, translating the app, or provide other services that help to advance the project. The contributors work together in a cooperative way, having the chance to engage in a field they like and create something within a community that follows the same shared vision. Besides attracting students, educators, and other interested contributors from all over the world, Catrobat also benefited from being part of Google's Summer of Code and Code-In initiatives. These initiatives promote open source worldwide and motivate teenagers and students to get involved in projects such as ours. Our openness towards international contributors helps us to represent different cultures, bring in various viewpoints, and generate new ideas how the project can further develop. Catrobat's project management as well as development is done in an agile way, allowing our contributors to adopt new technologies, respond to user feedback, and embrace upcoming ideas quickly. An example where a feature request issued by users was implemented is our web based automatic APK (Android Package) generation. It is currently being extended for users to be able to sign their apps and add AdMobs based ads, so that they can publish their Catrobat projects on Google Play and other app stores and earn money with them. Another example is of technical nature and affects our deployment workflow, which was redesigned and fully automated to enable us to deploy to Google Play including the localized screenshots and app descriptions in 47 languages (not all our languages are supported on the Google Play store) with only two mouse clicks. All this helps to provide

³³ <https://www.catrobat.org>

a motivating user experience for our young target group, support them in their learning process, and foster collaboration.

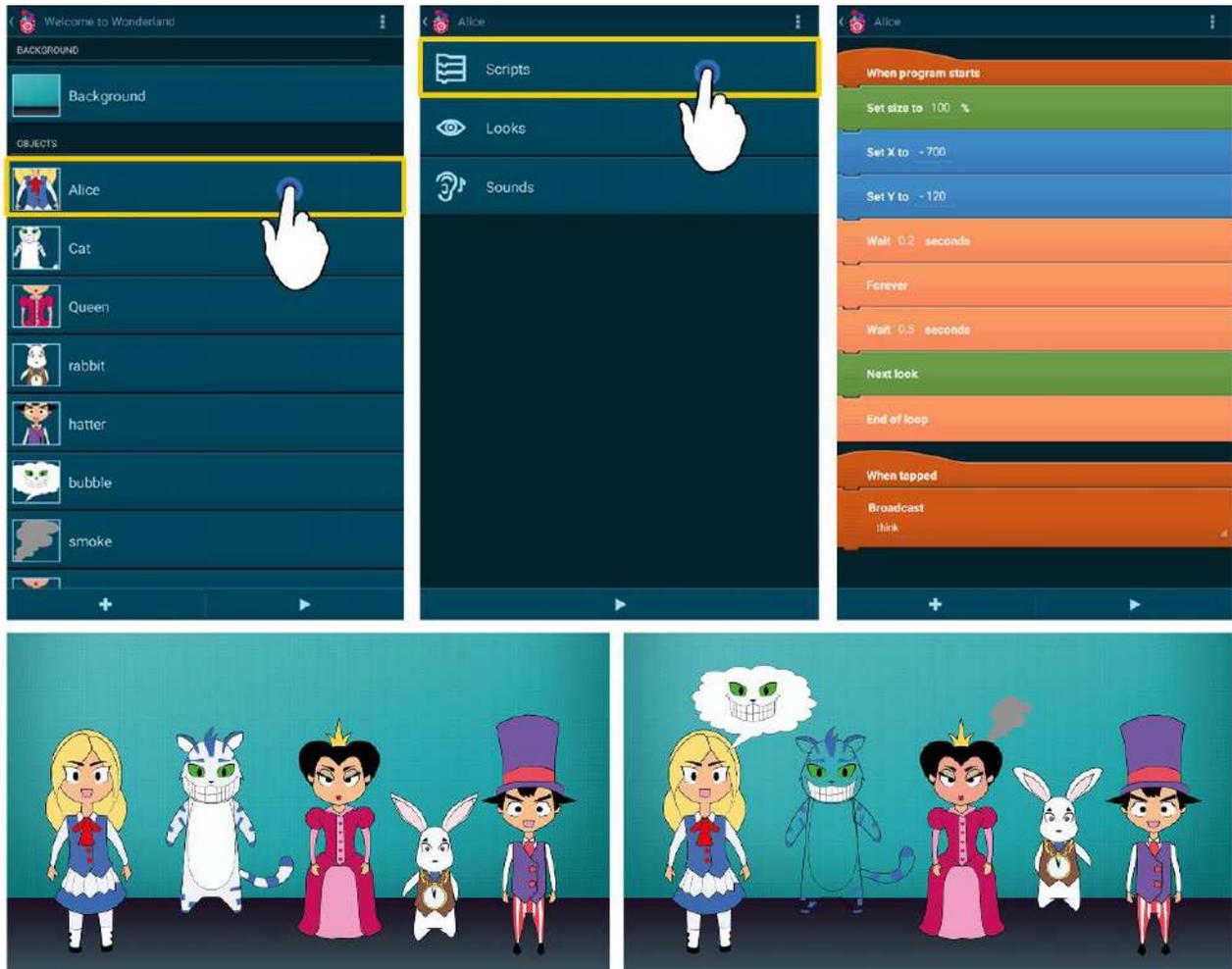


Figure 2: Pocket Code Alice themed program

Pocket Code: Creating your own Games

The app Pocket Code³⁴ is an Android-based visual programming language environment that allows the creation of games, stories, animations, and many types of other apps directly on phones or tablets, thereby teaching fundamental programming skills. This app consists of a visual Integrated Development Environment (IDE) and a programming language interpreter for the visual Catrobat programming language. The IDE automatically translates the underlying code parsed by the XML file into visual brick elements and vice versa. With the use of simple graphic blocks, users can create their own game, colorful animations, or extensive stories directly on the mobile phone without prior knowledge.

The drag and drop interface provides a variety of bricks that can be joined together to develop fully fledged programs. The app is freely available for Android on Google's Play Store and soon will be available on Apple's iTunes Store for iPhones. Figure 2 shows Pocket Codes' UI and an example project with "Alice in Wonderland" characters.

Pocket Code: the Mobile Integrated Coding Environment

Projects in Pocket Code follow a similar syntax to the one used in Scratch and are created by snapping together command bricks. They are arranged in scripts that can run in parallel, thereby allowing

³⁴ <https://catrob.at/pc>

concurrent execution. To communicate between objects, to trigger execution of scripts, or scripts beyond objects, broadcast messages are used. By means of this mechanism, sequential or parallel execution of scripts is possible, either within the same object or over object boundaries. In addition to the basic control structures, Pocket Code offers event triggering building blocks for event-driven programming. Familiar concepts, such as variables, lists, or Boolean logic, are included as well.

In addition to Scratch, Pocket Code has a 2D physics engine which enables the user to define certain physical features of objects and the stage (collision detection, velocity, gravity, mass, a bounce factor, and friction) to create from simple up to complex simulations of the real world. An example that showcases both the physics engine as well as the use of inclination sensors is a simulated wooden maze through which a metal ball needs to be navigated from a starting position towards the winning end position, all while avoiding a number of holes on the floor of the maze, by tilting one's phone, as if it were a real, physical wooden maze. The wooden walls initially execute a physics brick named "Set motion type" with the pull-down option "others bounce off it", and the ball executes the same "Set motion type" brick with the option "bouncing with gravity". The movement of the ball by tilting the phone is realized by a "Set gravity for all objects to X: $-3 \times \text{inclination}_x$ Y: $-3 \times \text{inclination}_y$ steps/second²" brick that is executed in a "Forever" loop. Voilà, that's all that is needed. To increase the realism of the simulated maze, there is an additional "When physical collision with anything" brick followed by a "Vibrate for 0.02 seconds". Even more, the ball's metallic reflection sheen is oriented always in the same direction using the magnetic compass sensor of the phone, thus giving the impression that the light always comes from the same side. It would be easy to also influence the gravity vector using the acceleration sensors built into the phones, which would make the ball not only react to tilt, but also to shaking and quick moves in any direction. The rest of the scripts handles the animations when the ball "falls" into one of the holes, and also shows the amount of holes that have successfully been avoided so far in the top left corner. Note that all objects that execute one of the two variants of the "Set motion type" brick mentioned above, automatically have their convex hull computed for the physics collisions, based on the visible parts of their current look. No other game engine to our knowledge does this automatically: in other gaming frameworks, the bouncing box is either quadratic, or has to be manually specified by a professional developer. Catrobat's use of convex contours makes it extremely intuitive and simple for the user to create complex games using the physics engine. Note that arbitrary 2D forms going beyond the convex hull, e.g., patterns of the form "U" or "8", are much more complex to handle automatically, and it is probably easier for users to handle special cases on an individual basis, e.g., by forcing several objects to move in synchronicity. In contrast, for all practical purposes it is impossible to realize a physically correct collision and bouncing of objects from each other for arbitrary shapes in other visual programming languages such as Scratch. This aligns with the "rock bottom" metaphor of Catrobat corresponding to the "low floor" metaphor of Logo and Scratch, making complex games easy and intuitive to realize, with a very low entry threshold, with physics being a concept everyone is intimately familiar with. At the same time, the maze project also has a resolution of 1920 x 1080 of which every pixel is fully used, giving it a very high-resolution and also highly realistic look, on par with professionally created game apps. This exemplifies Catrobat's "the sky" metaphor corresponding to the "high ceiling" of Logo and Scratch, since with Pocket Code, there literally is no upper limit in realism and performance of the created games: a project's resolution is only limited by the capabilities of the phone on which it is created, so, e.g., a project with a 3840 x 2160 pixels resolution can be created using a Sony Xperia XZ Premium smartphone. Figure 3 shows the stage as well as some of the scripts mentioned above. The project can be found under the name "Tilt maze 1.0" on Catrobat's sharing site from within Pocket Code.

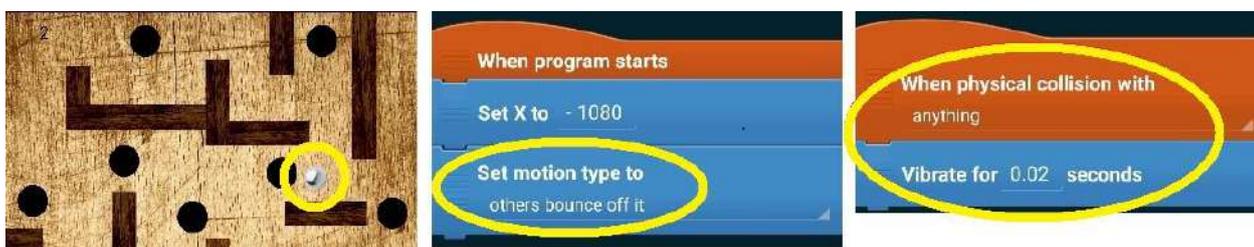


Figure 3: Partial screenshots from “Tilt maze 1.0” that relies on Pocket Code’s physics engine, where the physical behaviour of objects is set through its “motion type”, e.g., “others bounce off it” for the wooden walls of the maze (middle script), and a short vibration when the ball touches a wall (script on the right). The movement of the ball by tilting the phone is realized by a “Set gravity for all objects to X: -3 x inclination_x Y: -3 x inclination_y steps/second2” brick that is executed in a “Forever” loop (not shown here). Direct link to the “Tilt maze 1.0” project on the web version of Catrobat’s sharing site <https://catrob.at/TiltMaze>

With Pocket Code’s intuitive merge functionality, the new parts of two projects can be seamlessly merged together into one larger project – parts of the two initial projects that exist in both are not duplicated. This makes programming cooperatively much easier. Modern smartphones are equipped with a large number of sensors, although most mobile games only use few or none of them (Kafai and Vasudevan, 2015). Within Pocket Code, users can create games using the device’s sensors, such as inclination, acceleration, loudness, face detection, GPS location, or the compass direction, which makes user input easy and engaging. With Pocket Code it is also possible to connect via Bluetooth to Lego® Mindstorms robots or Arduino™ boards. The following extensions are available: Lego Mindstorms NXT/EV3, Parrot AR.Drone 2.0 and Parrot Jumping Sumo Drone, Arduino, Raspberry Pi (via WiFi), NFC tags, Phiro robots³⁵, and Chromecast. These kinds of computational construction kits make creating programmable hardware accessible to even novice designers and combines coding and crafting with a rich context for engaging teenagers (Kafai and Vasudevan, 2015). In the context of robots, being able to program a smartphone makes much more sense, as the smartphone can be mounted on the robots, thus allowing to give it a face, a voice and other sounds, and additional sensors such as acceleration, inclination, magnetic field, GPS, voice recognition, computer vision. Also, since only a smartphone is needed with Catrobat, the programming can be done on the spot, outside, e.g., when using one’s land-based robot or flying drone outdoors. With Catrobat no laptop or PC is necessary, thus, coding can take place anytime and anywhere, and in particular can be widely made available even in less affluent communities around the world. In addition, Catrobat released a Scratch Converter to allow the conversion of existing Scratch projects to the Catrobat language directly within the app, so there are, in fact, now more than 30 million projects available for remixing and inspiration to our users.

The Pocket Code interface consists of several very distinct areas: First, the app itself with a main menu and the collection of downloaded or developed-by-oneself projects, second a community sharing³⁶ platform, which is integrated into the app as a web-view, and which serves as a learning, sharing, remixing, cooperation, and publishing place, third the “stage” where projects get executed on the phone, and additionally a sophisticated graphical editing program that allows to draw and edit the looks of all actors, objects, and backgrounds of one’s projects. This community website provides an online platform for users to download and upload programs, share them with other users, search for programs, and to provide feedback, e.g., write a comment to a project or rate a project. In addition, tutorials and starter projects are provided. In the community website’s project overview, users can execute the project directly in desktop browsers (HTML5 web player), download the project to the Pocket Code app, or download the project as a standalone Android app. In addition, the tool automatically creates statistics from Pocket Code projects and provides an online code overview. The project details page is illustrated in Figure 4. Figure 5 illustrates the options within the main menu.

³⁵ <https://catrob.at/Phiro>

³⁶ <https://share.catrob.at>

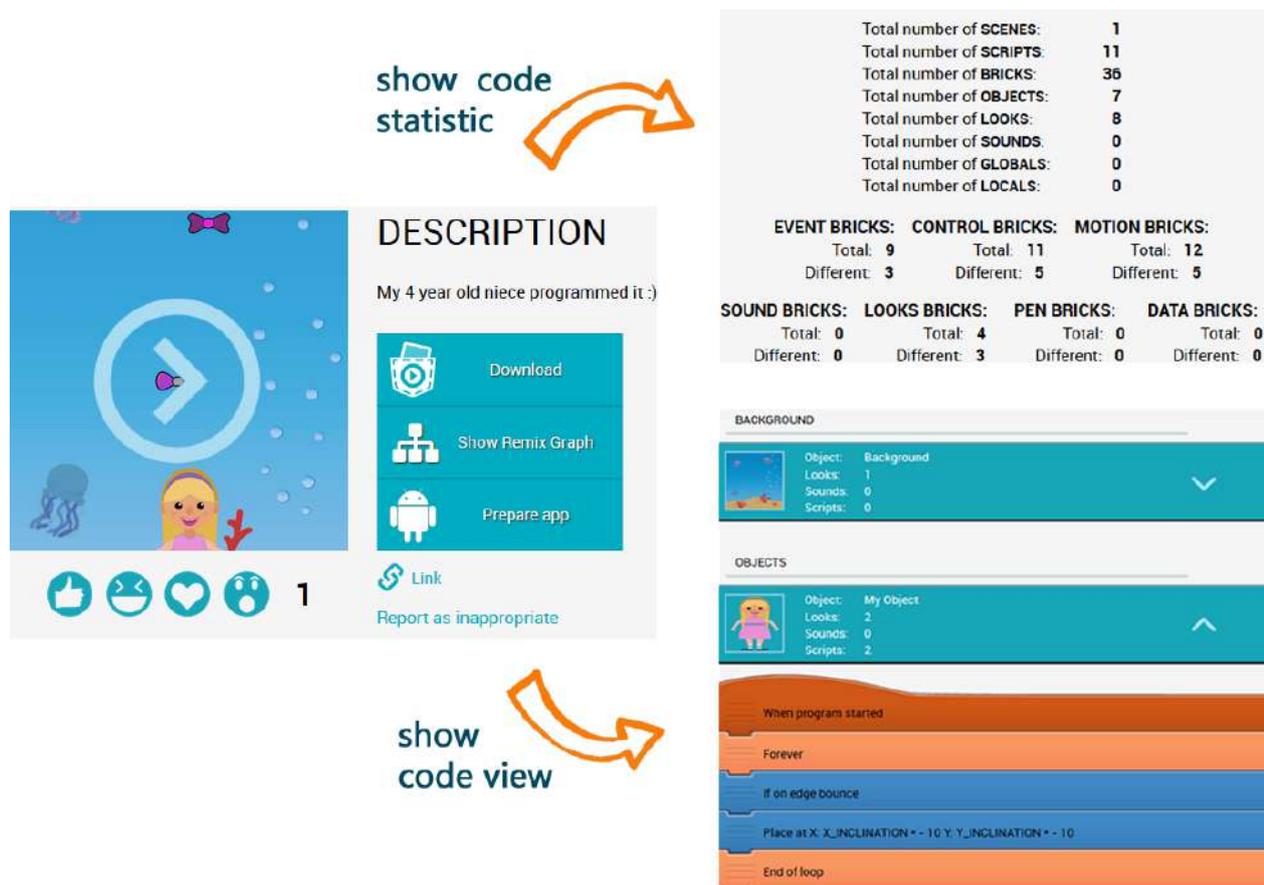


Figure 4: Pocket Code web share: project details page with code statistic and code view

If the user starts first with a new and empty program, it initially only contains one empty background object. With the “+” sign users are able to add objects, looks, or sounds (depending on which activity he or she is in). The background object itself can be assigned to several backgrounds, which can be exchanged during runtime. The background can also have its own scripts. Every project can consist of multiple objects and at least one background (which is a special kind of object). Every object can hold a.) scripts that define the behavior of the object, b.) looks which can be changed and used, e.g., for object animation, and c.) sounds to make the object play music, other sounds, or recorded speech. Scripts can control the looks and sounds. Looks can be drawn and edited with Pocket Paint. Pocket Paint³⁷ is a second app of Catrobat available on Google Play, which allows users to create their own objects with a pencil or different shapes (note that we currently work on integrating this second app completely in Pocket Code in order to simplify the installation for our users). Distinctive features of Pocket Paint include the ability to use transparency, to zoom in up to pixel level, to change the dimensions of the looks, and to use layers, the latter being particularly interesting to create consecutive looks from an animation series. In addition, users can add looks with their camera, from their phone’s memory, or use Catrobat’s Media Library with a collection of predefined graphics. To add a new sound the user can either record a sound directly in Pocket Code, add a sound from the Catrobat Media Library, or add a sound from the phone’s memory. This workflow is illustrated in Figure 6.

³⁷ <https://catrob.at/PPoGP>



Figure 5: Pocket Code main menu: within the settings menu you can find, e.g., the accessibility preferences or the Scratch Converter; 2) create a new project by starting with an example game or with an empty game; 3) project overview: tap on one to execute or modify it; 4) find help: videos, tutorials, step-by-step tutorials, education page for teachers and students or google groups forum; 5) download and play games from other users; 6) upload your game to the sharing platform.

A script is a collection of code blocks that contain the logic of programming and define the operations of the object. Thus, it is possible to move the object and access its properties and change them. For adding scripts there are seven different brick categories (see Figure 7): a.) The *Event* category in dark orange that contains hat-bricks or broadcast bricks. Hat bricks are special kinds of bricks that, depending on certain circumstances such as a tap on an object, start the attached script; b.) The *Control* category in orange contains if-then-else bricks, loop-bricks to control the flow of the script, bricks to switch between scenes, clone bricks, etc; c.) The *Motion* category in blue color contains bricks to manipulate the object’s position, orientation, or movements; d.) The *Sound* category in purple contains bricks to start and stop sounds, manipulate the volume, or accept spoken input; e.) the *Looks* category in green contains bricks to change the graphical appearance of the object, e.g., set/change size, brightness, transparency or hide/show the object as well as set a certain look to animate the object, to show speech and think bubbles, or to ask for written user input; f.) The category *Pen* in dark green holds bricks for drawing lines (a pen that follows the object) and the option to leave stamped marks of the object on the background, g.) The *Data* category in red contains bricks to manipulate variables and lists, e.g., to set/change variables, maintain lists, add/insert/replace items, and show variable content on the stage. This color scheme makes it possible to understand scripts more easily through of the bricks’ color which supports readability. By activating extensions in the settings menu, additional categories appear for Lego (yellow), Drone (brown), Arduino, the Phiro robot (both in cyan), etc.

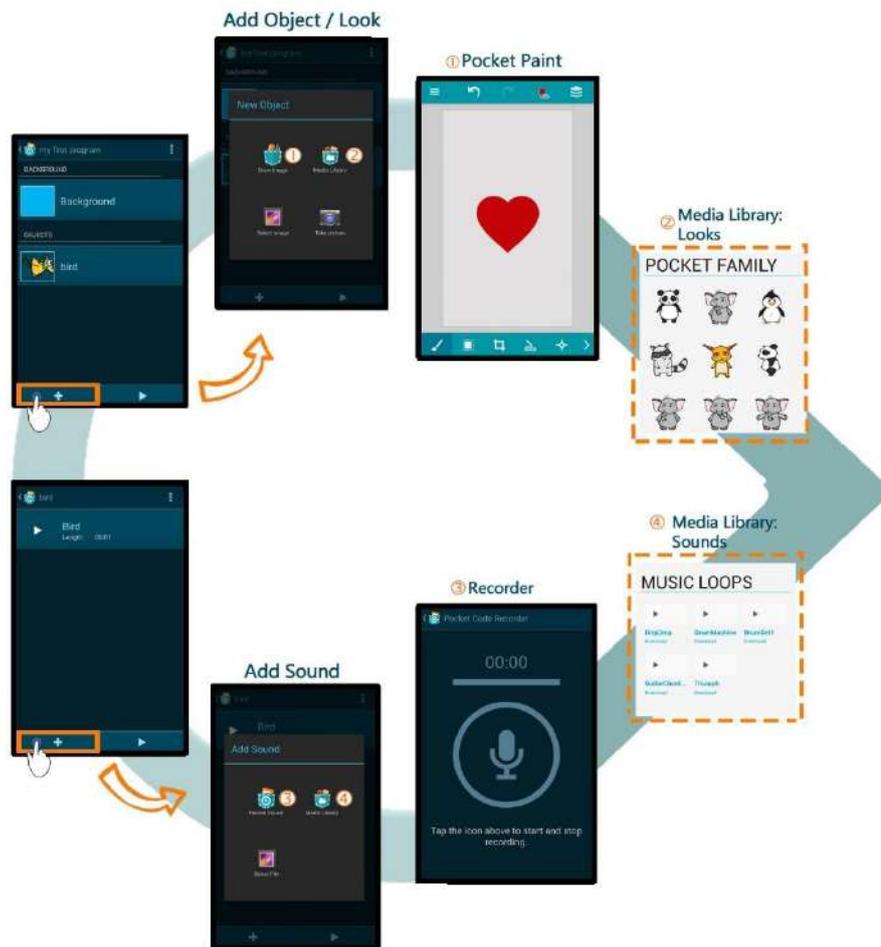


Figure 6: Pocket Code's UI: add a look/object or add a sound with the "+" sign.

In contrast to Scratch and Snap!, Pocket Code does not use bricks for formulas. Instead, there is a **formula editor** that looks like a calculator and allows the creation and execution of mathematical and logical formulas that can be used in bricks. In (Harzi et al. 2013) we compared of blocks-based (like in Scratch and Snap!), text-based (like with traditional programming languages such as Python), and hybrid ways (such as Pocket Code) to enter and edit formulas, and showed that teenagers can create and debug complex formulas using a hybrid editing mode faster and with less errors than with the blocks-based as well as purely text based approaches. All advantages of block-based interaction modes, such as easy discoverability of features, translation into many human languages, avoidance of syntax errors, and immediate feedback about current values (including dynamic sensor values), are not only present in Pocket Code's formula editor, but are additionally enhanced by the familiarity of how formulas are written, changed, and read by users in other contexts, as well as by the familiar interface of an electronic pocket calculator (app). Pocket Code's formula editor is shown in Figure 8. It consists of an input field to show and compose the formula, a keyboard, and a compute button to display the current result. On the keyboard, five categories for various values, functions, and operators are available. a) *Object*: a collection of values of the current object, e.g., values for the X- and Y-coordinate, or the current speed, b) *Functions*, such as sin or cos, a random number generator, or list and string functions, c) *Logic* is used to compare values or to combine logical expressions, d) in *Device* there is information that the smartphone or tablet records, e.g., inclination, loudness, or GPS data, and e) *Data* stores created variables and lists and shows their last value.

With a tap on the play button the program starts. The objects are shown on **the stage** and the scripts are executed. To stop or to pause the program, the user has to tap on the back button of the phone. A stage menu appears which can be seen in Figure 9. The stage is organized in a logical coordinate

system with an X- and Y-axis, which allows an exact positioning of the objects. This axis can be displayed in the stage menu (see Figure 9c).



Figure 7: Script categories: choose bricks from the seven basic available categories.



Figure 8: Formula editor: the value for the direction can be defined as a constant or, e.g., a sensor can be chosen by tapping on "Device".

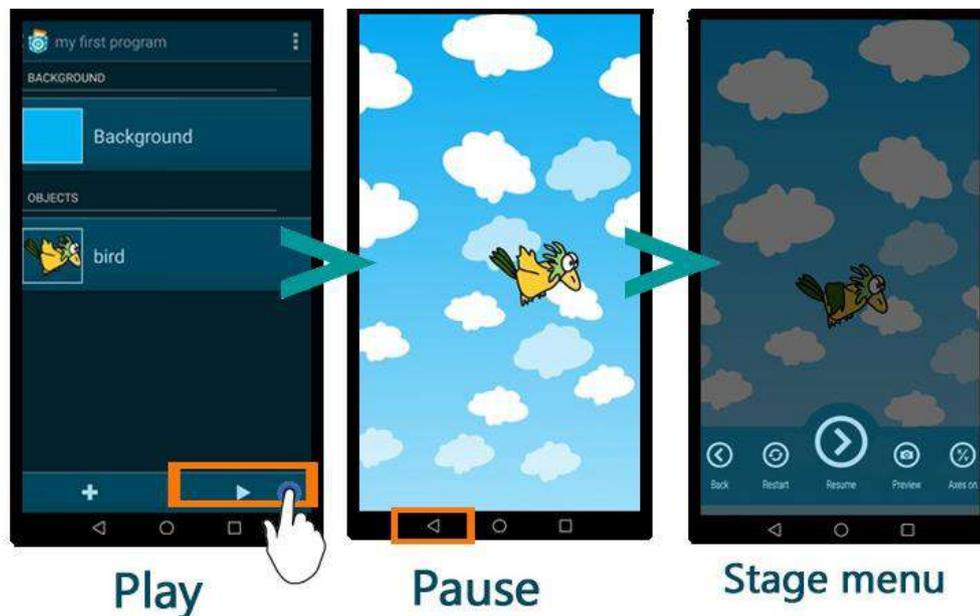


Figure 9: Stage; a.) tap the play button to start the program, b.) tap the back button of the phone to pause the game, c.) in the stage menu the user has five options: 1. tap back again to stop the game and switch back to editing of project, 2. restart the game, 3. resume the game, 4. add a new preview picture to the project (this will be shown, e.g., on the sharing platform), and 5. display the x/y axes on the device screen.

Projects and Further Work

This new and forward-thinking approach to code on mobile devices received national and international recognition. The Catrobat project has won a number of awards, including 2016 two Lovie Awards ex aequo with Red Bull and Doctors without Borders, evaluating the best European digital projects in London, and the Reimagine Education Award for innovative educational projects at the Wharton

Business School of Pennsylvania³⁸. Additionally, in March 2017, Catrobat won the “Platinum Award” in Best Mobile App Awards Best Educational App category and 2016 the “Internet for Refugees”³⁹ award for a Right-to-Left language implementation of Pocket Code, which supports several RTL languages, e.g., Arabic or Farsi, and particularly focuses on refugees and teenagers in crisis or development areas. A new project was started in January 2018 which promotes remote mentoring by connecting female role models with female programming beginners. This idea was awarded with the “Closing the Gender Gap” prize of the Austrian Netldee in November 2017. During the European H2020 project “No One Left Behind” (NOLB), the team developed a special flavoured school version of the app with the name “Create@School”. This version compromises, e.g., the gathering of analytics data for visualization, the integration of accessibility preferences, and the development of pre-coded templates. Currently a new flavored version customized for female teenagers is in development. This version with the name “Luna&Cat” promotes special content for girls, like featured and user-contributed programs, media assets, and tutorial videos.

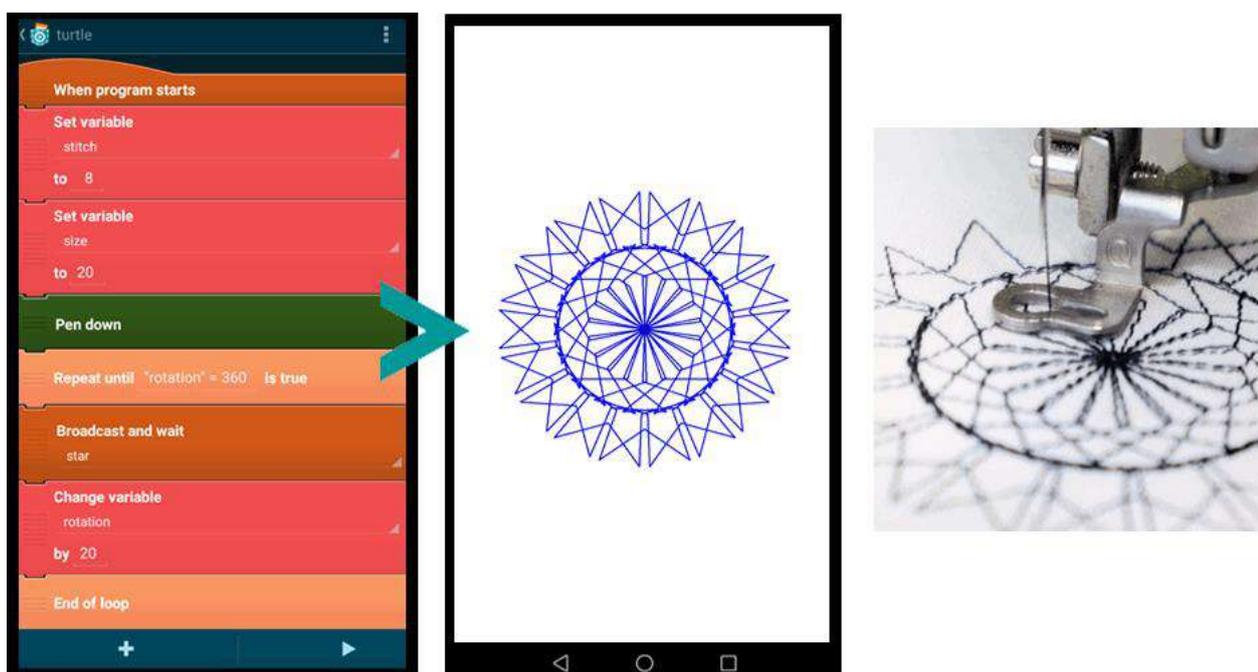


Figure 10: “Stitched” patterns in Pocket Code. Picture on the right with kind permission from Andrea Mayr-Stalder, www.TurtleStitch.org project.

One new feature which is currently under development is an extension to program embroidery machines. Once available, self-created patterns and designs can be stitched on t-shirts, pants, or even bags or shoes. With Pocket Code, the embroidery machines will be programmable, similar to the existing TurtleStitch⁴⁰ project, which realizes this concept on a PC (while with Pocket Code only a smartphone is needed). As a result, teenagers have something they can be proud of, something they can wear, and they can show to others. This feature has proven to be especially engaging for female teenagers and shows them new ways of expressing themselves creatively through coding. Figure 10 shows an example of an embroidery pattern made with Pocket Code.

Another new beta feature allows registered users to sign and release the Catrobat project as apps on Google Play. Users optionally also add mobile ads to earn money. In developing countries, mobile technologies are playing an important role in developing economies (Alderete, 2017). This is because mobile phones and the mobile internet require considerably fewer financial resources in comparison to a traditional desktops and laptops (Stork, 2013). Due to lack of alternate employment opportunities in

³⁸ <http://www.reimagine-education.com/awards/reimagine-education-2016-honours-list/>

³⁹ <http://www.tugraz.at/en/tu-graz/services/news-stories/tu-graz-news/singleview/article/preis-internet-for-refugees-fuer-programmier-app-der-tu-graz>

⁴⁰ <http://www.turtlestitch.org/>

developing countries, the low cost of investment is a critical enabling factor for new entrepreneurs (Alderete, 2017). The economic advantage of releasing an app on Google Play or integrating AdMob within apps can motivate many people to learn programming and solve issues digitally. This is especially beneficial for under-privileged user groups who have access to limited resources like computers and continuously available electricity. They can benefit by sharing their creativity with others and serve a global market with minimal resources such as a low-cost smartphone and mobile internet, which are increasingly available everywhere. Figure 11 shows an example of apps with AdMob integration.

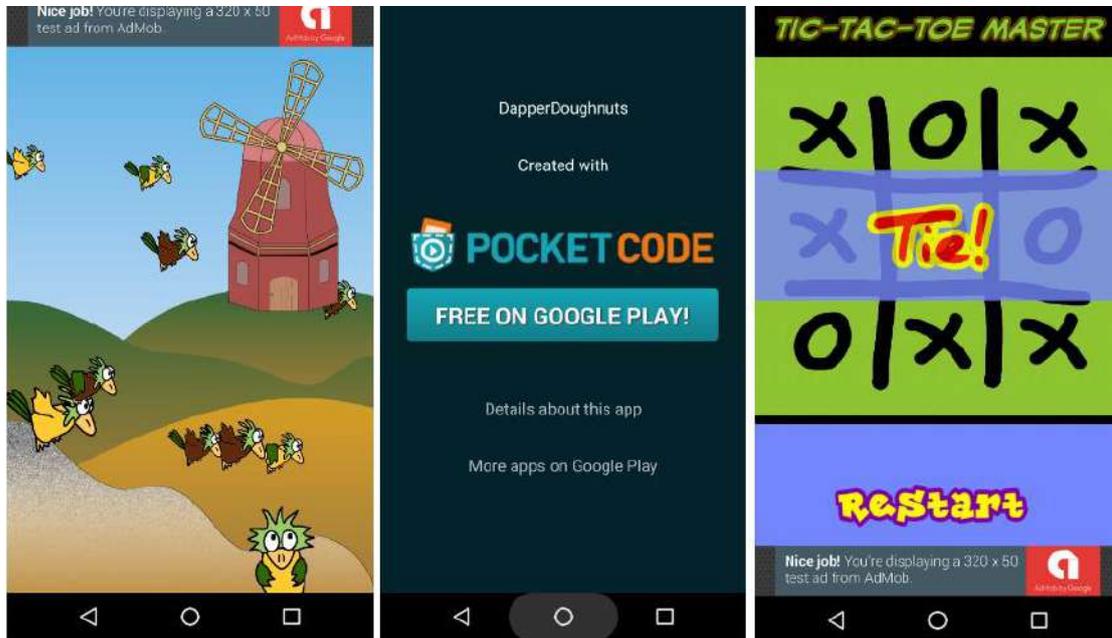


Figure 11: Android apps with AdMob banner advertisement created with Pocket Code.

Conclusion and Discussion

The aim of this paper was to provide an overview about the Catrobat project and the Pocket Code app as of July 2018. Pocket Code provides an easy way to start coding. It is not intended to allow the development of standard applications, but to promote understanding of the logic behind coding and foster computational thinking skills, thus following a constructionist approach in learning by doing and the creation of sharable artefacts. Creative and artistic talents can be recognized and learning can occur in a user-centered, project-based setting with the use of new media. Users of Pocket Code are mostly teenagers who can learn from each other and share their ideas to create new games and other apps together. The community sharing platform allows users to give and receive feedback, support, and assistance from others around the world, thus allowing our users to stand on the shoulders of their peers and learn from each other. They can try out new ideas and realize the projects they define for themselves, aided and inspired by likeminded others in a user-friendly and social environment. Catrobat fosters diversity and learning in a worldwide community. Our goal is to empower teenager all over the world to realize their potential and express themselves creatively with today's and any future technology.

References

- Alderete, M. V. (2017) Mobile Broadband: A Key Enabling Technology for Entrepreneurship? *Journal of Small Business Management*, 55(2), 254-269.
- Awwad, A. , Schindler, C., Luhana, K. K., Ali, Z., and Spieler, B. (2017) Improving Pocket Paint usability via material design compliance and internationalization & localization support on application level. *In Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (ACM)*, p. 99:1-99:8.

- Balanskat, A., and Engelhardt, K. (2015) Computing our future. Computer programming and coding Priorities, school curricula and initiatives across Europe. *European Schoolnet (EUN Partnership AIBSL)*.
- Barnett, T., Lawless, B., Kim, H., and Vista A. (2017) Complementary strategies for teaching collaboration and critical thinking skills. *Education Plus Development*. [online] <https://www.brookings.edu/blog/education-plus-development/2017/12/12/complementary-strategies-for-teaching-collaboration-and-critical-thinking-skills/>, accessed: 3.4.2018.
- Bevans, A. (2017) Who plays mobile games? [online] <https://www.gamesindustry.biz/articles/2017-06-14-who-plays-mobile-games>, accessed: 6.4.2018.
- European Commission (2016a) Women in digital - a gap and an opportunity. [online] <https://ec.europa.eu/digital-single-market/en/blog/Women-digital-gap-and-opportunity>, accessed: 8.4.2018.
- European Commission (2016b) A new skills agenda for Europe. Working together to strengthen human capital, employability and competitiveness, [online] <http://ec.europa.eu/social/main.jsp?catId=1223>, accessed: 1.3.2018.
- Ford J.L. (2009) Scratch programming for Teens. *In Computer Science Books*.
- Ghuman, A., Mahajan, J., Bhatia, S., Singh, J., & Kulkarni, M. D. (2017, August). Empowering e-learning with localization. *In Proceedings of the 5th IEEE National Conference on E-Learning & E-Learning Technologies (ELELTECH), 2017*, p. 1-6.
- Google and NewZoo (2017) Change the Game. THE WORLD OF WOMEN AND MOBILE GAMING. A White Paper. [online] http://services.google.com/fh/files/misc/changethegame_white_paper.pdf, accessed: 8.3.2018.
- Greenberger, M. (1962) Computers in the World of the Future. *In Cambridge, MA: MIT Press*.
- Harzl, A., Krnjic, V., Schreiner, F., and Slany, W. (2013) Comparing Purely Visual with Hybrid Visual/Textual Manipulation of Complex Formula on Smartphones. *In Proceedings of the 19th International Conference on Distributed Multimedia Systems (DMS 2013)*, p. 198-201.
- Jingli, S. (2017) China's mobile games market posts \$15b revenue in 2017. [online] <http://www.chinadaily.com.cn/a/201801/12/WS5a5851a0a3102c394518edcc.html>, accessed: 7.4.2018.
- Kafai, Y., and Vasudevan, V. (2015) Hi-Lo tech games: crafting, coding and collaboration of augmented board games by high school youth. *In Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*, p. 130-139.
- Kahn, K. (2017) A half-century perspective on Computational Thinking. *In tecnologias, sociedadee conhecimento*, Vol. 4, No. 1.
- Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., and Settle, A. (2014) Computational thinking in K-9 education. *In Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, p. 1–29.
- National Center for Women Information Technology (2015) NCWIT Fact Sheet, [online] <https://www.ncwit.org/ncwit-fact-sheet>, accessed: 1.4.2018.
- National Center for Women Information Technology (2017) NCWIT's Women in IT: By the Numbers presents the most compelling statistics on women's participation in IT on a single page. [online] www.ncwit.org/bythenumbers, accessed: 1.4.2018.
- Papert, S. (1985) Mindstorms. Children, Computer, and Powerful Ideas. *In Basic Books Inc*.
- Piaget, J. and Inhelder, B. (1967) A Child's Conception of Space (F. J. Langdon & J. L. Lunzer, Trans.). *New York: Norton (Original work published 1948)*, p. 375-418.
- Resnick, M. (2017) Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play (MIT Press).

- Ryan, R., and Deci, E. (2000) Intrinsic and extrinsic motivations: Classic definitions and new directions. *In Contemporary Educational Psychology*, Vol. 25, No. 1, p. 54-67.
- Slany, W. (2014) Tinkering with Pocket Code, a Scratch-like programming app for your smartphone. *In Proceedings: Constructionism 2014*, Vienna, August 2014.
- Statista Market Analytics (2016) Forecast of the smartphone user penetration rate in Austria, France, Germany, and United Kingdom (UK) from 2014 to 2021. [online] <http://www.statista.com/statistics/567976/predicted-smartphone-user-penetration-rate-in-austria>
<http://www.statista.com/statistics/568093/predicted-smartphone-user-penetration-rate-in-france/>
<https://www.statista.com/statistics/568095/predicted-smartphone-user-penetration-rate-in-germany/>
<https://www.statista.com/statistics/553707/predicted-smartphone-user-penetration-rate-in-the-united-kingdom-uk/>, accessed: 8.4.2018.
- Stephenson, N. (1995) *The Diamond Age: Or, A Young Lady's Illustrated Primer: a Propædeutic Enchiridion in which is told the tale of Princess Nell and her various friends, kin, associates, &c.* (Bantam Spectra).
- Stork, C., Calandro, E., and Gillwald, A. (2013) Internet Going Mobile: Internet Access and Use in 11 African Countries, *Info* 15(5), 34–51.
- Takahashi, D. (2017) Sensor Tower: Mobile game revenues grew 32% in Q2 as Asian titles surged. [online] <https://venturebeat.com/2017/07/24/mobile-game-revenues-grew-32-in-q2-as-asian-titles-surged/>, accessed: 8.4.2018.
- Tedre, M., and Denning, P.J. (2016) The Long Quest for Computational Thinking. *In Proceedings of the 16th Koli Calling Conference on Computing Education Research*, November, Koli, Finland, p. 120-129.
- Tumlin, N. (2017) Teacher Configurable Coding Challenges for Block Languages. *In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, p. 783-784.
- Verto Analytics (2015) Who Plays Mobile Games and When? [online] <http://www.vertoanalytics.com/who-plays-mobile-games-and-when/>, accessed: 7.4.2018.
- Wing, J. (2006) Computational thinking. *In Communications of the ACM*. Vol. 49, No. 3, p. 33–35.
- Wu, B., and Wang, A.I. (2012) A Guideline for Game Development-Based Learning: A Literature Review. *In: International Journal of Computer Games Technology*, Vol. 2012, No. 103710, 20 pages.

Making Constructionism Great Again

Gary S. Stager, gary@stager.org
Constructing Modern Knowledge, USA

Abstract

The Constructionism community is at a crossroads with the passing of Seymour Papert, the uncertain future of Logo, and the emergence of simpatico movements outside of the academy. This session will explore unfinished work in addition to the challenges and opportunities faced by the next generation of constructionists. What will it take to sustain its relevance and make constructionism great again?

Topics explored include:

- The Premature Death of Logo
- Seeing the “Entire Elephant”
- Piaget versus Popular Coding Curricula
- Twenty Things to Do with a Computer Today
- Agency and the Lost Art of Teaching
- Know Who Your Friends Are
- The Progressive Imperative



Keywords

Progressive education, constructionism, Logo, Seymour Papert, Scratch, physical computing, CS4All

Turning Theory into Practice – Spreading Constructionism

Gary S. Stager, gary@stager.org
Constructing Modern Knowledge, USA

Sylvia Martinez, Sylvia@inventtolearn.com
Constructing Modern Knowledge, USA

Abstract

In age marked by ascendant instructionism, the presenters have led two initiatives that introduced constructionism successfully to preschool – high school educators around the world. Without compromising the powerful ideas of Papert or his learning theory, the presenters have made constructionism accessible and resonant among practicing educators through publishing and professional development efforts. Countless educators have been inspired bring constructionism to life, often accompanied by the use of cutting-edge technology, without government, foundation, or academic support.

Five years ago, the presenters published *Invent To Learn: Making, Tinkering, and Engineering in the Classroom*. This book is quite possibly the most popular text ever written about constructionism and has been translated into multiple languages. *Invent To Learn* sought to situate the emerging maker movement in a theoretical context of constructionism and historical context of progressive education while building a bridge between the informal learning movement outside of schools and sound classroom practice. Constructionism pervades the text explicitly and tacitly. The success of *Invent To Learn* led to the publication of ten other books by constructionist educators. During this plenary session, the authors will reflect upon lessons learned about learning-by-making, teaching, and school change since the time of publication.

Nearly twenty years ago, Gary Stager and Seymour Papert engaged in multiple conversations about building a different bridge; one between our progressive education colleagues suspicious of modernity and an educational technology community that, in Papertian terms was increasingly “idea averse.” While Papert was never able to convene such a summit, Gary Stager created the annual Constructing Modern Knowledge summer educator institute. Over eleven years, Constructing Modern Knowledge has created an immersive learning environment modeling constructionism and pedagogical strategies developed collaborative with Seymour Papert during their “prison project.” Papert’s unique emphasis on the competence of educators, the absence of coercion, computer as material, powerful ideas, technology as prosthetic, project-based learning, and the centrality of the learner – especially when the learner is a teacher - create the conditions for countless educators to not only develop exceptional computational fluency, but construct personal lessons for creating productive contexts for learning in their personal school contexts.

At the Constructionism 2018 Conference, we will share the unique structure of Constructing Modern Knowledge, along with learning stories and project vignettes supporting the efficacy of constructionism by and for educators willing to take off their teacher hats and put on their learner hats.

Keywords

Constructionism; Logo; Seymour Papert; maker movement; coding; fabrication; professional development; progressive education

The Evolution of a Constructionist Teacher (with Some Reminders from Seymour)

Carol Sperry Suziedelis

Millersville University, USA

Abstract

Many teachers who are aligned with the organic, creative, and dynamic ideas of Constructionism find it difficult to navigate in traditional waters. The obstacles are many: rigid curriculum, excessive testing, lack of resources, few allies, inability to articulate philosophy to the satisfaction of the powers that be. It takes courage, and this paper attempts, through anecdote and narrative, to offer ways and means to develop a Constructionist mindset and the tools and attitudes to effect changes. We hope to inspire discussion of topics such as what it means to learn, thinking about thinking, the importance of teacher engagement, relationships, relevancy, aesthetics, gender issues, and project-based learning.

Along the way, we will remember Seymour Papert, his dedication to the possibilities of “learning as a dimension of life,” and resurrect some of the ideas” he used to inspire us.

Keyword

Constructionist teacher; dynamic

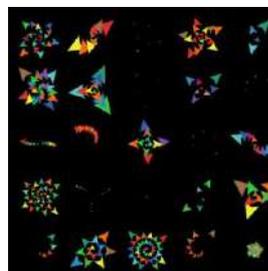
Reempowering Powerful Ideas

Uri Wilensky, uri@northwestern.edu
Northwestern University, Chicago, IL, USA

Abstract

In *Mindstorms*, Papert expressed many ideas foundational to Constructionism. Among these are how children programming can be transformative for learning, the empowerment children gain by creative expression with the computer, and powerful ideas of computation. In the nearly four decades since *Mindstorms*, there has been great progress in realizing these potentials. In particular, there has been wide recognition of the importance of coding and of computational thinking for all and many initiatives have arisen to galvanize these efforts. These new efforts are exciting for constructionists and we have seen public and academic vindication for these ideas.

In this paper, I point out that, in addition to these ideas, there is a part of Papert's vision in *Mindstorms*, that is more neglected. That is the more general notion of powerful ideas, not solely ideas inherent in computation such as variables, procedures and recursion, but also powerful ways of thinking from other disciplines that are made more powerful and accessible through computation. *Mindstorms* expounded on turtle geometry, making it accessible both through programming the turtle and engaging with turtle microworlds. In either mode, the turtle added to the Euclidean point one other property, that of a heading, and therefore angular velocity. This change results in a new definition of a circle, one that is intrinsic and so connects geometry to the powerful ideas of calculus. This change in the Euclidean point is a case of what Wilensky and Papert called a restructuration, a change in the representational infrastructure used to encode knowledge. Restructurations throughout history have increased the power, usability and learnability of formerly difficult knowledge. Classic examples are the restructurations of arithmetic from Roman to Hindu-Arabic representation and the restructuration of kinematics from natural language to algebra. Both of these dramatically democratized access to these powerful ideas. Wilensky & Papert argued that, like in turtle geometry, computational representations can serve as the basis for significant restructurations -- restructurations that increase the power and learnability of powerful ideas in science. Like the turtle, a powerful means of creating restructurations is to add agency to primitive elements. This can be achieved through agent-based computational approaches. In this paper, I'm going to show examples of the many different restructurations that members of the CCL lab have constructed in the past decade including NetLogo-based restructurations of powerful ideas of biology, materials science and economics -- and invite the constructionist community to increase its efforts in creating computational representations of powerful ideas. These ideas are accessed through multi-turtle programming and emergent systems microworlds. Agent-based representations provide "objects-to-think with" that facilitate powerful ideas of discrete mathematics, probability and network theory. As the world increases in complexity, citizens of a society increasingly require use of these powerful ideas to make sense of their natural and social worlds and to be empowered to make meaningful changes in society.



Agent-based models of a) artificial selection of sunflowers b) predator-prey ecosystem

Keywords

powerful ideas; restructurations; agent-based modeling, NetLogo

Research papers

Agent-based Construction (a-b-c) Interviews: A Generative Case Study

Ümit Aslan, umitaslan@u.northwestern.edu
Learning Sciences, Northwestern University, USA

Uri Wilensky, uri@northwestern.edu
Learning Sciences, Computer Science and Complex Systems, Northwestern University, USA

Abstract

We propose agent-based construction (a-b-c) interviews as a new research methodology specifically designed to expose patterns of reasoning about emergent phenomena and complex systems. In an a-b-c interview, the researcher acts as an active mediator between the participant and an agent-based modeling environment. As the participant describes the model, the researcher tries to write the corresponding code and probes the participant about his or her reasoning. In this paper, we present a generative case study in which an adult participant constructs a NetLogo model of aging with the help of a researcher. We conduct a preliminary grounded analysis of this case study and trace the evolution of the participant's model throughout the hour-long interview. Our findings show that the act of mediation between the participant and the agent-based modeling environment can potentially afford, at times even obligate, the researcher to continuously make on-the-fly hypotheses about the participant's thinking, present these hypotheses through writing the model's code, and get immediate feedback from the participant. Our findings also show that a-b-c interviews can potentially expose more fine-grained, spontaneous, and connected reasoning processes that cannot easily be studied through traditional task-based or verbal clinical interviews.

Keywords

agent-based modelling; complex systems; emergence; knowledge; reasoning; research methodologies; clinical interviews

Introduction

Many of the world's pressing issues can be conceptualized as *emergent phenomena*. That is, they are macro level observable patterns emerging from micro level interactions between numerous individual entities (Johnson, 2006; Mitchell, 2009; Wilensky, 2001). Some examples are climate change, migration, and epidemics (United Nations, 2016). Research suggests that learning about complex systems and developing relevant reasoning skills would greatly benefit those who engage with the world's pressing issues in any formal or informal manner such as democratic participation, personal choices, or policy making (e.g., NRC, 2012; Sterman, 1994). However, studies have shown that it is difficult for most people to make sense of emergent phenomena, and the complex systems such phenomena arise from, because they are non-linear, non-deterministic, decentralized, and multi-leveled (e.g., Chi, 2005; Wilensky and Resnick, 1999). More importantly, even though lots of research has been done, there is controversy about the nature of reasoning processes that cause such difficulties (e.g., Chi and Roscoe, 2002; Jacobson, 2001; Hmelo-silver, Marathe and Liu, 2007; Levy and Wilensky, 2008; Penner, 2000; Sengupta and Wilensky, 2011).

Almost all of the studies on reasoning about emergent phenomena and complex systems rely on clinical interviews focusing on relatively simple, well-understood emergent phenomena such as diffusion of liquids (Chi, 2005) or ants foraging for food (Jacobson, 2001). During a clinical interview, a participant is presented with tasks, cases, or verbal questions and is expected to produce verbal answers (Clement, 2000; Ginsburg, 1997). We argue that the expectation of verbal articulation might be a limiting factor when it comes to emergent phenomena because it may be too difficult to form coherent on-the-fly explanations about phenomena or systems that include many actors, interactions, levels, and

stochasticity. It may be necessary to support participants with an infrastructure that helps offload some of the more difficult aspects such as randomness and hypothesis testing.

We propose a new research methodology that emerged from our previous research on learning and thinking about emergent phenomena through agent-based modeling (Aslan and Wilensky, 2016a; 2016b). We tentatively call this approach agent-based-construction (a-b-c) interviews. As the name suggests, a-b-c interviews ask for participants to construct an agent-based model of an emergent phenomenon. A-b-c interviews incorporate agent-based modeling because it has been shown to be a particularly powerful methodology in learning and reasoning about emergent phenomena (e.g., Klopfer, 2003; Wilensky, 2001; Wilensky and Reisman, 2006; Wilensky and Papert, 2010; Wilkerson-Jerde and Wilensky, 2015). We hypothesize that observing people's reasoning as they are trying to construct an emergent phenomenon through agent-based modeling can offer us insights that may otherwise not be possible through traditional clinical interviewing. We also hypothesize that a productive avenue would be for a researcher to act as an active mediator between the participant and the agent-based modeling environment. This way, it would be possible to work with participants who do not have any prior experience in computation or agent-based modeling. It would also enable the researcher to observe and model the participants' reasoning on a moment-by-moment basis (Sherin, Krakowski and Lee, 2012).

In this paper, we attempt to formulate a working definition of a-b-c interviews through presenting a *generative case study* with an adult participant who constructed a model of aging with the help of a researcher over the course of an hour. We analyze the interaction between the participant, the researcher, and the agent-based modeling environment and reconstruct a timeline of the evolution of the participant's model. We discuss the results of our analysis and determine main features, as well as major challenges, of a-b-c interviews.

Theoretical underpinnings

Our proposal of a-b-c interviews as a research methodology is founded on two paradigm shifts in the way we study knowledge and reasoning, and two paradigm shifts in the way we make sense of the world around us: (1) the theory of constructivism, (2) the methodology of clinical interviewing, (3) the field of complex systems, and (4) the practice of agent-based modeling. We hypothesize that a fruitful way to study people's ways of reasoning about the world could be through studying their reasoning about emergent phenomena and complex systems, so we propose a new research methodology specifically designed for the intersection of these four paradigm shifts. In this section, we review each of these topics briefly and explain our reasoning on why they are important for our proposal.

We begin with the emergence of the theory of constructivism, which fundamentally changed our understanding of the nature of human knowledge and reasoning. Piaget (1972) and his colleagues successfully demonstrated that knowledge is not readily acquired from an outside source but is actively constructed by the learner. Underpinning our proposal are two specific theories that extend Piaget's theory of constructivism: (1) constructionism and (2) knowledge-in-pieces. Constructionism is a theory of learning that takes constructivism's connotations of "*learning as building knowledge structures*" and "*to know an object is to act on it*" (Piaget, 1972, p.20), and adds the idea that this "*happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity*" (Papert, 1980; Papert and Harel, 1991). Knowledge-in-pieces (KiP), on the other hand, is a constructivist theory that builds on artificial intelligence researchers' attempts to model knowledge and learning (e.g., Anderson, 1983; Minsky, 1986; Newell and Simon, 1972). KiP challenges conventional notions of knowledge as monolithic self-contained structures and conceptualizes it as a loosely organized network of primitive elements that are activated spontaneously and continuously reconfigured as the reasoning process is happening (diSessa, 1993).

The second paradigm shift is the methodology of clinical interviewing, which was invented by Piaget in the process of developing his theory of constructivism (Ginsburg, 1997). In contrast to standardized tests, which cannot go beyond behavioral manifestations of knowledge in pre-determined boundaries, clinical interviews give researchers the freedom to design object manipulation tasks, ask clarification questions, make on-the-spot hypotheses about subjects' reasoning, and test these hypotheses with

follow-up questions. Thanks to these strengths, clinical interviews provide opportunities to study more naturalistic forms of reasoning and to uncover hidden structures and processes (Clement, 2000; Ginsburg, 1997). Even though the original clinical interviews conducted by Piaget and his colleagues were mostly based on simple tasks for children, the methodology has evolved considerably over the years and became more reliant on verbal question-answer sequences. In addition, researchers have shown that participants' reasoning about a subject may evolve and even develop further during the course of a clinical interview (Sherin, Krakowski and Lee, 2012).

The third paradigm shift that we incorporate in our proposal is the emergence of the field of complex systems due to the way it dramatically changed our understanding of the world around us (Bar-Yam, 2004; Mitchell, 2009; Waldrop, 1993; Wilensky, 2001). A complex system is defined as "a group or organization which is made up of many interacting parts" (Mitchell and Newman, 2001, p. 1). Our lives are embedded in many complex systems such as the internet, economies, the brain, ecosystems, the weather and the immune system. Within these systems, many simple entities — often called components or agents — organize themselves without any central controller and the interactions between them result in a "collective whole that creates patterns, uses information, and, in some cases, evolves and learns" (Mitchell, 2009, p.4). The macro-level patterns which arise out of micro-level interactions between the parts of complex systems are called *emergent phenomena* (Wilensky, 2001). As a result, many real-world phenomena are non-linear, non-deterministic, stochastic and multi-leveled (Mitchell, 2009; Wilensky, 2001; 2003).

Lastly, agent-based modeling (Epstein, 2006; Wilensky & Rand, 2014) is a practice that emerged from the field of complex systems that "makes use of simple computational rules as the fundamental modeling elements" (Wilensky and Papert, 2010, p. 7) Hence, it is a paradigm shift from traditional aggregate-level models that are built using methods such as linear algebra or differential equations. The main reason that makes agent-based modeling compelling for our proposal is the fact that it offers a better epistemological match to our intuitive notions of *parts* that make up complex systems as distinct individuals or entities instead of aggregate populations (Wilensky and Papert, 2010). Thus, it is possible to teach the basics of agent-based modeling to a research participant and have them describe a real-world phenomenon through characteristics of entities such as people, organizations, or objects. In our initial formulation of the a-b-c interviews, we use the NetLogo agent-based modeling environment (Wilensky, 1999) as the construction tool because it is the most widely used agent-based modeling environment and it has its roots in both the field of complex systems and the field of learning sciences (Wilensky, 1999; 2001). It is a direct descendant of the Logo programming language (Papert, 1980) and it is designed to be a "low threshold, high ceiling" programming environment. Research has shown that students as young as in upper elementary school level can learn to develop models (e.g., Wilensky, 2003), but it is also used by professional scientists in cutting edge research (e.g., Maroulis et al., 2010; Pumain and Reuillon, 2017).

We argue that, much like the wood blocks used in Piagetian interviews on conservation of volume (Piaget, Inhelder and Szeminska, 1960), agent-based modeling offers material affordances that match well with real world phenomena. Actively constructing an agent-based model would require a participant to explicitly think about the constituents of an emergent phenomenon, making it possible to observe the participants' reasoning processes at more fine-grained levels as they unfold over the course of an interview. Being an active mediator between the participant and the agent-based modeling environment enables researchers to work with non-programmer participants on complex phenomena through an active dialogue that involves making and testing on-the fly hypotheses about the participant's reasoning.

A generative case study

In this section, we present a preliminary form of an a-b-c interview as a *generative case study* (Clement, 2000). The data presented here is taken from a previous study that was designed as an intervention to introduce the basics of agent-based modeling to adults with no prior experience in computational practices, and then help them develop a NetLogo model with the specific goal of exploring the effects of such an experience on their reasoning about stochastic phenomena. The intervention consisted of three one-on-one meetings with each participant. In the first meeting, the participant was

shown the basics of the NetLogo agent-based modeling environment. In the second and third meetings, a researcher helped each participant develop a model of a real-world issue of their choice. The participant described the model he or she wanted to develop and the researcher wrote the corresponding NetLogo code.

As we were analyzing the video data from this study, we came to notice the rich interaction between the researcher and the participants during the model construction process and decided to recalibrate our focus on the ideas presented in this paper. The case study shared in this section is one such interview conducted with **Karina** (pseudonym), who works as a special education paraprofessional in a public school in a large city in the U.S. We chose this case because it captures a single-meeting that starts with Karina expressing her idea and ending with her being satisfied with the model she developed with the help of the researcher.

Methodology

The interview with Karina lasted 50 minutes and 7 seconds and it is recorded in video. We analyzed the video by watching it through a qualitative video analysis software and marking instances in the video during which the participant was actively talking. We discarded in-between instances during which the researcher either just worked on writing the NetLogo code or explained how the code works to the participant. After this initial round of data reduction, we ended up with 22 short episodes. Out of these 22, we determined 9 of them as *main episodes* for the purposes of this paper. Then, we transcribed and analyzed each episode in detail by marking the parts of transcripts that highlighted Karina's reasoning. Finally, we built textual and visual representations of Karina's model for each episode (i.e., Sherin, Krakowski and Lee, 2012). We are going to present transcripts from these 9 episodes with visual snapshots of the NetLogo model, as well as our visual reconstruction of her model.

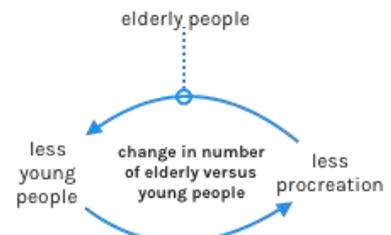
The topic of the interview: population decline

The emergent phenomenon presented in this section is chosen by the participant herself. Before the interview, she mentioned to the researcher that she read some news about declining populations in Japanese villages and she wanted to build a model of this issue. Population decline, or ageing, is listed as one of 18 global issues that "*transcend national boundaries and cannot be resolved by any one country acting alone*" by the United Nations (2016). It exhibits itself in many levels of the society and it is a constant headline in news, especially in developed countries, due to its implications for global economy in terms of workforce and health care systems (e.g., Anderson and Hussey, 2000; Rowe et al., 2016). It is a very suitable topic for the purposes of this paper because it is an emergent phenomenon embedded in greater complex systems in various levels such as local populations, global economies and healthcare systems.

Karina's model of population decline

The interview starts with Karina telling the researcher that she wants to work on a completely new idea that she came up with after coming across a story about the issue of ageing in Japanese villages in the news. The researcher welcomes her decision and asks her to articulate her idea on the record (see Table 1). She briefly talks about her idea and stops. The NetLogo model shown in Table 1 only contains two conventional buttons, *setup* and *go*, but no code.

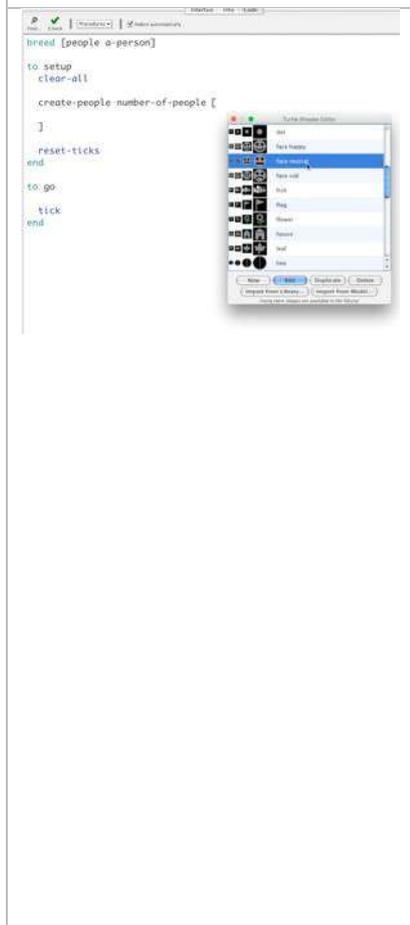
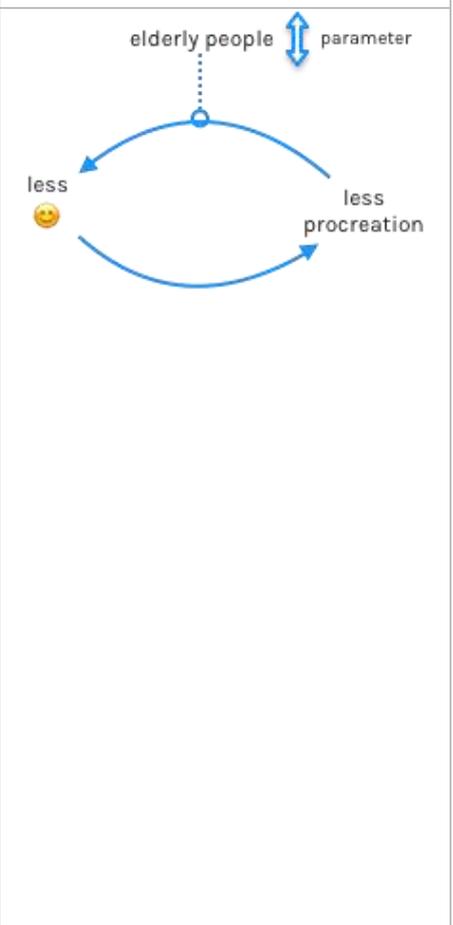
Table 1. Karina's description of her original idea

Screenshot	Transcript	Diagram model
	<p>(K)arina: So, the idea is based on a fact, which is, in Japan 40% of the population is over 60 years old if I'm not mistaken. So, they are coming across, umm, they are experiencing, what, what's my word? what did I say? I don't, I can't remember ...</p>	

	<p>(R)esearcher: Deaths and child birth and aging ...</p> <p>K: Well, that the population is, no-one there <i>will be</i>, the percentage of young people is less, <i>therefore there is a lot less people procreating</i>. So, in <i>some instances</i> villages are dying out because of the, umm, big <i>number of elderly versus young people</i> and so for example, umm, a village of 300 is now down to 30 <i>because of such a high population of old people</i>.</p>	
--	---	--

After Karina's initial explanation, the researcher proceeds to create a simple model in which there are old people and young people. He stops before creating any people in the model and asks Karina about how to visualize people in her model. As seen in Table 2, this question prompts Karina on not only deciding how the people in the model should look like but also talk about actual agent behavior. She also briefly mentions how she wants to be able to manipulate this model. We update our visual and textual representation of her model accordingly.

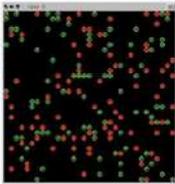
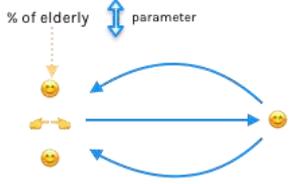
Table 2. Karina's first description of the agents

Screenshot	Transcript	Diagram model
	<p>R: So, the turtle shapes, which one do you prefer for people?</p> <p>K: The people, oh, the <i>old people should be unhappy</i>.</p> <p>R: Old people should be unhappy? OK.</p> <p>K: Yeah, <i>just because they can't, yeah, sad face</i>. Just for the fun of it (laughs).</p> <p>R: Let's start with the natural face, ...,no no no, but they will get older maybe. Or are we gonna just put older people initially and leave it like that? Or do you wanna create people like young ages, get them older, make them reproduce? Stuff like that?</p> <p>K: My idea was, <i>increase and decrease the number of old people and how that affects the population</i>.</p> <p>R: Oh, I see. So, then people will be old and young, right?</p> <p>K: Yes. Well, <i>old people will be considered people that can't have kids, obviously</i>. Or they are not reproducing.</p>	

Once again, the researcher proceeds to write the code that, he thinks, will produce the model that reflects *the model in Karina's mind*. He adds a number-of-people slider to the interface that determines the size of the population's village and an elderly slider that determines what percentage of this village is elderly people. When the setup button is clicked, the people of the model are created. Each person is designated as either young or old and then they are placed on random locations in the model's two-

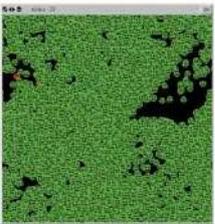
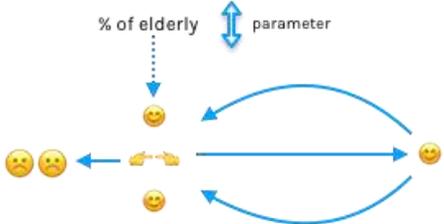
dimensional world. Once he finishes writing this code, the researcher clicks the setup button and asks Karina what she wants to do next. Table 3 shows Karina's response to this question.

Table 3. Karina's first description of agent behaviour

Screenshot	Transcript	Diagram model
	<p>R: OK. So, I have these people, they are in random places. They are the residents of this village. Umm, what's next? What are these people gonna do?</p> <p>K: Well, I'm assuming, umm, that's what I'm trying to wrap my head around. Umm. I guess if two people meet, they are gonna reproduce one person. And then, then now I'm thinking, yeah, we'll start there.</p>	

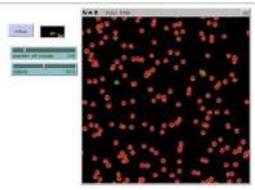
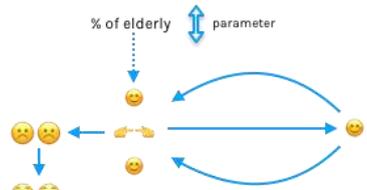
After hearing Karina's idea, the researcher starts writing the code so that each person continuously moves around randomly. When two young people touch each other, they produce a new young person. Once he writes this code and shows it to Karina, they run the model and notice that the number of young people grows exponentially in a very short time. In Table 4, Karina reacts to this outcome and notices that the model is not complete. She adds that she wants the parents to become immediately old when they make a baby.

Table 4. Karina's update of agent behavior

Screenshot	Transcript	Diagram model
	<p>R: Let's see if this works, ... oh yeah ...</p> <p>K: Wait, what about my old people?</p> <p>R: They are, they are in there but we are making so many babies randomly ...</p> <p>K: (laughs) ...</p> <p>R: Maybe we should like decrease the ...</p> <p>K: So, my thing is now, now the green people, the parents should be old now!</p> <p>R: They should be gr, oh, wha? ...</p> <p>K: When they make one baby, yeah, let's just, yeah.</p>	

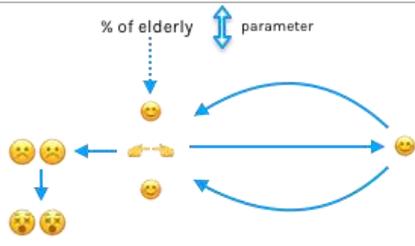
Once the researcher modifies the code accordingly, they run the model and they see that the number of young people decrease very quickly but the number of elderly people keeps increasing. Here, Karina thinks the outcome is interesting but her reaction indicates that she is not content with her model yet. The researcher senses her hesitation and asks her to elaborate. She hesitantly deliberates on whether the old people in the model should eventually die and what would be the implications of such an addition on the size of the village

Table 5. Karina's further updates of agent behavior

Screenshot	Transcript	Diagram model
	<p>K: I think it is interesting still, to see ...</p> <p>R: Yeah, almost everybody is old red, yeah ...</p> <p>K: But ...</p> <p>R: No, say it.</p> <p>K: I was just gonna say, not that it would affect our findings, but the old people have to [expletive] die ...</p> <p>R: Wanna do that?</p> <p>K: But that doesn't really matter, you know what I'm saying?</p> <p>R: Yeah! They just turn red and they cannot reproduce again.</p> <p>K: Right, exactly. But what does it tell about the size of the village? Do you know what I'm saying?</p> <p>R: Yeah! Do you wanna do that?</p> <p>K: (Nods approvingly).</p>	

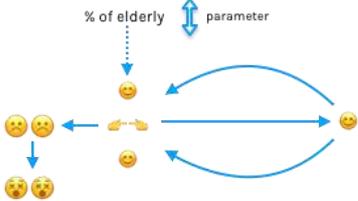
After this change, the researcher runs the model and Karina expresses excitement with the model because everyone dies. This point is marked as the completion of her initial model.

Table 6. Karina's reaction to the first completed version

Screenshot	Transcript	Diagram model
	<p>R: Let's see ... So?</p> <p>K: [expletive]! Everyone's [expletive] oh damn! Look at that!</p>	

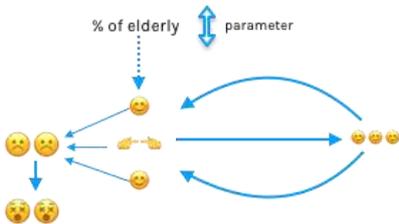
After the addition of the mechanism of death, they run the model again and Karina now displays excitement about her model. The researcher probes her about the model's outcomes. He also intervenes with his own ideas. He first suggests adding a plot that shows the number of all the people in the village and also the number of elderly over time. When they add this plot and run the model, they see that the percentage stays relatively stable even though the village continuously declines and eventually dies. The researcher probes her about this outcome. She asks to change the model in a way that young people may get old even if they never made babies. The researcher intervenes and suggests adding the possibility of making more than one baby, too. When the researcher asks this question, Karina decides that she wants each family to have between 1 to 3 kids.

Table 7. Researcher's intervention with his ideas

Screenshot	Transcript	Diagram model
	<p>K: I think it is interesting still, to see ...</p> <p>R: I don't know what to make of this because the difference between the elderly and, didn't change much.</p> <p>K: Yeah.</p> <p>R: There was always similar proportion of elderly and young people.</p> <p>K: Hmm, but then the young people don't stay, they can't go on forever until they make a partner, until they meet a partner, right? Young people also die!</p> <p>R: Yeah. Young people, if they make just one baby they die out.</p> <p>K: Right. That's the only reason why they die. They don't die because of natural causes.</p> <p>R: (Nods) They turn to old and they die. I think we should also add the more than one baby thing here ...</p> <p>K: Yeah, yeah, yeah.</p> <p>R: Because it's like, of course this is gonna die because they don't make more than one baby, so like they keep decreasing.</p> <p>K: Yeah, yeah, yeah. OK!</p>	

The idea of having multiple babies in Episode 7 causes confusion between the researcher and Karina, probably due to the fact that the researcher introduced the idea himself. Episode 8 starts when the researcher implements a mechanism for multiple babies. Karina pushes back because she wants to know how this model works. Once the researcher explains how the current version of the model works, she realizes that the model does not do what was on her mind and explains exactly how she wants the model to work. Even more, she explains why she wants the model to work that way.

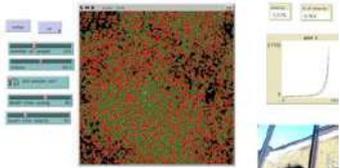
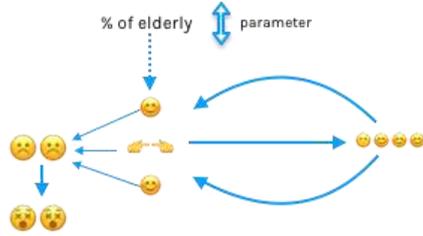
Table 8. Karina's first implementation of the idea of fertility rate

Screenshot	Transcript	Diagram model
	<p>R: So, we, we give them some initial fertility. You know what that means?</p> <p>K: OK. But, but, how many babies can be made after the initial interaction? Is there a range or are they all automatic? OK there is a ...</p> <p>R: Yeah. Right now, every single time I create a person, I give them a baby count. So, I have 3, you have 2, and after making 2 babies you turn old, after making 3 babies I turn old.</p> <p>K: Oh! I see.</p> <p>R: Minimum is 1, maximum is 3 and it's random.</p> <p>K: OK. I see. But I guess, I guess what I'm saying is, from you with, you and I</p>	

	<p>reproduce, how many babies are we gonna have?</p> <p>R: Just one. Every single time ...</p> <p>K: Oh! See, that's where, that's where I wasn't wrapping my head. I wanted it to be where they can either make 1 baby to 3 baby, the family size.</p> <p>R: Ah! That's so much better!!!</p> <p>K: Yeah. Because I'm thinking 3, because 3 is like the average in US.</p>	
--	---	--

In the final episode of the interview, the researcher and Karina run the version of the model where each pair of parents procreate either 1 or 2 or 3 young people randomly and then turn to old people, as well as some young people eventually becoming old even if they cannot find a partner to make babies. When they run this version of the model, they notice that the population grows exponentially. Once again, Karina probes the researcher about the way the model works. She asks whether a couple can have 0 babies. When she realizes that it is not the case, she tells the researcher that she actually wanted some couples to have 0 babies because there are "some sterile people". After this episode, Karina was satisfied with the model and the interview ended.

Table 9. Karina's reaction to the first completed version

Screenshot	Transcript	Diagram model
	<p>Researcher: It goes crazy.</p> <p>Karina: And they can have 0 babies, right?</p> <p>Researcher: No!</p> <p>Karina: Yeah, they can have 0 babies, too!</p> <p>Researcher: Let's make it ...</p> <p>Karina: I thought that's what we have said, Umit! [expletive]!</p> <p>Researcher: Let's do ...</p> <p>Karina: Because I mean there is some people that are sterile.</p> <p>Researcher: Yeah. So, ..., this time the population died out. Let's make it 200, maybe ... Now it is like, interesting, ...</p> <p>Karina: (yelling) I'm [expletive] smart!!! When am I gonna get accepted to U of C, (yelling) when? (laughs)</p>	

Discussion

Although it was an early form of a-b-c interviews, this case study provides us valuable insights to hypothesize about the main features of a-b-c interviews, as well as the major challenges in conducting such interviews systematically. We summarize our preliminary findings in two arguments: (1) the interaction between Karina, the researcher and the model afforded us to observe patterns in Karina's reasoning about this emergent phenomenon that may not have been possible through asking only verbal questions, (2) the interview highlighted a reasoning process that resembles models or mini theories that Karina constructed on the fly and continuously reconfigured drawing on many smaller

pieces of her knowledge from many different contexts and levels. We also discuss the researcher's moves as an interviewer in two arguments (1) the researcher's active participation ended up problematizing the validity of the findings but also provoked fruitful reasoning as seen in episodes 6 through 8, (2) the researcher failed in taking full advantage of Karina's model in probing her to elaborate her reasoning in greater detail.

Karina's reasoning

Episode 1 starts with Karina explaining the phenomenon she chose to model: the decline of the population in Japanese villages. Within itself, this episode resembles a short verbal interview with her. She mentions a positive feedback loop mechanism when she says "*therefore there is a lot less people procreating*", brings up two variables when she utters "*the number of elderly versus young people*", and finally speculates on the cause of this emergent phenomenon with her utterance "*because of such a high population of old people*". As the interview proceeds, we observe Karina develop a more and more sophisticated model. She first adds a mechanism for procreation and then adds a mechanism for aging, both through definitions of how young people behave and how old people behave. She also describes how she envisions using the model by "*increasing and decreasing the number of old people and how that affects the population*". Her reasoning goes beyond a simple causal relationship between the percentage of elderly people and her final model incorporates the idea of *fertility rate* as the major factor, although rather accidentally and implicitly. We argue that without the construction element of the interview, we would not have been able to observe her reasoning about the individuals that make up this complex system and we could have even concluded that she holds misconceptions (or naive theories) about the phenomenon of population decline such as assuming deterministic mechanisms (Wilensky and Resnick, 1999).

Constructing an agent-based model with her enables exposition of her reasoning about an emergent phenomenon at a fine-grained level including some less salient ideas. For instance, we notice that she draws from a number of ideas from her personal knowledge about how people behave and how people procreate. Even though she is specifically thinking about the population of a Japanese village, she justifies her design decisions based on ideas from various resources in various levels such as making old people look like *sad faces* because "*they can't*" procreate, making families have up to 3 babies because "*it is the US average*", and making it possible for some families to have no babies because "*some people are sterile*". Taken together, these episodes show us that she is not just throwing random ideas at the model. Quite the contrary, she has a specific explanation for each of her modeling decisions.

Lastly, her modeling decisions expose a non-monolithic reasoning process about this emergent phenomenon. For example, she decides to make the model so that "*two people produce only one baby and immediately get old*" but with the caution that this is only where she wants to "*start from*". We see another such instance in Episode 4, when she hesitates whether the old people in the model should eventually die or not. In both cases, we see her bringing together many pieces of her knowledge about people, aging and procreation, but also having difficulty in figuring out how all these pieces fit together when it comes to population decline. This finding is consistent with diSessa's (1993) knowledge-in-pieces theory, but we are cautious about such straightforward associations because we do not have any data that could highlight the exact nature of this process, and further research needs to be conducted.

Researcher moves

We notice a number of potentially problematic interventions from the researcher throughout the interview. This is mainly due to the fact that his goal was not to interview Karina but to help her create a NetLogo model. The first of these interventions come at the very beginning of the interview. When Karina asks the researcher to remind her of what she said about population decline, the researcher mentions "*deaths and child birth and aging*". We observe all these ideas in Karina's utterances in the following episodes. It seems like Karina does not immediately pick up these ideas and keeps talking about her original idea but it is still unclear whether the researcher's move implicitly impacted Karina's reasoning. Hence, a potential argument against the validity of our preliminary findings would be that

such moves might have led her towards specific ideas. We acknowledge that this is a *major* challenge in conducting a-b-c interviews; much attention must be paid to prevent the interview from turning into a teaching intervention. On the other hand, we argue that it is important to position the researcher as an active mediator not only because the participant is assumed to be a novice in agent-based modeling, but also because the researcher attempts to model the participant's reasoning through writing the model's code and gets immediate feedback from the participant. This is a direct extension of a major strength of clinical interviews; the researcher is expected to formulate on-the-fly hypotheses based on the participants' responses and ask follow-up questions (Ginsburg, 1997).

In this specific case study, the researcher's interventions also end up triggering fruitful episodes that, we argue, may not have happened otherwise. Such an intervention happens in Episode 6, when Karina brings up the idea to make young people old after some time. The researcher suggests adding an initial mechanism of "*multiple babies*" and she agrees. However, instead of probing Karina on how to implement this idea in the model, the researcher proceeds to implement his own mechanism, which ends up being quite different from what was in Karina's mind. She eventually catches the researcher's intervention and forces the researcher to correct the model. This episode affords us to notice that Karina wants to configure procreation in her model as a process that only happens in monogamous families. This observation may or may not reflect her actual reasoning but implies that she assumes other possibilities as negligible. Unfortunately, the researcher does not ask follow-up questions about such important points. This brings us to our last point that the researcher's moves fail in asking Karina questions that probe her to elaborate her ideas in greater detail. We end up not being able to certainly assert whether she actually changed her focus from the percentage of elderly to fertility rate. We also cannot know whether she left out some factors, such as polygamy, nutrition and migrations, deliberately or she did not think of those.

Concluding Remarks

It has been shown that people, young and old, struggle greatly in making sense of emergent phenomena (Chi, 2005; Wilensky and Resnick, 1999) although our lives are embedded in such phenomena in many levels (Mitchell, 2009; Wilensky, 2001). Agent-based modeling has been shown to be effective at progressing learners in their understanding about emergent phenomena (Wilensky, 2003; Wilensky and Reisman, 2006). We proposed a new research methodology, tentatively named agent-based construction (a-b-c) interviews, specifically to study people's reasoning of emergent phenomena and complex systems. We situated our methodology at the intersection of four major paradigm shifts: (1) the theory of constructivism (Piaget, 1972), (2) the methodology of clinical interviews (Ginsburg, 1997), (3) the field of complex systems (Mitchell, 2009), and (4) the practice of agent-based modeling (Wilensky, 2001). We then presented a generative case study, which was originally designed as an intervention, but offered a first glimpse on potential affordances and challenges of a-b-c interviews. In the case study, an adult participant, Karina, described a model of aging in a Japanese village and the researcher wrote the code for her. We argued that the interaction between Karina and the researcher exposed complex, spontaneous patterns of reasoning, which could not have been observed through verbal questions or simple tasks such as drawings. We also argued that the main challenge of a-b-c interviews is also the main strength of them. The act of mediating is simply the act of hypothesizing about the participant's reasoning, much like traditional clinical interviews (Ginsburg, 1997), but testing these hypotheses through writing the code and getting feedback from the participant instead of only asking verbal questions. Yet, if not done carefully, this can easily evolve into intervening rather than mediating. We end this paper by offering a very first working definition: a-b-c interviews are a special class of clinical interviews that are conducted through an open-ended agent-based modeling task that is actively mediated by a researcher.

References

- Anderson, J. R. (1983). Cognitive science series. The architecture of cognition.
- Anderson, G. F., & Hussey, P. S. (2000). Population aging: a comparison among industrialized countries. *Health affairs*, 19(3), 191-203.

- Aslan, U., & Wilensky, U. (2016a). Restructuration in Practice: Challenging a Pop-Culture Evolutionary Theory through Agent Based Modeling. In *Proceedings of the Constructionism 2016 Conference*. Bangkok, Thailand.
- Aslan, U., & Wilensky, U. (2016b). Old Tricks Revisited: Studying Probabilistic Reasoning through Incorporating Computer Modeling into Piagetian Research. *Paper presented at the Jean Piaget Society 46th annual meeting*. Chicago, IL, June 9 - 11.
- Bar-Yam, Y. (2004). *Making things work: solving complex problems in a complex world*. Knowledge Industry.
- Chi, M. T. (2005). Commonsense conceptions of emergent processes: Why some misconceptions are robust. *The journal of the learning sciences*, 14(2), 161-199.
- Chi, M. T., & Roscoe, R. D. (2002). The processes and challenges of conceptual change. In *Reconsidering conceptual change: Issues in theory and practice* (pp. 3-27). Springer, Dordrecht.
- Clement, J. (2000). Analysis of clinical interviews: Foundations and model viability. *Handbook of research design in mathematics and science education*, 547-589.
- DiSessa, A. A. (1993). Toward an epistemology of physics. *Cognition and instruction*, 10(2-3), 105-225.
- Epstein, J. M. (2006). *Generative social science: Studies in agent-based computational modeling*. Princeton University Press.
- Ginsburg, H. (1997). *Entering the child's mind: The clinical interview in psychological research and practice*. Cambridge University Press.
- Hmelo-Silver, C. E., Marathe, S., & Liu, L. (2007). Fish swim, rocks sit, and lungs breathe: Expert-novice understanding of complex systems. *The Journal of the Learning Sciences*, 16(3), 307-331.
- Jacobson, M. J. (2001). Problem solving, cognition, and complex systems: Differences between experts and novices. *Complexity*, 6(3), 41-49.
- Johnson, C. W. (2006). Complexity in design and engineering. *Reliability Engineering & System Safety*, 91(12), 1475-1588.
- Klopfer, E. (2003). Technologies to support the creation of complex systems models—using StarLogo software with students. *Biosystems*, 71(1-2), 111-122.
- Levy, S. T., & Wilensky, U. (2008). Inventing a “mid-level” to make ends meet: Reasoning between the levels of complexity. *Cognition and Instruction*, 26(1), 1-47.
- Maroulis, S., Guimera, R., Petry, H., Stringer, M. J., Gomez, L. M., Amaral, L. A. N., & Wilensky, U. (2010). Complex systems view of educational policy research. *Science*, 330(6000), 38-39.
- Mitchell, M. (2009). *Complexity: A guided tour*. Oxford University Press.
- Mitchell, M., & Newman, M. (2002). Complex systems theory and evolution. *Encyclopedia of Evolution*, 1-5.
- National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press.
- Piaget, J. (1972). Development and learning. *Readings on the development of children*, 25-33.
- Piaget, J., Inhelder, B., & Szeminska, A. (1960). The child's conception of geometry.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Penner, D. E. (2000). Explaining systems: Investigating middle school students' understanding of emergent phenomena. *Journal of Research in Science Teaching*, 37(8), 784-806.
- Pumain, D., & Reuillon, R. (2017). *Urban dynamics and simulation models*. Springer International Publishing.

- Rowe, J. W., Fulmer, T., & Fried, L. (2016). Preparing for better health and health care for an aging population. *Jama*, 316(16), 1643-1644.
- Sengupta, P., & Wilensky, U. (2011). Lowering the learning threshold: Multi-agent-based models and learning electricity. In *Models and Modeling* (pp. 141-171). Springer, Dordrecht.
- Sherin, B. L., Krakowski, M., & Lee, V. R. (2012). Some assembly required: How scientific explanations are constructed during clinical interviews. *Journal of Research in Science Teaching*, 49(2), 166-198.
- Sterman, J. D. (1994). Learning in and about complex systems. *System Dynamics Review*, 10(2-3), 291-330.
- United Nations (2016, November 10). Global Issues Overview. Retrieved from <http://web.archive.org/web/20161110161502/http://www.un.org/en/sections/issues-depth/global-issues-overview/>.
- Waldrop, M. M. (1993). *Complexity: The emerging science at the edge of order and chaos*. Simon and Schuster.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U. (2001) Modeling nature's emergent patterns with multi-agent languages. *Proceedings of EuroLogo 2001*. Linz, Austria
- Wilensky, U. (2003). Statistical mechanics for secondary school: The GasLab modeling toolkit. *International Journal of Computers for Mathematical Learning*, [Special Issue on agent-based modeling]. 8(1), 1-41.
- Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep or a Firefly: Learning Biology through Constructing and Testing Computational Theories -- an Embodied Modeling Approach (PDF). *Cognition & Instruction*, 24(2), pp. 171-209.
- Wilensky, U., & Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Perspective to Making Sense of the World (html) (pdf). *Journal of Science Education and Technology*, 8(1).
- Wilensky, U., & Papert, S. (2010). Restructurations: Reformulations of Knowledge Disciplines through new representational forms. In J. Clayson & I. Kalas (Eds.), *Proceedings of the Constructionism 2010 Conference*. Paris, France, Aug 10-14. p. 97.
- Wilkerson-Jerde, M. H. & Wilensky, U. (2015). Patterns, probabilities, and people: Making sense of quantitative change in complex systems. *Journal of the Learning Sciences*, 24(2), 204-251. doi: 10.1080/10508406.2014.976647

Active Learning of Computer Science Using a Hackathon-like Pedagogical Model

Jake Rowan Byrne, *jake.byrne@tcd.ie*

The Trinity Centre Research for IT in Education, School of Education and School of Computer Science & Statistics, Trinity College Dublin, the University of Dublin, Ireland

Kevin Sullivan, *kevin@bridge21.ie*

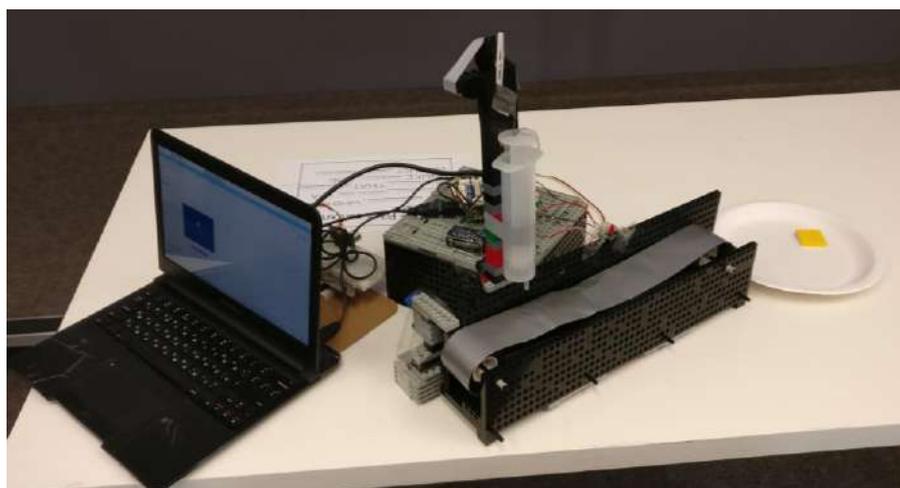
The Trinity Centre Research for IT in Education, School of Education and School of Computer Science & Statistics, Trinity College Dublin, the University of Dublin, Ireland

Katriona O'Sullivan, *Katriona.OSullivan@mu.ie*

Department of Adult and Community Education, Maynooth University, Kildare, Ireland

Abstract

Despite extensive literature on Computer Science (CS) education, there are few pragmatic pedagogical models that support active and project based learning. The new Irish CS course for post-primary schools advocates such an active learning approach through a series of “Applied Learning Tasks” (ALT). This paper explores the use of a constructivist/constructionist 21st Century pedagogical model, delivered as a weeklong “Hackathon-like” activity, targeting the learning outcomes for the embedded systems ALT from this new CS course. Twenty-one students participated in the workshop and completed pre and post surveys to assess their confidence in CS topics and their associated learning outcomes. Analysis revealed that students were more confident in almost all the learning outcomes surveyed. There were particularly significant increases in the embedded systems learning outcomes, which was the focus of the students’ projects. The findings suggest that the combination of a “Hackathon-like” event and a constructivist/constructionist learning model can be effective in increasing student motivation, confidence and learning of Computer Science concepts and skills at post-primary level.



“Froasties”: Participants created an automated food processor, specializing in French toast.

Keywords

constructivism; constructionism; hackathon; computer science education; creativity; design thinking; problem solving; prototypes; teamwork

Introduction

This paper builds on prior work (Byrne, O’Sullivan, & Sullivan, 2017) that explored students aspirations for careers in Computer Science to examine how a similar intervention can be used to support Ireland’s

new Leaving Certificate examination in Computer Science (NCCA, 2017a). The Leaving Certificate (LC) is a specialized terminal exam for post-primary schools used to determine university admissions in Ireland. The LC in computer science is being introduced in September 2018, so there is a need to establish effective teaching methodologies which will lead to success for students preparing for the exam.

This work outlines the rationale for the design and delivery of a 21st century learning experience modelled around a hackathon-like event. Reportedly, hackathon attendees go with the hope of “learning” something new (Briscoe & Mulligan, 2014). This is the most popular rationale for attendance, which implies that hackathons provide a viable context for learning. Additionally, the fact that hackathons are most commonly technology-oriented creates a persuasive argument for incorporating a hackathon-like activity into the classroom to promote a “learning by doing” constructionist approach to learning (Kafai & Resnick, 1996).

The Bridge21 model promotes project-based and “learning by doing” values inherent in constructionist learning (Kafai, Lee, et al., 2014). The model also emphasizes the importance of peer-to-peer learning through a social constructivist philosophy and technology mediated learning (Byrne et al., 2017; Lawlor, Conneely, Oldham, Marshall, & Tangney, 2018). All students attending the learning experience will have had prior experience with the Bridge21 model.

Through their previous exposure, they learned to work in teams and to program with Blockly and Scratch. These skills will be called upon during the learning experience when the participants are divided into teams and guided in the creation of their product designs. They had four days to generate an idea in the realm of IoT and wearables, prototype it, and prepare a pitch to “sell” the product. They had Arduino, Python, Scratch and an assortment of sensors at their disposal throughout the process. The collaboration and creativity required is expected to enhance their confidence and abilities with these technologies.

Background

Introduction of Computer Science at post-primary level in Ireland

The Irish government has recently announced the introduction of a new curriculum/specification for Computer Science as an optional subject as part of the Leaving Certificate (LC) at post-primary level (NCCA, 2017a). The Leaving Certificate is a high stakes set of exams which takes places at the end of a two year cycle, equivalent to 11th and 12th grades in K-12. Leaving Cert results are used to determine entrance to third level education. The new course hopes to develop traditional CS content knowledge as well as project and team management skills in the students, in line with the “*broad, balanced*” educational aims of the Leaving Certificate (NCCA, 2017b).

There are three strands to the current draft specification (NCCA, 2017a): Practices and Principles, Cross-Cutting Concepts and Computer Science in Practice (Table 1).

There are a total of 66 learning outcomes spreads across these three strands. The aim is to use strand 3 projects to iterate through the learning outcomes from strands 1 and 2. This means not every learning outcome will be addressed in each project, but after the students have completed all five projects they will have covered all 66 learning outcomes.

This is the first time that Computer Science has been introduced as a dedicated subject in Irish schools so potential teachers will have limited experience and practice in delivering such courses. This suggests a need to develop practices and activities that can both support the learning of traditional content while also supporting the development of team and project management skills.

Approaches for Enhancing STEM Education

There is a lot of discussion about various approaches to STEM education. This varies from approaches that explore STEM education as a general phenomenon (Johnson, Brown, Cummins, & Estrada, 2012; Sanders, 2009; J. Williams, 2011) to practical projects that explicitly describe STEM learning experiences (Barker, Melander, Grandgenett, & Nugent, 2015; Fee & Holland-Minkley, 2010; Nugent,

Barker, Grandgenett, & Welch, 2014; Tangney, Oldham, Conneely, Barrett, & Lawlor, 2010). Despite differing views on pedagogy, e.g. enquiry based, integrative or problem based learning, the need for more STEM graduates is generally agreed upon.

Table 1. Leaving Certificate Computer Science Course Draft Specification

Strand 1	Strand 2	Strand 3
Practices and Principles	Cross-Cutting Concepts	Computer Science in Practice
Computers & Society	Abstraction	User Centered Design
	Computer Systems	Information Systems
Computational Thinking	Data	Analytics
	Algorithms	Modelling & Simulation
Designing & Development	Evaluation/Testing	Embedded Systems

Supporting the approach of the new Leaving Certificate course, Roberts (Roberts, 2012) suggests that both domain knowledge and 21st century “soft” skills such as, teamwork, creativity, problem solving and inquisitive thinking are all important in STEM education. Roberts also suggests that domain knowledge should be learnt through “*authentic problem solving in rich social, cultural and functional contexts*”. These approaches would provide students with the rich learning experience that the new curriculum is aspiring to promote.

Hackathons

A hackathon usually involves the prototyping of some digital artefact and pitching or presenting that prototype as part of a team-based “problem-focused computer programming” activity (Briscoe & Mulligan, 2014) over a 24-48 hour period. Hackathons generally aim to combine both team dynamics and the “authentic problem solving in rich social, cultural and functional contexts” that Roberts suggests. This is a common motivating element of hackathons, which tend to focus on some cultural or social issue that the participants can identify with.

There have been some attempts to explore which pedagogical approaches are used in hackathon-like activities (Duhring, 2014; Nandi & Mandernach, 2016; Skirpan & Yeh, 2015). These studies fall short as they have limited practical and theoretical basis and are generally exploratory in nature. However, they do suggest that hackathon-like activities might be useful when considering the design of contextualized and practical learning experiences.

Contextualizing STEM topics is seen as important in broadening participation so it is recommended to structure activities so that they emphasize creativity, design and problem solving while working on real world problems (Cooper & Heaverlo, 2013). Hackathon-like approaches include these elements and have been used with pre university students in STEM subjects (Kafai, Rusk, et al., 2014). This suggests that hackathons could make for a viable framework to develop effective and inclusive STEM activities but there is limited research on how to structure such a hackathon-like activity for learning.

21st-Century Learning

In addition to the introduction of Computer Science as its own subject there are efforts being made to integrate 21st century skills across the curriculum (Dede, 2010; NCCA, 2014; Rychen & Salganik, 2005). These involve the development of key skills such as effective communication, thinking critically and teamwork. The Bridge21 Model (Tangney et al., 2010) has been used to teach Computer Science using a social constructivist pedagogical approach (Byrne et al., 2017). This approach focuses on team based activities that are designed to develop both traditional content knowledge and 21st century skills.

Materials and Methods

Bridge21 Pedagogy and Model

The Bridge21 learning experience emphasizes teamwork, learning by doing and a technology-mediated approaches (Lawlor et al., 2018). The model has been adapted for use in a wide range of subjects such as history (O'Donovan, 2015), mathematics (Tangney & Bray, 2013) and computer science (Byrne et al., 2017; Tangney et al., 2010). It is a social constructivist approach that moves away from traditional teacher-led learning as it is designed to foster intrinsic student motivation and learning potential (Lawlor, Marshall, & Tangney, 2015). The teachers' goal in this model is to facilitate the activities, by encouraging students to think for themselves and model problem-solving processes. In line with the literature on STEM and CS education, this approach promotes collaboration, discovery learning and problem solving. Vygotsky's (Vygotsky, 1978) "more able other" informs the team formation and facilitation to promote peer learning and mentorship. Constructivist and constructionist approaches are increasingly being used to develop learning experiences especially when looking for students to be creative while working with computing technologies (Brady et al., 2017; Kafai & Resnick, 1996; Przybylla & Romeike, 2014).

Bridge21 Activity Model

The Bridge21 activity model (Byrne, Fisher, & Tangney, 2015) follows 7 phases that can be implemented over the course of an hour, a day, a week or as needed depending upon the scope of the activity. The 7 phases are: Setup, Warm up, Investigate, Planning, Create, Present and Reflect. It incorporates many elements known to be conducive to teamwork; self-directedness, creativity, and positive self-driven experience. The activity model is a largely linear structure, with some iteration to cater to the personalized needs of the students, as well as the demands of a particular project. This method also mirrors the Agile model of software development (Kastl, Kiesmüller, & Romeike, 2016; Rico & Sayani, 2009; Romeike & Göttel, 2012). To maximize the educational outcomes, the structure of the hackathon follows the Bridge21 activity model.

Before the "Hackathon"

The attendees have previously taken part in introductory Bridge21 Computer Science workshops. From these experiences the students develop a working knowledge of relevant technologies. For example, they have been introduced to Scratch, Blockly, and LEGO Mindstorms through Bridge21. They've also learned computing essentials such as conditionals, looping, initialization, variable instantiation. This means that there is a consistent elevated baseline throughout all attendees and simulates a base level of experience, equivalent to rudimentary learning that might occur in early projects as part of the Leaving Certificate coursework. The students, having volunteered for this programme, have expressed an interest in computer science, similar to the demographic who would choose to study CS as a subject at LC.

Hackathon Challenge

Internally the event is referred to as "Invent Week" and is adapted from the popular hackathon model already employed at levels ranging from hobbyist to professional settings. Traditionally, these hackathons are held over the course of a consecutive 24-72 hours during which the participants are nourished, housed, and free to work according to their own (often frantic) schedule in productive teams. This breakneck pace and high-pressure environment would be inappropriate for many of the students considering their age (15-17 years) and alternative commitments they may have. The same limitations would apply in any secondary school trying to implement our hackathon-like model. As a result, we've tailored the structure to be more accommodating.

Rather than leaving all of Invent Week as a solid block of time over the course of 1-3 days, the in-class hackathon-like event is run as a 4-day project where the students work from 9:30 am to 3:00 pm each day. This allows them to tend to their other commitments, and maintain a healthy lifestyle without the restraints imposed by a typical hackathon. This helps to make the model replicable at other institutions.

Each of the 4 days will contain a specific focus as the project builds towards a final deadline at lunchtime on the fourth day.

The goal of the week is to target one of the four Applied Learning Tasks (ALT) on the new LC CS course. The Embedded Systems ALT was chosen as it was well suited to a hackathon-like event. The learning outcomes from this project dealt with a number of topics: digital and analogue inputs, controlling digital outputs, storing analogue input data, automating processes and designing a program that utilizes all of these elements. Invent Week was designed to explore these topics experientially, rather than through traditional content delivery.

Day one will correspond with the first 3 steps in the Bridge21 activity model, and partially the fourth. This day will consist of the teams coming together for the first time (set up) and performing some ice breakers including divergent thinking (warm up) about computers in everyday life. Students also learned about the technologies they used including Python, Arduino, and Raspberry Pi through the completion of a number of set tasks (investigate).

The concept/idea planning phase should conclude early on day 2 as team members are assigned roles, then production begins. They will be loosely monitored as they progress, but a mainstay of social constructivist/constructionist education and the Bridge21 model is the experiential learning that comes from working through a process with peers using real world artefacts and programs. They will spend days 2, 3, and part of 4 completing their project (create) before concluding the week by presenting their project on day 4.

Technical Infrastructure

In terms of embedded systems hardware the teams were provided with a Raspberry Pi running Raspbian OS with an Apache webserver, PHP and Python; Arduino Uno boards; a motorized robotic chassis with Grove shield/hat adapter add-ons for Arduino and Raspberry Pi; and an array of input sensors (heart-rate detector, light detector, microphone etc.) and outputs (LEDs, servos, motors, etc.).

Additionally each team had access to a video camera, sound recorder, open source software (Audacity, Scratch, etc.) and 2 PCs running Windows. Limiting the 4 person teams to 2 PCs is intended to encourage paired programming (L. Williams, Kessler, Cunningham, & Jeffries, 2000). The students were encouraged to think beyond the confines of the provided hardware and software, as approaches can be taken to simulate a desired outcome. They were supported by the mentors with help in finding such approaches.

Mentoring and Facilitation

The learning experience was facilitated by one lead coordinator and 4 mentors, proffering a 4:1 student/staff ratio and a 1:1 team/staff ratio. The mentors were not assigned to specific teams. They moved from team to team, monitoring progress and stayed out of the way when not needed but were always open to query from any team. This was done intentionally to encourage the students to dissect their problems as thoroughly as they could before requesting external assistance. This reflects the real world practice of reaching out for expert advice from a technical consultant.

The lead coordinator had a background in mechatronic engineering, while the mentors each had backgrounds in computer science. When called upon, the mentors would help the students to reach a solution, while taking care to not simply tell students the answer. This required a question-based or "Socratic" teaching approach where small but relevant bits of information are revealed until the student could form their own solution. The aim is for students to develop their computational thinking skills, which ideally helps them to grow in the LC learning outcome areas and become more independent problem solvers.

Methodology

Twenty-one students participated in Invent Week. Ten of the participants were female, and eleven were male providing a gender-balanced cohort. Ethical approval was sought from and granted by the School of Computer Science and Statistics at Trinity College Dublin.

The data collection consisted of a pre and post questionnaire design, each of which was completed at the students' own discretion. None of the questions were mandatory, and the students were made aware that they could skip any part they did not feel like answering. Every question was personally evaluative, asking things such as "I feel confident in my ability to evaluate different solutions to one problem" or "I feel confident in my ability to describe what an algorithm is and give an example". Every answer was based on a 5-point Likert scale (with 5 being strongly agree, and 1 being strongly disagree). The total of 63 questions were drawn from two sources.

The first 34 questions came from a questionnaire used in previous instances of Invent Week (Byrne et al., 2017) These queries establish the student's interest and knowledge about pursuing careers in computer science, through university and as a realistic future. This data is valuable for tracking what information is resonating with the students, even when it's not necessarily the central focus of the week or this paper. The following 29 questions were drawn from the learning outcomes for the LC CS course (NCCA, 2017a). These address much more specific skills like confidence with embedded systems or I/O operations. It is very important that a student has these mastered in order to pass their exams at the end of secondary school.

By providing a pre and post questionnaire oriented towards the learning experience, we assess the students' self-perceived learning during Invent Week. The level of complexity and technical-knowledge behind the prototype creation also helped us to evaluate the learning occurring during the program. This did not provide quantitative data, but it helped us to tailor our lessons according to the students' needs. This adaptability is replicable in a typical classroom setting.

Results

Research Questions

Can a Bridge21 style Hackathon improve students' motivation, confidence and learning of computer science concepts and skills, particularly in the area of embedded systems?

Prototypes

Each of the five teams participating in Invent Week created an embedded systems prototype in areas of Internet of Things, wearables, home automation, and robotics. A heavy emphasis was placed on incorporating internet connectivity into the devices, considering the increasing trend of "smart" technologies and the important skills gleaned from working with internet communication. Furthermore, every team created a website for their product and many went on to add social media pages and videos illustrating the needs addressed by their product. The artefacts and code related to each group has been compiled into folders and saved for later review.

"Poc Doc" was a wearable medical device intended for use in cases ranging from minor injuries to life threatening situations. The product is worn on the arm and links to a medical API that can offer treatment suggestions based upon the symptoms it detects in the user. Detection is completed with an array of Arduino sensors which track heart rate, body temperature, and sudden changes in acceleration - indicating a fall or concussive force. The device will automatically call an ambulance to its location if the user doesn't interact with it for a certain time period following a concussive event. This team also made attempts at calling a Google voice API to respond to medical queries.

"Froasties" was an automated food processor, specializing in French toast. It is designed for industry use in hotels and restaurants. Users are given the option to log into a website and activate the machine with the press of a button. This was accomplished through a crossover of HTML, PHP, and Arduino scripts working through an embedded Raspberry Pi. The physical prototype was created with Lego Technic.

"Benny the Bin Buddy" was a smart attachment that a user could attach to their bins for automatic sorting between recyclables and rubbish. This was accomplished by simulating a spectrometer with colored LEDs and an Arduino light sensor, in tandem with Arduino Servos that could dump the objects into the appropriate bin depending upon their colors. The detected colors were checked against a website before

selecting if it was recyclable or not. Further development would lead to linkage with a database that would be updated to successfully sort more object types over time.

“iWindow” was an automated home technology and a smart window. It had internet connectivity, allowing a user to remotely open or close it or lock it with a button on their website. It was also outfitted with sensors that tested UV levels, air quality, and temperature both inside and outside the window. It then made the decision to automatically open or close the window with an Arduino Servos based upon the inputs.

Lastly, “Emo-Tee” was a smart clothing device employing a wearable LCD and internet connectivity. It was a t-shirt with connectivity to a web-app which allowed the user to select which image, from a range of emoji, would appear on the front of the shirt. The images were preset by the team as a form of censorship, but it was theoretically possible to display any image on the clothing.

Table 2. Statistically significant questions from the Invent Week survey.

Question Number	Question
18	I am confident that I can program a computer to detect a button push and turn on an LED.
21	I am confident that I can create a program to control outputs over the internet.
23	I am certain that I can design a Wearable/IoT technology
24	I am confident that I can effectively communicate a Wearable/IoT product
25	I am confident that I can create/prototype a Wearable/IoT technology
26	I use a step by step process to solve problems.
30	In order to solve a complex problem, I break it down into smaller steps.
37	I feel confident in my ability to evaluate different solutions to one problem.
47	I feel confident in my ability to think about the technology design process.
49	I feel confident in my ability to use data types that are common to procedural high-level languages: Boolean, integer, real, char, string, date.
50	I feel confident in my ability to describe what an algorithm is and give an example.
57	I feel confident in my ability to describe the difference between digital and analogue inputs.
58	I feel confident in my ability to use and control digital inputs and outputs.
59	I feel confident in my ability to measure and store data returned from an analogue input.
60	I feel confident in my ability to develop a program that utilizes digital and analogue inputs.
61	I feel confident in my ability to design automated applications.

Pre- and Post-Workshop Comparison

The students were administered one survey before the week began, and a follow-up survey after their presentations on the final day. All of the questions were the same, except that the follow-up asked the students for their group names and roles that they played within the groups. Apart from these short answers, all of the other questions were evaluated with a 5-point Likert Scale where 1 indicated a lower confidence on the given topic, and 5 indicated a higher confidence. There were a total of 63 questions. After conducting a t-test on the data, although all questions showed increases, only 16 of the questions showed a statistically significant improvement from the beginning to the end of the week, as displayed in Table 2.

Questions 37 to 61 in Table 2 were drawn directly from the learning outcomes section for the Leaving Certificate Draft Specification (NCCA, 2017a). Of interest, questions 35 to 63 have statistically significant improvements for the students, which is 9 of the 29 questions that address those learning

outcomes. These are encouraging results, as this was the focus for Invent Week. Figure 1 illustrates the improvements.

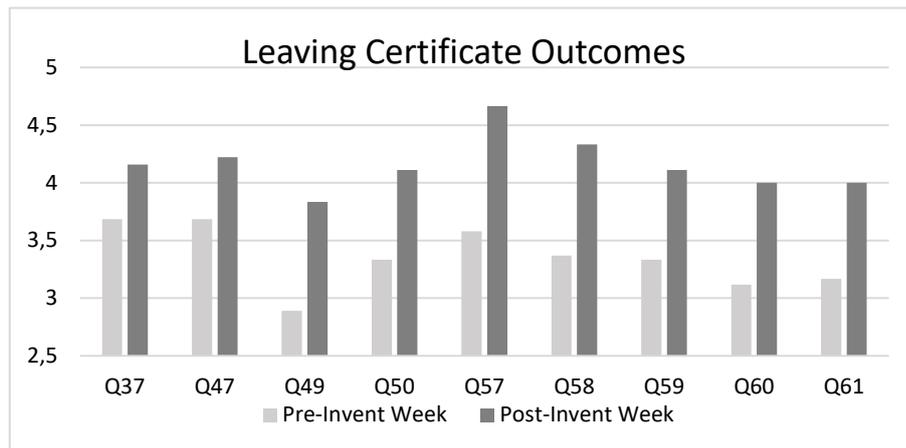


Figure 1. The statistically significant improvements in learning outcome areas for the Leaving Certificate.

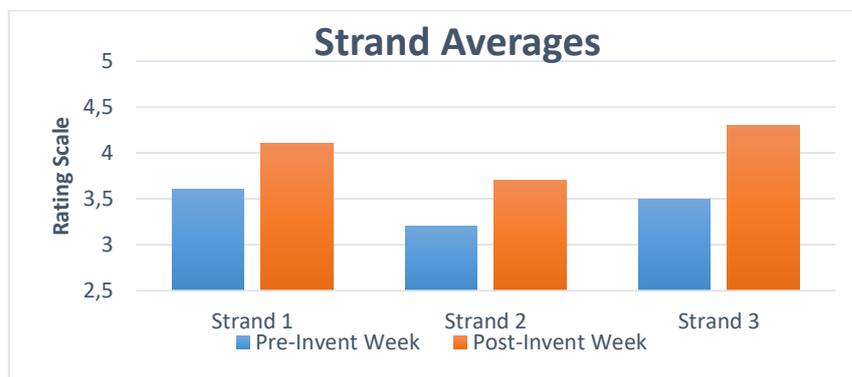


Figure 2. Displays the increases in the mean score of questions related to each Strand.

Furthermore, the Leaving Certificate learning outcomes are broken into 3 sections, or “strands” which are intended to be intertwined throughout learning activities to form a metaphorical “rope” of interconnected skills and abilities. These strands are classified as follows: Strand 1 is “Overarching Practices and Principles”, Strand 2 is “Cross-Cutting Core Concepts”, and Strand 3 is “Computer Science in Practice”. Different learning outcomes are targeted in each strand. This strand includes four different and highly specific projects, such as information systems, analytics, modeling/simulation, and embedded systems. Invent Week focused on the embedded systems aspect of Strand 3 as it was a logical fit with the IoT theme. Questions 57-61 applied directly to this portion of strand 3, and accordingly every one of them showed a statistically significant improvement over Invent Week. Figure 2 illustrates the improvements within each of the strands as an aggregate.

Most importantly, we found a statistical significance in the improvement within the aggregated Strand 3. This was our target goal throughout the week. Additionally, we found a statistical significance in the improvements of Strand 1 (Practices and Principles). The entire statistical report is in Table 3.

Table 3. The statistics behind each Leaving Certificate Outcome Strand over Invent Week.

	Pre-Invent Week (M)	Post-Invent Week (M)	T value	df	P value
Strand 1	3.6	4.1	-2.661	17	0.016
Strand 2	3.2	3.7	-1.910	14	0.077
Strand 3	3.5	4.3	-3.828	15	0.002

Lastly, Figure 3 depicts the increases of the Strand 3 outcomes individually. This is a significant chart because it displays the central topic of the week and has major implications for project impact on learning, and optimizations that teachers can make to target different skills. The dramatic improvements and significance of the relevant Strand 3 topics indicate that a similar learning experience can provide a massive learning opportunity for students in a given central topic.

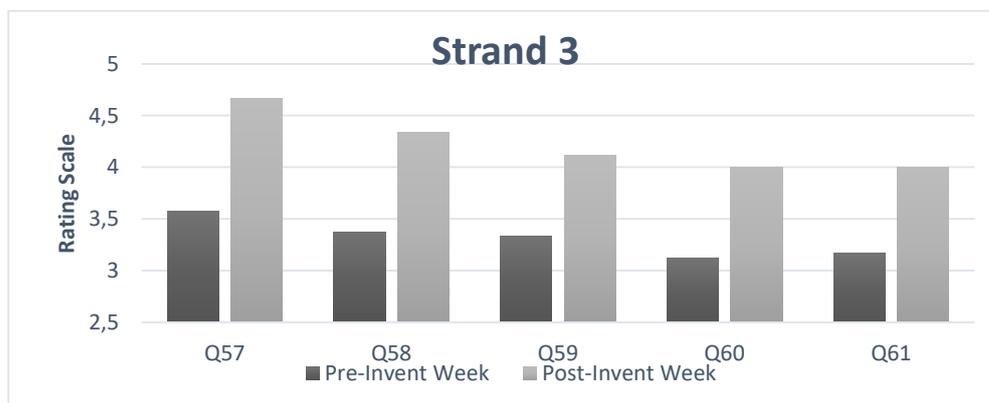


Figure 3. Displays the increases in topics related strictly to embedded systems and Internet of Things.

Discussion

The pre and post questionnaires indicated growth in all surveyed areas. These include self-efficacy, understanding of careers in computer science, 21st Century skills (team work, presenting etc.), inputs and outputs, embedded systems, coding, debugging and more. The largest and most critical area of growth was in embedded systems. The self-reported confidence levels of the students within these topics, coupled with the robust prototypes that they created serve as sufficient evidence for their knowledge, capability and learning. Furthermore, the differences from time one and two suggest a self-perceived growth in confidence to practice CS from their participation in Invent Week. This growth in practical experience is a direct result of the “learning by doing” pedagogical design.

The results from the surveys suggest that the students received a broad learning experience from the hackathon-like delivery method. This is evidenced by the fact that there were only increases in scores i.e. none of the averages decreased because of Invent Week. Beyond that, 16 of the questions returned statistically significant increases. Importantly, all of the targeted questions for the embedded systems ALT showed significant increases but, outside of those, 11 other questions also achieved significant improvement. The programme managed to encourage a wide array of skill sets, including 21st Century skills, even though those were not the primary objective of the learning experience. Those were rather a favourable consequence of the constructivist/constructionist pedagogical approach.

Conclusions

Limitations

The small participation numbers is a limitation for our study, however it is comparable to a common classroom size. The high level of mentor involvement in the process makes the hackathon-like difficult to scale up beyond the size we’ve already achieved. Perhaps a classroom could call upon external technical help via remote mentoring with IT professionals.

The elevated prowess of the students also posed a limitation for us. The pre-survey indicates that everyone involved had a decent-to-good understanding of all of the topics we surveyed before the programme commenced. Thus it is hard to track how Invent Week would assist a student with a lower level of knowledge at the beginning. The students were hand-picked based on their expressed interest and performance in prior Bridge21 activities, this may not be strictly comparable with a school setting.

A second issue that arose from this hand-picking was the relative uniformity of the student skills. Not only were they all high at the beginning, many of them gravitated around the same level. This may not be representative of a post-primary school classroom either, considering that many different backgrounds could be better represented without the selection process. This could lead to greater diversity in other settings, however, as skills beyond computer science are helpful and constantly developed through this 21st Century learning method.

Finally, we are operating under the assumption that high self-reporting on confidence translates into an elevated level of actual learning in a given area. The related literature supports this assumption, but further research must be conducted.

Final remarks

There has been only a limited amount of work done measuring the effects of hackathons as pedagogical approaches (Byrne et al., 2017; Duhring, 2014). There is a need to further this field of research as teaching methods move to accommodate a more technology and future oriented focus. Furthermore, Ireland's introduction of Computer Science is an example of a global trend towards CS education for pre-university students, which means that exploring a variety of approaches to teaching the topic will be more important than ever. This paper addressed both of those problems by measuring the amount of self-perceived learning that occurred because of the learning activity, while also putting these measurements in terms of the new Irish Leaving Certificate learning outcomes.

The outcomes of the pre and post questionnaires indicate that the students grew academically due to Invent week. They apparently left the week more prepared for the Leaving Certificate than they were when they entered it. It is encouraging to note the measurable success of Invent week, because this means that there is at least one proven method for project facilitation relevant to Irish Computer Science course. This model could be adapted to target learning outcomes at different levels and in different jurisdictions.

It will be important to attempt a similar approach with other groups of students, more often, using different curriculum outcomes and hopefully with larger groups to confirm that this approach is reliably and globally applicable.

Other research that should follow as a result of this would include tracking each individual's role within the group and seeing how that impacted their specific areas of learning. Also, more methods of project facilitation such as Agile (Kastl et al., 2016; Rico & Sayani, 2009) or SCRUM (Scharf & Koch, 2013) should be considered and measured in order to discover new and effective pedagogical methods. Finally, methods currently in development (such as the hackathon-like activity) should be compared to more traditional approaches to evaluate the relative effectiveness.

In conclusion, allowing students to be immersed in a scaffolded, but ultimately open-ended hands on project gave them a chance to develop their technical and domain knowledge along with their interpersonal and 21st Century skills.

References

- Barker, B., Melander, J., Grandgenett, N., & Nugent, G. (2015). *Utilizing Wearable Technologies as a Pathway to STEM*. Paper presented at the Society for Information Technology & Teacher Education International Conference.
- Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilensky, U. (2017). All Roads Lead to Computing: Making, Participatory Simulations, and Social Computing as Pathways to Computer Science. *IEEE Transactions on Education*, 60(1), 59-66.
- Briscoe, G., & Mulligan, C. (2014). *Digital Innovation: The Hackathon Phenomenon*. Retrieved from <http://www.creativeworkslondon.org.uk/wp-content/uploads/2013/11/Digital-Innovation-The-Hackathon-Phenomenon1.pdf>

- Byrne, J. R., Fisher, L., & Tangney, B. (2015). *Computer science teacher reactions towards raspberry Pi Continuing Professional Development (CPD) workshops using the Bridge21 model*. Paper presented at the Computer Science & Education (ICCSE), 2015 10th International Conference on.
- Byrne, J. R., O'Sullivan, K., & Sullivan, K. (2017). An IoT and Wearable Technology Hackathon for Promoting Careers in Computer Science. *IEEE Transactions on Education*, 60(1), 50-58.
- Cooper, R., & Heaverlo, C. (2013). Problem Solving And Creativity And Design: What Influence Do They Have On Girls' Interest In STEM Subject Areas? *American Journal of Engineering Education (AJEE)*, 4(1), 27-38.
- Dede, C. (2010). Comparing frameworks for 21st century skills. *21st century skills: Rethinking how students learn*, 20, 51-76.
- Duhring, J. (2014). *PROJECT-BASED LEARNING KICKSTART TIPS: Hackathon Pedagogies as Educational Technology*. Paper presented at the National Collegiate Inventors and Innovators Alliance. Proceedings of Open, the Annual Conference.
- Fee, S. B., & Holland-Minkley, A. M. (2010). Teaching Computer Science Through Problems, Not Solutions. *Computer Science Education*, 20(2), 129-144.
- Johnson, L., Brown, S., Cummins, M., & Estrada, V. (2012). The Technology Outlook for STEM+ Education 2012-2017: An NMC Horizon Report Sector Analysis.
- Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014). A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)*, 14(1), 1.
- Kafai, Y. B., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge.
- Kafai, Y. B., Rusk, N., Burke, Q., Mote, C., Peppler, K., Fields, D., . . . Elinich, K. (2014). *Motivating and Broadening Participation: Competitions, Contests, Challenges, and Circles for Supporting STEM Learning*. Paper presented at the Proceedings of the 11th International Conference of the Learning Sciences: Learning and Becoming in Practice.
- Kastl, P., Kiesmüller, U., & Romeike, R. (2016). *Starting out with Projects: Experiences with Agile Software Development in High Schools*. Paper presented at the Proceedings of the 11th Workshop in Primary and Secondary Computing Education.
- Lawlor, J., Conneely, C., Oldham, E., Marshall, K., & Tangney, B. (2018). Bridge21: teamwork, technology and learning. A pragmatic model for effective twenty-first-century team-based learning. *Technology, Pedagogy and Education*, 1-22.
- Lawlor, J., Marshall, K., & Tangney, B. (2015). Bridge21—exploring the potential to foster intrinsic student motivation through a team-based, technology-mediated learning model. *Technology, Pedagogy and Education*, 1-20.
- Nandi, A., & Mandernach, M. (2016). *Hackathons as an Informal Learning Platform*. Paper presented at the Proceedings of the 47th ACM Technical Symposium on Computing Science Education.
- NCCA. (2014). Short Course - Coding Retrieved from <http://www.curriculumonline.ie/Junior-cycle/Short-Courses/Coding>
- NCCA. (2017a). Draft Leaving Certificate Computer Science Specification. Retrieved from <https://www.ncca.ie/media/3184/lc-computerscience.pdf>
- NCCA. (2017b). The Leaving Certificate (Established). Retrieved from <https://curriculumonline.ie/Senior-cycle/Senior-Cycle-Subjects/Computer-Science>
- Nugent, G., Barker, B., Grandgenett, N., & Welch, G. (2014). *Robotics camps, clubs, and competitions: Results from a US robotics project*. Paper presented at the Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education Padova (Italy) July.

- O'Donovan, D. (2015). Enquiry Based Learning at Bridge21 Retrieved from <https://sites.google.com/site/enquirybasedlearningatbridge21/home>
- Przybylla, M., & Romeike, R. (2014). Physical computing and its scope-towards a constructionist computer science curriculum with physical computing. *Informatics in Education*, 13(2), 225.
- Rico, D. F., & Sayani, H. H. (2009). *Use of agile methods in software engineering education*. Paper presented at the Agile Conference, 2009. AGILE'09.
- Roberts, A. (2012). A justification for STEM education. *Technology and Engineering Teacher*.
- Romeike, R., & Göttel, T. (2012). *Agile projects in high school computing education: emphasizing a learners' perspective*. Paper presented at the Proceedings of the 7th Workshop in Primary and Secondary Computing Education.
- Rychen, D., & Salganik, L. (2005). *The definition and selection of key competencies: Executive summary*. OECD.
- Sanders, M. (2009). STEM, STEM Education, STEMmania. *Technology Teacher*, 68(4), 20-26.
- Scharf, A., & Koch, A. (2013). *Scrum in a software engineering course: An in-depth praxis report*. Paper presented at the Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on.
- Skirpan, M., & Yeh, T. (2015). *Beyond the Flipped Classroom: Learning by Doing Through Challenges and Hack-a-thons*. Paper presented at the Proceedings of the 46th ACM Technical Symposium on Computer Science Education.
- Tangney, B., & Bray, A. (2013). *Mobile Technology, Maths Education & 21C Learning*. Paper presented at the QScience Proceedings.
- Tangney, B., Oldham, E., Conneely, C., Barrett, S., & Lawlor, J. (2010). Pedagogy and processes for a computer programming outreach workshop—The bridge to college model. *Education, IEEE Transactions on*, 53(1), 53-60.
- Vygotsky, L. S. (1978). *Mind in society* (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman, Eds.): Cambridge, MA: Harvard University Press.
- Williams, J. (2011). STEM education: Proceed with caution. *Design and Technology Education: An International Journal*, 16(1).
- Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE software*, 17(4), 19-25.

EduRobot Taxonomy and Papert's Paradigm

Dave Catlin, *dave@valiant-technology.com*

Valiant Technology Ltd., UK

Martin Kandlhofer, *mkandlho@ist.tugraz.at*

Graz University of Technology, Austria

Stephanie Holmquist, *sh2@usf.edu*

University of South Florida, USA

Andrew Paul Csizmadia, *A.P.Csizmadia@staff.newman.ac.uk*

Newman College, UK

Julian M. Angel-Fernandez, *Angel-Fernandez@acin.tuwien.ac.at*

Technische Universität Wien, Austria

John-John Cabibihan, *john.cabibihan@qu.edu.qa*

Qatar University, Qatar

Abstract

Seymour Papert was the first person to suggest using robots in education. And for nearly 50 years his constructionism principles guided their application in school. Over the last few years, a multitude of new robots have become available. Do they all meet Papert's Paradigm? What of those that don't? In this paper, we further develop the EduRobot Taxonomy which puts an ever-increasing number of education robots into order. We'll examine what they've got in common and how they differ: we'll also see if they all comply with Papert's ideas and what does it mean if they don't.

Keywords

education robots; EduRobot taxonomy; build bots; use bots; social robots

Introduction

In 1969 Seymour Papert set Mike Paterson, a visiting graduate student from England, the task of writing a specification for a Turtle robot (Paterson, 1969). Papert believed the robot enriched Logo, a computer language he'd invented a few years earlier. He'd imagined the Turtle as a 'transitional object', a link between students the computer and some powerful idea.

With the 'birth' of the Turtle in the early 1970s, Papert became the father of educational robots. In the following years a few commercial Turtles became available and the odd non-Turtle education robot, like, Hero found their way into schools. Piaget's constructionist theories underpinned Logo and the Turtle. By 1993 Papert had gradually created constructionism, a variation on Piaget's ideas where students gain knowledge of the world by making 'stuff' for it. As part of this progression he'd looked beyond Turtles and started to explore cybernetic machines by using Lego to build robots (Papert, *The Children's Machine*, 1993). Eventually, this led to his collaboration with the Lego Company, the intelligent brick and Lego Mindstorms.

So Papert had created two types of education robots: those 'out-of-the-box-ready' to explore ideas and those you learn through building them. Both these types of robot shifted the focus from teachers teaching to students learning. Both captured the spirit of intellectual adventure: children learnt to think, to examine ideas – not simply remember what teachers told them. In a review of education robots in special needs education, Catlin and Blamires (In Press) realised that Papert had created a Paradigm in the sense of Kuhn's Paradigm Shifts and scientific revolutions (Kuhn, 1996). Everything about education robots since 1970 embraced what Papert called the 'Spirit of Logo' – effectively Papert's Paradigm.

Catlin and Blamires reviewed work which reported negatively on Papert's efforts. Unlike Popper's falsification theory of science (Popper, 2002), where a single conflicting result is enough to destroy a scientific proposition, Kuhn's Paradigm supports a theory based on probability. More importantly, a Kuhnian Paradigm is a theory, like Newton's Laws or Einstein's Relativity, which changes the way we view the world. Papert's work on robotics and education did this. This paper finds new robots in education which don't comply with Papert's ideas. However, this new perspective can coexist with Papert's propositions in some ways but clashes with them in others.

Until roughly about 2010, a few small companies and Lego supplied the education space with robots. Since then a host of robots have appeared: for example, in January 2010 ten new robots appeared at London's BETT Show. Catlin and colleagues realised a schema grouping similar robots together would help our understanding of their use in education. So they presented an provisional schema called EduRobot Taxonomy version 1.03 at the Robotics in Education (RiE) Conference (Catlin, Kandlhofer, & Holmquist, 2018).

This paper has two objectives. The first is to develop the EduRobot classification further and the second is to draw attention to the paradigm issues mentioned above. We'll start by presenting our methodology and then introduce EduRobot version 1.03. Then we'll look at issues with this version and solutions which lead to EduRobot version 2.01. We'll conclude with a short discussion and conclusions on the work done so far and the next steps.

Method

EduRobot version 1.03 resulted from the expert experience of its authors whom between them have worked with education robots for over 50 years. We chose a team from Europe and the USA to ensure a broad viewpoint. We presented the paper and a poster at the RiE (Robotics in Education) Conference held in Malta, April 2018. Delegates to the conference included about 40 experts from Europe, North America, the Middle East and the Far East.

Following the presentation, half the attendees completed a questionnaire which assessed approval, tested EduRobot, elicited suggestions, and highlighted confusions. This paper incorporates the information from the survey and the direct input from a further three co-authors from Europe and the Middle East. How easily we and the Malta attendees' classified robots tested the effectiveness of provisional schema. For the Malta Conference, we classified 30 robots – enough to show an example of each class. The issues raised by conference delegates and a new analysis from the team questioned some of the original decisions and enables us to present EduRobot version 2.01. This we've tested by classifying a further 30 robots (including those classified by the Malta delegates).

EduRobot Taxonomy

What is an Educational Robot?

We define an education robot with a combination of three of the ten Educational Robotic Application (ERA) Principles: The 'embodiment, Intelligence and Interactive principles' (Catlin & Blamires, 2010). We can summarise these by saying that an education robot is a physical robot, in the same space as the student. It has an intelligence that can support learning tasks and students learn by interacting with it through suitable semiotic systems.

We haven't included virtual robots since they do not offer the same experience as tangible robots in the same space and time as the learner. Sylvia Weir, who'd started working with Papert's Logo team during the early days, pointed out the original developers didn't see a difference between the physical and virtual robots. But, she went on to say – the children did (Weir, 1987). We're not saying that virtual robots have no educational value, but that it's different to physical robots and would need a different classification approach.

What is EduRobot's Focus?

EduRobot focuses on the robot's construction and not its applications or educational value. The ERA Principles deals with these issues. We note you can use a certain class of robot in many ways: you can also use different classes of robots to do tasks of similar educational nature and value. If you try and include these traits in the schema you'll find it hard to distinguish one robot from another.

Why an Education robot Taxonomy?

Given EduRobot doesn't say anything about the education value of robots, what is its purpose. We cite three reasons for EduRobot:

1. It means you can apply research results from one robot to all members of the same class.
2. It gives teachers a way of reviewing and comparing the technicality of different robots available.
3. It provides a way for the web to organise information about education robots.

EduRobot Version 1.03

Figure 1 shows the basic structure of EduRobot.

Type: We identified two basic types of education robots. These align with Papert's ideas as robots you build (Build Bots) and robots you use (Use Bots).

Class: This refers to robots like Turtles, Robot Arms and Robot Kits and so on.

Subclass: It is acceptable to subdivide some classes into distinct subclasses.

Brand: A specific robot product from a supplier.

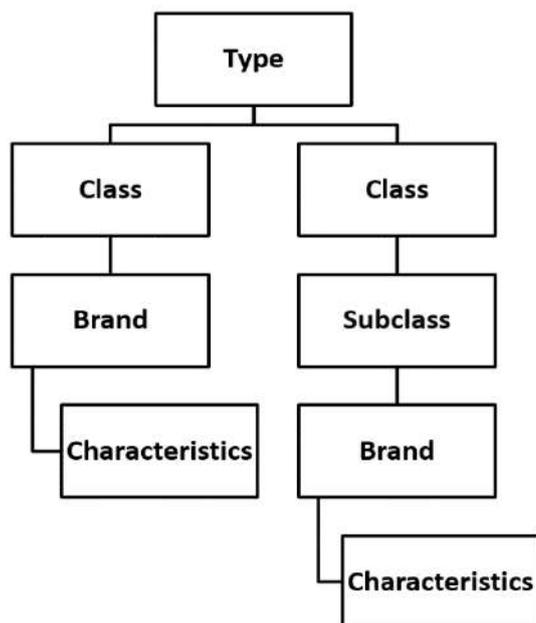


Figure 1 EduRobot Taxonomic Structure.

Characteristics: This is the novelty of EduRobot: animal taxonomies continue with a tiered structure, trying to do this with education robots becomes complicated. Characteristics define a robot's nature and key details that don't change. If a designer adds new features, it enriches the robot, but, usually, it doesn't change its type, class or subclass. You

Build Bots: Lego Mindstorms

Build Systems: Component Parts



A flexible system based on Lego building bricks with special robot parts. Suitable for K5 to University. Students build their robot using either RCX, NXT or EV3 intelligent bricks.

Characteristic Tags:

Locomotion: Wheels, Walking, Crawling, Tracked Drives, Static. **Power:** Batteries. **Command and Control:** Computer, Mobile Devices, Autonomous, Human-Robot Interaction, Robot to Robot. **Communication:** Interface Cable, Bluetooth, Wi-Fi, Infrared. **Sensors:** Digital, Analogue. **Outputs:** Digital, Analogue, Servomotor, Sound Effects, Screen. **Architecture:** Behaviour, Subsumption. **Programming:** Graphical, Text. **Modularity:** Expandable. **Morphology:** Personalised, Component Based.

Use Bot: Roamer

Turtle



Roamer is a Use Bot, but also a platform capable of modular expansion and maker space development. Pre K to 12. Designed to support special needs, gifted and low achievers.

Characteristic Tags:

Locomotion: Wheels. **Power:** Batteries. **Command and Control:** Onboard Keypads, Computer, Tablet, Mobile Phone, Human -Robot Interaction, Human-Computer Interaction, Tangible Computing, Robot to Robot. **Communication:** Interface Cable, Bluetooth, Wi-Fi. **Sensors:** Digital. **Outputs:** Digital, Analogue, Servomotor, Music, Speech, Drawing. **Architecture:** Logo, Behaviour. **Programming:** Icon, Graphical, Text, Tangible Computing. **Modularity:** Expandable. **Morphology:** Personalisation, Modular Changes, Maker Platform.

Figure 2 Examples of Build Bot and Use Bot Classifications.

can think of characteristics as a set of tags that list the robot's important features. The classification allows for a picture and 30-word technical description (See Figure 2).

One point that's not so obvious: a robot can only have one classification. Discovering the platypus caused scientists great consternation. It was a mammal with reptilian characteristics like laying eggs. It took 85 years to agree a classification: it was an egg-laying mammal. The higher taxonomic rank makes the decision. You can build a Turtle out of Lego Mindstorms, but this does not change its nature: it's still a Build Bot. Similarly, you can build and transform a Roamer Turtle into a social robot, but it's still a Use Bot.

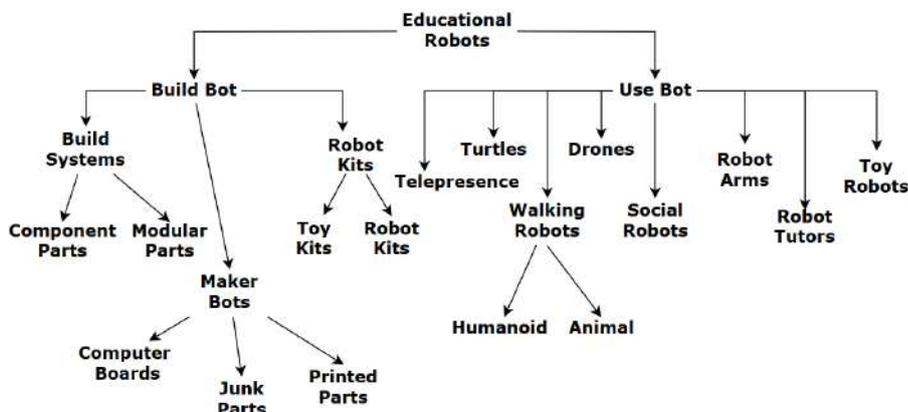


Figure 3 EduRobot Taxonomy version 1.03 Presented to the 2018 Robots in Education Conference in Malta.

Summary of RiE Questions

We received 13 Responses from the 40 delegates. Because several delegates attended from the same institutions this represented about 50% response.

1. Out of the responses 23% strongly agreed with the approach, 38% agreed, 38% felt undecided and 1% disagreed.
2. To the question, "Is two Types of robot enough?" 72% felt we had enough the rest did not answer.
3. Did we need more Classes? 85% said no - one of these felt EduRobot was already too complicated for teachers. Another thought we might need more in the future.
4. Did we need more characteristics? 77% said no and 23% said yes.

Because of the small sample size, these results do no more than encourage us to continue. The richness of the *survey* comes from the comments. They helped us understand the respondent's misunderstandings. For example, the person who disagreed with the approach recanted in conversation when we clarified some of his concerns. Although people said they were happy with the Types, Classes and Characteristics their comments inspired positive changes. We've referenced these in the next section.

Issues with Version 1.03

Social Robots

Seymour Papert once asked the rhetorical question, "Will computers program children, or will children program computers" (Blikstein, 2013) (Papert & Solomon, 1972). With robots that comply with Papert's Paradigm, students use the robot as a tool to express themselves and explore ideas. Students act and the robot responds. With social robots, it's the other way. Autonomous social robots respond to students in pre-set ways: normally the learner cannot change the response. Social robot research records children's reactions to different stimuli. Designer's use this data to make the robot recognise and

respond in apt social and cultural ways. We wondered whether we should classify Social Robots as a “Type Bot”. The answer became clear when we tried to classify social robots as Build or Use Bots. The RiE Poster for version 1.03 showed we could but, it felt forced. Once we gave Social Bots the status of ‘Type’ it allowed us to create a more fitting set of characteristics – which focus on Human-Robot Interactions (HRI).

Social Bot Ontology

In education, Social Bots are autonomous robots that interact and communicate with students using HRI technologies. The interactions aim to embody accepted social and cultural norms (Li, Cabibihan, & Kee Tan, 2011).

Normally social robots can interact with their environment, each other and people, but in an educational context, our main interest is in the student robot rapport. We can classify many of the robots as "social" because of the physical and behavioural ‘affordances’ they display. For example, when students switch on Nao its eyes and ears light up, it stands upright and turns its head towards the user. Table 1 defines the Social Bot Classes for EduRobot version 2.01.

Marvin Minsky described Telepresence as “instruments that will allow us to work remotely” (Minsky, 1980) (Minsky, 2010). His definition included robots in the next room or the Mars Rover. While in principle explorer and bomb disposal robots meet this definition, in education we normally use the noun Telepresence for students projecting themselves into remote social locations. We’ll reserve the class for robots of this definition.

People have used these robots to allow children confined to hospitals to go to school in their place. They control the robot to take part in the lesson. Classmates treat the robots as if it was their friend and the socialisation has a positive effect on the patients’ health (Mills & Laughlin, 2018).

Table 1 Definition of Social Bot Classes

Humanoid	Social robots that look like mechanical people: Example Nao.
Human-Like	Social robots trying to resemble humans: example Kaspar.
Toys	Smart toys with social robot characteristics: example Furby
Animal-Like	Social robots that resemble animals: example Paro
Creatures	Social robots without a form template: example Tega
Telepresence	Robots that students can use to represent themselves in social circumstances: example Pebbles.

Social Bot Characteristics

Table 2 lists the characteristics applicable to social robots. We consider this table an initial effort to characterise the Human-Robot Interaction (HRI) spectrum. The extensive literature highlights the complexity problem. For example, the studies of Fernando Alonso-Martín and colleagues classified human touches into tap, pat, push, stroke, scratch and slap (Alonso-Martín, Gamboa-Montero, Castillo, Castro-González, & Ángel Salichs, 2017). Their work focuses on the human touching the skin of the robot. This forms one part of ‘Recognition’ and one aspect of ‘Gestures’. If you try and define the detail of the characteristics you run into complexity issues which defeat the purpose of the EduRobot enterprise. If necessary we can expand these definitions later after we’ve more experience classifying social robots.

Table 2 Characteristics of Social Bots.

Recognition	The robot detects the presence of a human through its senses.
Acknowledgement	The robot acknowledges the human presence.
Motion	The robot detects and responds to movement.
Gestures	The robot notices and reacts to gestures.
Emotions	The robot senses human emotions and reacts to them.
Conversation	The robot and child communicate through sound.

Programming Characteristics

Our review EduRobot version 1.03 decided we could improve the programming characteristic ontology.

We can program education robots in a variety of ways that meet Papert's aim of "low floor and high ceiling" (Papert, 1993) later refined to "low floor, high ceiling and wide walls" (Resnik & Silverman, 2005). These provide students with easy entry points, but great exploration possibilities. Robots allow teachers to create environments which reflect Bruner's Spiral Curriculum (Bruner, 1960). Young children start with tasks like Roamer's Incy Wincy Spider where all they do is put symbols in the right order, but as their experience and interest grows they can end up coding in professional programming languages. Several robots provide rich educational environments by offering different ways for students to program them. Table 3 lists the characteristics of programming languages used with education robots.

Malta Feedback

David Miller from the KISS Institute raised the issue of Mechanical computing in his historical presentation which featured the robot 'Scarecrow'. We've changed Mechanical to Mechatronics which embraces programming by arranging mechanical, electrical and electronic parts. This made us rethink the 14-in-1 Solar Robot shown on the EduRobot version 1.03 Poster; we'd decided it failed to qualify as an education robot because it didn't meet the ERA Intelligence Principle. The Mechatronic characteristic revises this decision because you 'program' it by rearranging the parts. You can't do this with toy kit robots, for example, Owi's Solar Wild Boar Robot doesn't qualify because you can only build it one way.

Robot Architecture

Reflective analysis showed the characteristic we called 'Architecture' in version 1.03 needed improvement; the original effort was too detailed. EduRobot needs to find a balance between providing detail and giving a clear understanding of the robot's nature.

Table 3 Programming Characteristics for EduRobot version 2.01

Programming	This characteristic explains the different forms of software used to tell robots what to do and how to behave.
Icon	Symbols represent robot actions like move-forward. They're often used on keyboards mounted on the robot allowing students to program the robot directly. Example software: RoamerWorld Graphics and example robot: Roamer.
Tile-based	This approach shows instructions as tiles, each with an image to depicting and instruction. Learners create programs by putting the tiles in order and in a horizontal line. Example software: Edware and example robot: Edison.
Block-based	'Jigsaw' like blocks fit together in ways that create programs and reduces the chances of code bugs. Example software: Scratch and Blockly and example robot: We Do Robot Kit.
Text-based	Traditional text-based programming languages. Software examples: Java, Python. Example robot: Cozmo
Graphical	The program is a graphical representation showing actions in sequence for example flow diagrams and Finite State Machines. Examples: Flowol Software (Flow Diagrams) with Vex IQ robots, and Finite State Machines with TI-RSLK robots.
Mechatronics	Students create programs by organising mechanical, electrical and electronic parts in different arrangements. Early examples include Babbage's Machines and Braitenberg robots. Example robot: Scarecrow.
Tangible	This involves students arranging objects in a physical environment. The robot gets instructions from those objects. Example robot: Cubetto
Teaching	The learner shows the robot how to do the task. The robot remembers the movement and reproduces it. Example robot: Little Robot Arm

Robot architecture manages complex tasks in different environments based on a software approach. It comprises of Artificial Intelligence (AI) and software engineering. AI include problem abstraction, knowledge representation, decision making and behaviour execution. Software engineering includes programming, interfaces, data organization and transfer. Based on three ingredients (sense, plan and act) various authors propose three architectures (Figure 4): reactive, deliberative and hybrid (Siciliano & Khatib) (Murphy, 2000).

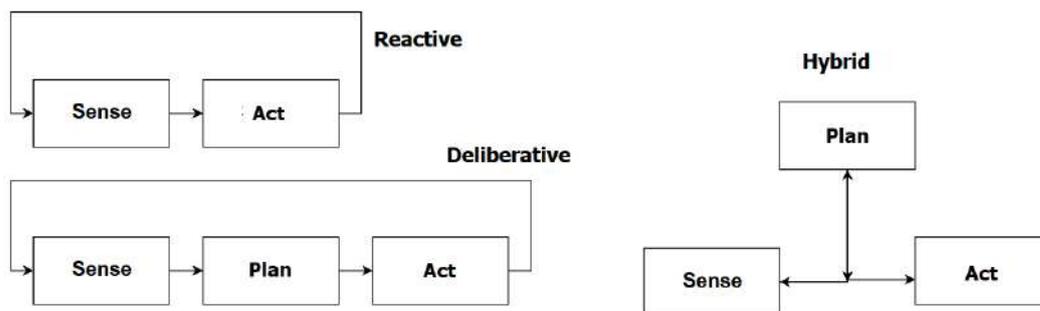


Figure 4 Robotic Architectures

Architecture defines a technical strategy used by the robot to perform its tasks. However, we can look at the issue from an educational perspective. In EduRobot version 1.03 we address the issue: can you call a machine a robot only if it has sensors? While insisting it must, may have merit in a technical debate, it's not essential for an effective education tool. Robotics is a subset of control engineering and in that discipline, you draw a boundary around the system under consideration and study what happens within its confines. If you draw the boundary to encompass machine and learner, you have a system which involves Papert's key ideas behind the Turtle robot. The child sets goals plans and fulfils the plan by programming the robot. The child sees the mistakes and debugs the program. 'Playing Turtle' is the second part of Papert's thinking: the child works out what the robot needs to do by imaging themselves to be the robot. We'll call this a direct architecture.

Table 4 Characteristics: Robot Architectures

Architecture	The strategies the robot uses for AI tasks like problem abstraction, knowledge representation, decision making and behaviour execution, and software engineering like programming, data organisation and interfaces.
Direct	A robot where the learner is responsible for the world model, goals and plans. Example robot: Bee-Bot.
Reactive	A direct connection between the robot's sensors and its actuators lacking an internal model of the world. For example the subsumption architecture (Brooks, 1986).
Deliberative	The robot contains planning and reasoning parts with its memory containing a model of the world. This model receives sensor data and decides what action to take and tells the action module to perform them. The model understands and accounts for the robot's long-term and short-term goals. Example robot Shakey (Nilsson, 1984).
Hybrid	A combination of the reactive and the deliberative methods which can preserve different levels of abstraction. A common hybrid architecture is the three-tier robot architecture (Bonasso, 1999).

What Else is New in EduRobot Version 2.01?

Status

Many of the presentations in Malta and similar conferences present robots created by the University for their teaching: schools cannot buy these robots. We, therefore, propose a new characteristic called status (Table 5) which will incorporate the historical characteristic of EduRobot version 1.03.

Table 5 Status Characteristic

Status	Status explains whether schools can get the robot. The status of a robot will change overtime.
Historic	Education robots no longer available.
Academic	Education robots used for teaching within a University and specific to that institution.

Commercial	Robots currently available to schools and teaching institutions.
------------	--

Robot Versions

Over the years Lego issued a few updates to their Mindstorms product. As a rule, brand development doesn't change type, class or subclass – if it does change any of these it's a new brand with its own classification.

We should always classify the latest brand but list the previous versions as an extra to the 30 word description (Figure 6).

Changing Build Bot: Maker Bots – Printed Parts

The Malta survey revealed two robots made using machined and laser cut parts. Rather than inventing a new subclass, we felt it better to broaden the ontology of the Printed Parts subclass. The new definition is: Robots where students make the parts using various methods, for example, 3D Printed Parts, machined and laser cut parts.

One Time Builds

You can only build some robots once, but it is clear building it is a technical experience aimed at developing student skills (example Pololu 3pi). You can also buy some Use Bots built or in a kit form. However, you get the impression the motive for the kit version is to reduce the sales price: the build is too complicated for the age range of the students (example Oh-Bot). The classifier needs to judge from available details.

Marine Robots

The 2018 European Conference on Educational Robotics (ECER)⁴¹ ran in parallel with RiE. It included a Botball competition and challenges staged by the PRIA (Practical Robotics Institute Austria). This includes a submarine challenge. At first, we thought this needed a new class of robot. However, the Shark robot classifies as a Build Bot: Maker Bot – Made Parts. However, we can anticipate a User Bot version one day when we'll add a new User Bot Class.



Figure 5 Lego Robot Classification EduRobot Version 2.01

⁴¹ An international scientific conference for students (aged 10 – 18 years old) to present their research.

EduRobot Taxonomy Version 2.01

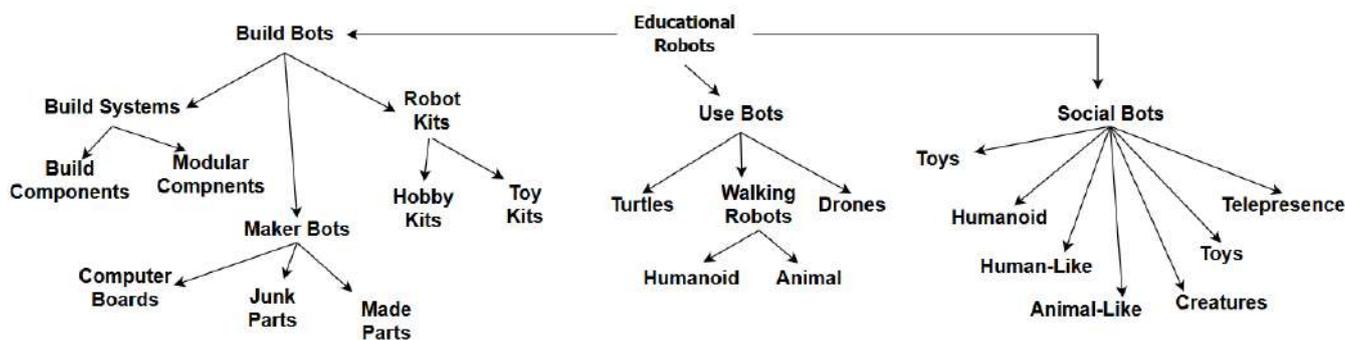


Figure 6 EduRobot version 2.01

Discussion and Conclusions

We feel EduRbot is developing satisfactorily. The response at RiE was positive and supportive and confirmed the need. However, a few issues still seem tentative and the true test of the proposal will come when people try and use it to classify new robots or find the classifications of robots they know.

Following the request of several people at the RiE Conference, we've set up a site for the EduRobot (www.about-educational-robots.com). This will include the latest brands. Once the site is live we plan to contact education robot makers and ask them to classify their products. We expect to adjust EduRobot during this phase.

Papert's work has so successfully justified the role of robots in education, it's a shock to find the social robots do not live up to expectation. This discovery highlights the need for a more detailed study. Can we find an underlying justification for this technology, or can we find a way of using the technology that helps the robots comply with Papert? This is the subject of a future investigation.

References

- Alonso-Martín, F., Gamboa-Montero, J. J., Castillo, J. C., Castro-González, A., & Ángel Salichs, M. (2017, May 16). Detecting and Classifying Human Touches in a Social Robot Through Acoustic Sensing and Machine Learning. (Xiaoning Jiang, & Chao Zhang, Eds.) Sensors. Retrieved May 11, 2018, from <http://www.mdpi.com/1424-8220/17/5/1138/htm>
- Blikstein, P. (2013). Seymour Papert's Legacy: Thinking About Learning, and Learning About Thinking. (Staford Graduate School of Education) Retrieved May 15, 2018, from Transformative Learning Technologies Lab: goo.gl/Kw4NkY
- Bonasso, P. (1999). Issues in providing adjustable autonomy in the 3T architecture. Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy.
- Brooks, R. (1986). A robust layered control system for a mobile robot. IEEE journal on robotics and automation, 2(1), 14-23.
- Bruner, J. (1960). The Process of Education. Cambridge, Ma: Harvard University Press.
- Catlin, D., & Blamires, M. (2010). The Principles of Educational Robotic Applications (ERA): A framework for understanding and developing educational robots and their activities. Constructionism 2010. Paris: Proceedings of Constructionism 2010. Retrieved May 15, 2018, from goo.gl/N7z84k
- Catlin, D., & Blamires, M. (In Press). Designing Robots for Special Needs Education. In L. Daniela, & M. D. Lytras (Eds.), Technology, Knowledge and Learning Special Issue on Education Robotics for Inclusive Education. Springer.

- Catlin, D., Kandlhofer, M., & Holmquist, S. (2018). EduRobot Taxonomy:: A Provisional Schema for Classifying Educational Robots. *Robots in Education*. Malta. Retrieved May 14, 2018, from goo.gl/2tdGTh
- Kuhn, T. S. (1996). *The Structure of Scientific Revolutions* (3rd ed.). Chicago: University of Chicago Press.
- Li, H., Cabibihan, J., & Kee Tan, Y. (2011, November 9). Towards an Effective Design of Social Robots. *International Journal of Social Robots*, pp. 333-335. Retrieved May 15, 2018, from goo.gl/dfn8ys
- Mills, J., & Laughlin, E. (2018). In the beginning.... Retrieved May 15, 2018, from Telepresence Robots.com: <https://telepresencerobots.com/beginning>
- Minsky, M. (1980, June). Telepresence. *Omni magazine*, June. *Omni Magazine*.
- Minsky, M. (2010, September). Telepresence: a manifesto.,. *Spectrum*.
- Murphy, R. (2000). *Introduction to AI robotics*. (R. (. Murphy, Ed.) MIT Press.
- Nilsson, N. J. (1984). Shakey the Robot. Retrieved May 15, 2018, from SRI International's Artificial Intelligence Center: <http://www.ai.sri.com/shakey/>
- Papert, S. (1993). *The Children's Machine*. New York: Basic Books.
- Papert, S., & Solomon, C. (1972, April). Twenty Things to Do with a Computer. *Educational Technology*. Retrieved May 15, 2018, from goo.gl/xsrWDG
- Paterson, M. (1969, August 3). LOGO Ambulatory Executor – Turtle Robot. Retrieved May 1, 2018, from goo.gl/VK2DMv
- Popper, K. (2002). *The Logic of Scientific Discovery* (2nd ed.). New York: Routledge.
- Resnik, M., & Silverman, B. (2005). Some Reflections on Designing Construction Kits for Kids. Retrieved May 15, 2018, from <https://web.media.mit.edu/~mres/papers/IDC-2005.pdf>
- Siciliano, B., & Khatib, O. (Eds.). (n.d.). *Springer Handbook of Robotics*. Springer Science and Business Media.
- Weir, S. (1987). *Cultivating Minds: A Logo Case Book*. New York: Harper Row.

Analysis of Constructive and Cognitive Activities of Participants in Online Competitions in Computer Science

Anton Chukhnov, *septembreange@gmail.com*

Saint Petersburg Electrotechnical University “LETI”, Russia

Sergei Pozdniakov, *pozdnkov@gmail.com*

Saint Petersburg Electrotechnical University “LETI”, Russia

Ilya Posov, *iposov@gmail.com*

Saint Petersburg Electrotechnical University “LETI”, Saint Petersburg State University, Russia

Athit Maytarattanakhon, *seaay2499@gmail.com*

Saint Petersburg Electrotechnical University “LETI”, Russia

Abstract

The paper discusses a certain type of competitions based on distance interaction of a participant with simulation models of concepts from discrete mathematics and computer science. The “Construct, Test, Explore” competition, developed by the authors, is chosen to be a representative of such competitions. One of the features of this competition is that all its tasks are accompanied not only with simulation models and tools to manipulate the model’s objects, but also with a hierarchical criteria system, that defines an objective function to be optimized while solving a task. The presence of such a criteria system allows for treating a task’s subject as a set of several tasks of different complexity (2–4 tasks). Each criterion has means to assess partial solutions, that provide permanent feedback for a participant. The transition of a participant from optimizing one criterion to optimizing the next one, means that he or she has already fully understood an idea corresponding to a former criterion. The work is supported by the Russian Foundation for Basic Research (Project No. 18-013-01130).

Keywords

olympiad; computer science; discrete mathematics; electronic manipulator; CS competition; mathematical thinking

Background and theory

One of the important issues in arranging for mental activity in online (distance) competitions is to automate an analysis of mental actions, that are performed by participants while they solve tasks. Some information about the characteristics of mental actions may be obtained by analyzing competitions based on multiple choice questions (Yagunova, 2016). But this information is not enough to draw conclusions about the degree of understanding, because the correct answer could be selected accidentally, or by means of indirect reasoning that have nothing to do with an idea presented in a task. Moreover, the very form of a task with a multiple choice question is very limiting. Authors think, that much more information about mental processes may be gained by analyzing solutions of constructive tasks. Constructive tasks are convenient because they may be automated in a straightforward way. They may be stated inside an informational environment, built on computer tools and simulation models. Computer tools and simulation models allow for analyzing constructive solutions of participants. These solutions are very diverse and provide interesting information about how strongly have participants formed understanding of concepts, on which the task is based.

Also, while a participant work with a tool, rich information about his or her actions may be collected. However, the problem of actions logs analysis aimed to get mental acts is quite complicated and does not have full solution by now (Gibson, etc., 2016). So we do not consider logs analysis in this work. At the same time, the experience of working with imitation models has been gained (Honey, etc., 2010;

Potkonjaka, 2016), their important role in tasks solving has been demonstrated (Baker, etc., 2008). It has been proved that the work with black-box models aids in better conceptualization of concepts: the work (Hosein, etc, 2008) demonstrates that “students using the black-box did better on the constructive tasks because of their increased explorations”. This work also shows that “students with low maths confidence resorted to using real-life explanations when answering tasks that were application related”. For this reason, this paper investigates subjects that have clear real-life sense and allow for transition to serious theoretical problems starting from understandable tasks that can be solved by participants only by common sense.

The work (Mislevy, 2011) demonstrates that it is important for concepts interpretations chosen for tasks statements to be obvious and natural. This approach corresponds to the idea of engineering design developed in the project WISEngineering (Chiu, 2013).

The type of feedback is important. The article (Attali, 2015) explores several feedback types: no feedback (NF), immediate knowledge of the correct response (KCR), multiple-try feedback with knowledge of the correct response (MTC), or multiple-try feedback with hints after an initial incorrect response (MTH), and determines that the latter is the most efficient. That is why this paper attaches great importance to the analysis of partial solutions and the automation of responding to them.

Not without interest is a study and development of such types of educational activities, that combine learning of new scientific and technical ideas with an ability to assess results of learners’ mental activities without specially set up additional assessments of obtained knowledge and skills. Such steps were taken in works (Pozdniakov, 2012, 2013; Posov, 2013; Yagunova etc., 2016; Akimushkin etc., 2015). This paper proceeds with a study of these means and abilities.

Methodology

We use a method of *software supported subject tasks* (Pozdniakov et al., 2013). By *subject* task we mean a task with some understandable real-world statement that does not need any specific knowledge to understand it and to make at least first steps in the solution. By *software supported* we mean that a task is accompanied with a computer tool, that demonstrates the statement and allows for searching for a solution (Honey et al., 2010, Hosein et al., 2008). Thus, such tasks have to be constructive, and a tool exposes their constructive nature.

The approach presented in this paper is connected with the study of possible feedback for partial solutions of subject tasks. To provide a feedback, a system should have access to information about students actions. Mental actions of a student are accompanied with actions performed with the tool, and a potential ability to analyze and to control the solution search activity of a student arises [8].

To use the information about a work of a student within a computer tool we propose a method of a criteria system [9]. These criteria assess partial solutions. The hierarchy of criteria is built in the way that every progress of a student is somehow evaluated with a usually numerical value. It allows for comparison of partial solution based on how close are these solutions to the full solution of the task. Criteria of higher levels are intended to reveal harder aspects of a subject. Thus, subject tasks may be considered both as quite simple (over the lower level criteria) and as olympiad (over higher level criteria). Subject tasks usually either implicitly or explicitly require to find the optimal solution.

Let us look into a technique of designing software supported subject tasks on the example of the graph theory with the visual metaphor based on the idea of constellations. Such interpretation allows for decreasing of an introductory part of a statement because participants must have already been acquainted with the concept of constellations.

The idea of constellations comes from the need to orient oneself in the space quickly. This can be achieved using stars in the sky. One should split the set of bright points-stars into groups, each of which is considered to be a single object. For that purpose people invented an idea of figures called constellations. Each constellation is obtained by connecting several points with segments. After that, the laws of human perception make a human quickly restore segments which are absent on the sky. He or she perceives a sky as a graph, consisting of several connectivity components. Thus, the task to

introduce new constellations can be described in mathematical concepts as a construction of a non-connected undirected graph.

If the first stage is to come up with a metaphor, the second stage is to highlight a goal, that may be effectively supported by a computer tool. Such goal is to split stars into constellations. This task is natural from the context point of view, and thus the explanation of the “split” concept is not needed (see fig. 1).



Figure 1. the user interface of the “constellations” task

To measure the progress of students in achieving of the goal one needs to define criteria, that will assess the solution.

Criteria should not contradict the context of the problem, they should be natural inside this context, while leading a student to master a new idea, a concept or an algorithm. These requirements are satisfied by a splitting of stars, firstly, into different constellations, secondly, into constellations consisting of a fixed or upper-bounded number of stars. The requirement for the constellations to be different is described on the graph theory as a requirement for connectivity components not to be isomorphic. Thus, the important mathematical concept has a natural interpretation inside the chosen metaphor (context). The upper bound on the number of start is also natural, because a human can perceive a group of 5–7 objects as a unit, and he subconsciously splits the greater number of objects into subgroups with less number of elements.

Formalized task statement and conditions of the experiment

The participants of a competition were presented with a task. It was accompanied with a tool that allowed for building configurations of constellations and evaluated numerical values of criteria for each configuration. Here is the problem statement:

This problem asks you to define constellations on a starry sky, connecting stars with segments using your mouse. Segments must not intersect. A constellation should contain at least 2 (4, 5 for other levels) stars and not more than one cycle (a closed poly-line).

The more different constellations you build, the better. How to tell whether two constellations are equal? The stars of equal constellations can be numbered in the way that if a pair of stars is connected in one constellation, then in the other constellation stars with the same numbers are also connected (see fig. 2).

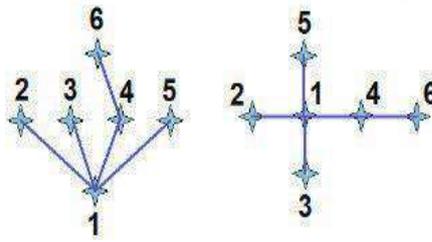


Figure 2. Isomorphic constellation

If you do not succeed to create a new form of a constellation, then the more constellations you have (including same), the better is the result.

Two solutions with the same number of different constellations and the same total number of constellations are compared over the total length of all the segments. A better solution has a smaller sum.

The proposed subject is an example of subjects proposed annually in the “Construct, Test, Explore” competition. Each year schoolchildren get access to three software supported subject tasks. Now they are implemented as web applications, until two years ago they were implemented in Flash. So, participants use their browser to work with tasks. Solutions, that are best for a participant according to the criteria of the task, are saved automatically. Participants may also save any other partial solutions they want, and they can return to saved solutions any time on any browser on any computer.

The work on tasks usually starts in a class and continues at home. The competition lasts for one week, and the time to solve tasks is not bounded during this week.

All participants' solutions have a log of actions made during the search for the solution. After the competition week, the best solutions of each participant are processed and compared between each other according to the same criteria, that were used for a single participant to obtain his or her best solution. So, each participant gets a rank for each task. The rank is a number demonstrating how many better solutions were found by other participants. The lower a rank is, the better is a solution. The participants are split into three groups according to their age, statements for each task for higher ages is more complicated. The ages split is: 0th level: 1–4 grades, 1st level: 5–8 grades, 2nd level: 9–11 grades.

Results and discussion

The qualitative analysis was done by viewing the best solutions of all participants in the order from the first rank to the last. Solutions sometimes contain mistakes that give evidence about not understanding by a participant of some corresponding mathematical concept. The bounds between solutions with and without such mistakes were found. Typical and bounding solutions are demonstrated on figures 3 and 4. This is the qualitative analysis of solutions for the “constellations” problem for the 1st level (5–8 grades). There were 666 solutions in total.

Fig. 3(f) is the last solution with the maximal number of non-isomorphic graphs.

Fig. 4(a) has 8 different graphs, and no isomorphic.

Fig. 4(b) has 6 different graphs, and no isomorphic.

Fig. 4(c) has 5 different graphs, and no isomorphic.

Fig. 4(d) has 5 different graphs, with some isomorphic.

Fig. 4(e) has 2 different graphs, with many isomorphic.

Fig. 4(f) has all graphs isomorphic.

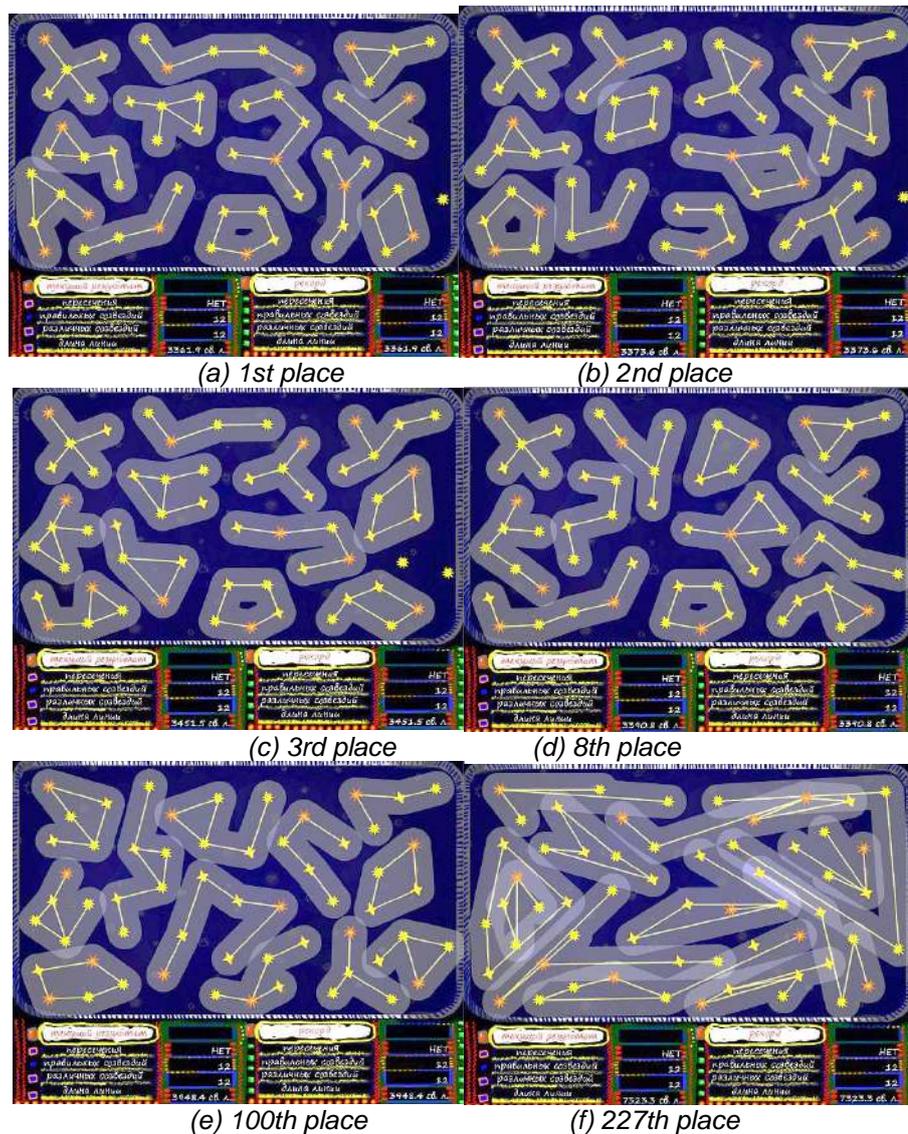


Figure 3. Best solutions of the 1st level according to the 1st criterion: the number of non-isomorphic planar connected components

Figures 3 and 4 present examples of solutions, that demonstrate different levels of diving into the essence of the problem. 227 participants of 666 (34%) managed to find the optimal number of constellations, that is, they managed to build a graph from a maximal number of non-isomorphic planar components, with at least 4 vertexes each (fig. 3a). What is the difference between the beginning of the list and its end? The answer came after the analysis: the difference is in the degree of taking into account the criterion of graph minimality (the minimality of total edges length). This is especially obvious from the last 227th solution, that surely does not take into account the minimality at all. Note, that well known greedy algorithms (for example, the Kruskal's algorithm) are not applicable here because they do not consider the requirement of non-isomorphism of components being built.

The qualitative analysis of solutions demonstrates that during the solution almost all participants (more than 90%) gradually mastered the concept of isomorphism. Only the lowest rated solutions demonstrate participants that were trying to optimize the second criterion (not the main one) about the number of graph components. Note that best solutions do not use all the stars. There were no requirement to build a graph on all vertexes, however only a small fraction of participants were looking for a solution on a subset of stars. The two best found solutions have this form.

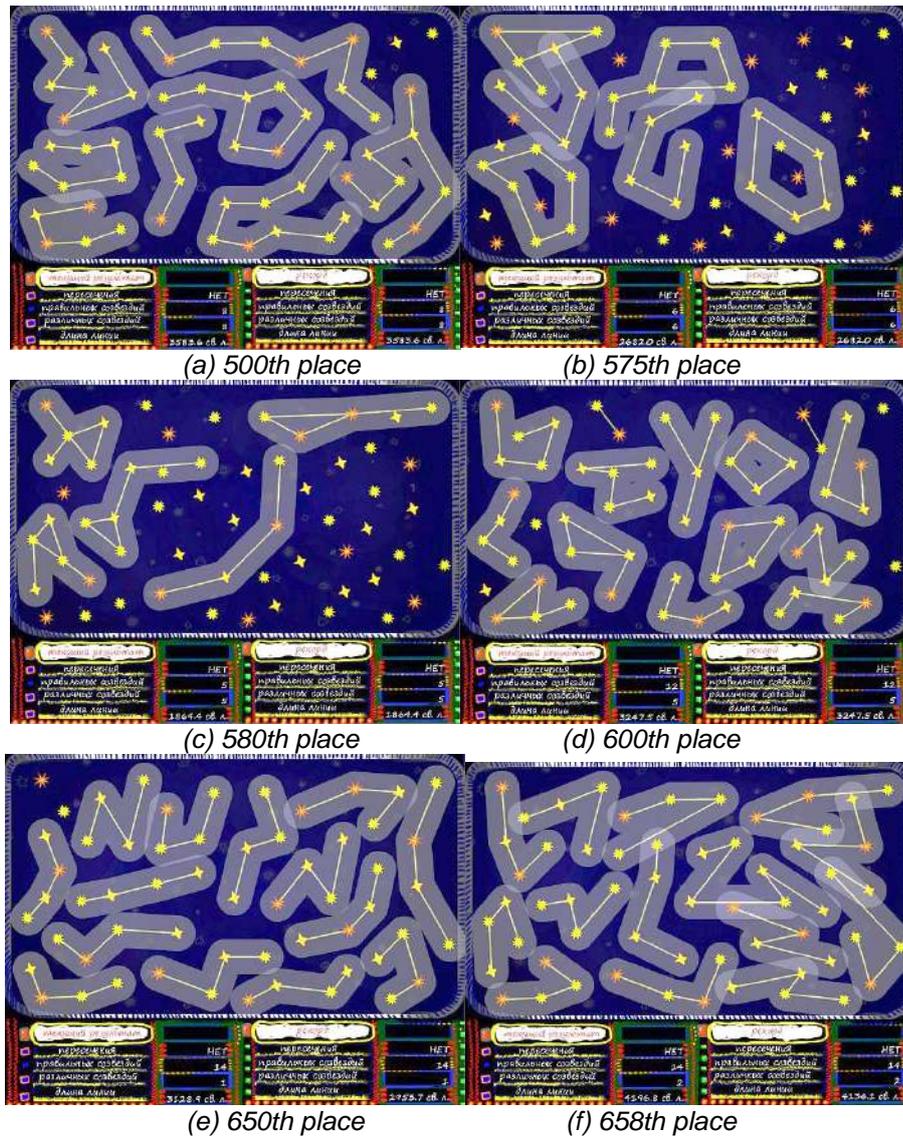


Figure 4. Examples of solutions of the first level with non-optimal solutions according to the main criterion

Statistical analysis of solutions for the “Constellations” task

In the “Constellations” task the best solutions over the main criterion (the number of non-isomorphic graphs) had also the best value of the second criterion (the total number of components) (fig. 5, 6). In this case the good dispersion of results was achieved by the third criterion: the total length of planar graph edges.

At the same time one can see, that the second criterion played the important role for solutions with a few non-isomorphic graphs (see fig. 7). If only the dispersion of results is important, then the first and the third criteria are enough.

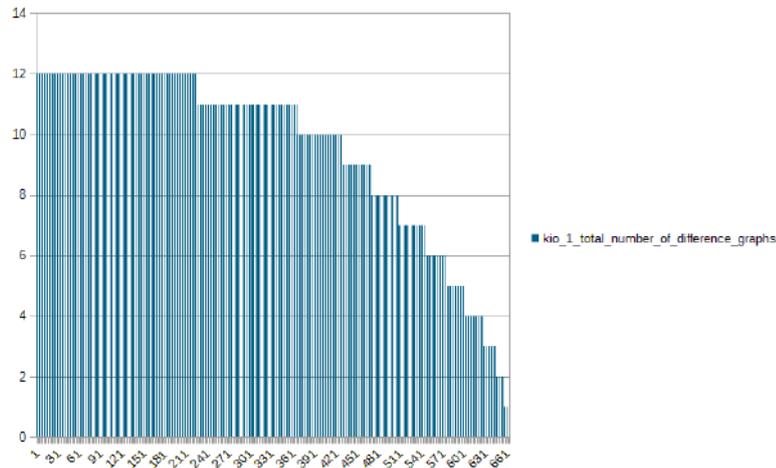


Figure 5. The number of different (non-isomorphic) graphs in the solution of the “Constellations” task on the 1st level. The first decrease from 12 to 11 corresponds to 227 optimal results according to the first criterion.

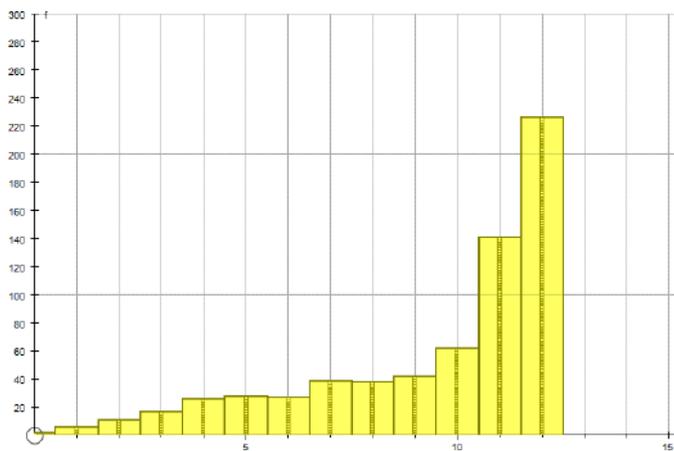


Figure 6. A histogram of the number of different (non-isomorphic) graphs in the solutions of the “Constellations” problem on the 1st level

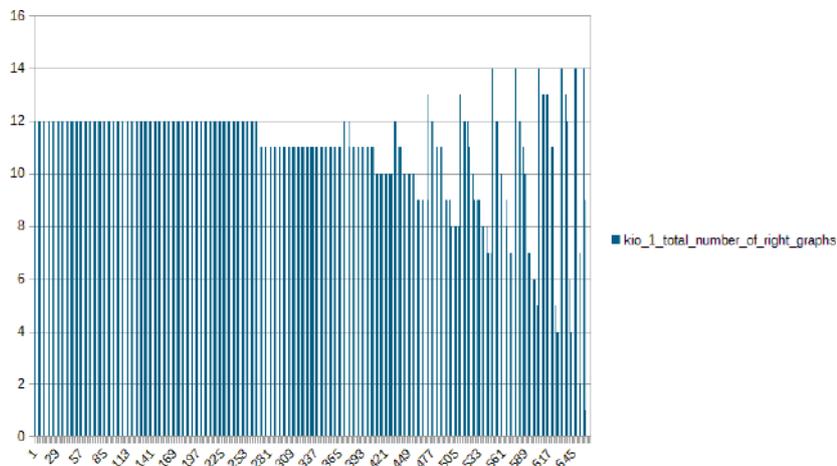


Figure 7. The number of all graphs in the solutions of the “Constellations” problem of the 1st level, results are ordered according to the number of non-isomorphic graphs (the main criterion). Splashes on the right of the plot shows differences by the second criterion for solutions that are neighbors by the first criterion.

However, the second criterion is important for the support of the individual work, because it is much more visually obvious for a participant: the number of built graphs is the first thing obvious for a

participant, and the total length of edges is hard to be estimated visually and thus it plays smaller role in the feedback (fig. 8, 9).

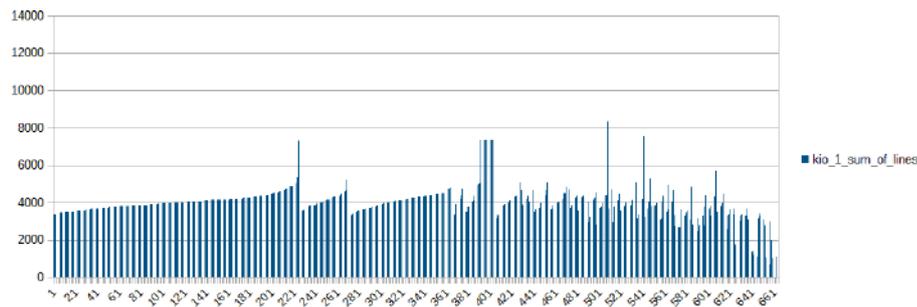


Figure 8. Total lengths of graphs in solutions of the “Constellations” task. Results are sorted according to the number of non-isomorphic graphs and the number of all graphs. There are no horizontal areas on the graph, that means that all solutions of participants are different according to the three criteria.

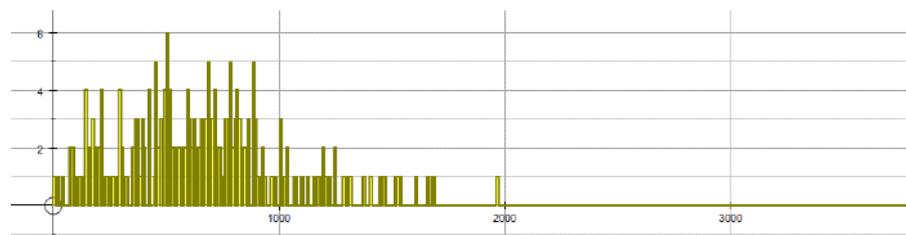


Figure 9. A histogram of deviations of total lengths of graphs from the best total length in the solutions of the “Constellations” task of the 1st level with the maximal number of non-isomorphic graphs (elements are grouped by 10).

Concluding remarks

The qualitative and statistical analysis of results of 6 subjects (the paper contains only one subject) leads to the following conclusion:

1. Practically all participants (99%) got partial solutions of the task, that proves that subjects are understandable for participants.
2. Participants solutions demonstrate that more than 90% of participants not only mastered the concept of graph (vertex, edge) and learned to build graphs with different properties (planar, connected), but also formed an understanding of the isomorphism concept.
3. More than 30% participants managed to optimize a solution over the main criterion, they built a graph with the maximal number of non-isomorphic components. And the majority of them formed an understanding of building minimal spanning trees.
4. The stated task contained elements of an open research problem. The two best found solutions (different) lay in an unobvious region of a solutions set, they do not use all the stars as graph vertexes.
5. The proposed approach to construct tasks and their automated support provides an ability for a vast majority of participants to achieve success in solving proposed task and not to reject searching for solutions despite that problems in the formal statement are quite hard and belong to the class of olympiad problems.
6. The implementation of problems in the form of constructive-research subjects provides wide abilities for building own solutions and individual routes of searching for these solutions. The analysis of results demonstrates that this potential of problems really shows up in the works of students. More of that, it is interesting that participants usually add their own implicit criteria to their solutions, that are not stated in the problem and that may be considered “esthetic”, this shows that it is possible to automatically support elements of creative activity by the proposed types of problems.

7. The proposed approach supposes an ability to state and assess success in solving problems with the optimal solution unknown to problems authors. The results of works sometimes really reveal effects of “discoveries” by participants and the arise of original solutions, that do not lay in the area of common tendencies of searching for the optimal solution. Thus, it confirms an ability to automate a support of creative activity by the proposed means.
8. The proposed approach of organization of online competitions, the usage of several subordinate criteria both for the feedback for the participants, and for the ranking of participants, allows for wide dispersion of results and an ability to objectively compare results of a big number of participants (thousands of different results).
9. The fact that subjects are based on hard and even unsolved problems connected with important ideas of mathematics and informatics, together with the huge amount of sensible solutions (more than 90%), show that it is possible to use proposed automated subjects for popularization of important and hard for understanding theoretical ideas, that do are not included in the school curriculum.

References

- Akimushkin V.A., Maytattanakhon A., Pozdnyakov S.N. Olympiad in theoretical computer science and discrete mathematics in "Informatics in Schools. Curricula, Competences, and Competitions. 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives", ISSEP 2015, Ljubljana, Slovenia, September 28 - October 1, 2015, Proceedings / Springer, LNCS 9378, 2015, p.94-105.
- Attali. Yigal. Effects of multiple-try feedback and question type during mathematics problem solving on performance in similar problems *Computers & Education*, Volume 86 Issue C, August 2015, Pages 260-267.
- Baker EL, Dickieson J, Wulfbeck W, O'Neil HF (editors): *Assessment of Problem Solving Using Simulations*. Mahwah, NJ, Erlbaum, 2008.
- Chiu, Jennifer L.; Malcolm, Peter T.; Hecht, Deborah; DeJaegher, Crystal J.; Pan, Edward A.; Bradley Michael; Burghardt, M. David. *WISEngineering: Supporting precollege engineering design and mathematical understanding: Computers & Education*, Volume 67, September 2013, Pages 142-155.
- Gibson, D. & de Freitas. *Exploratory Analysis in Learning Analytics. Technology, Knowledge and Learning*, April 2016, Volume 21, Issue 1, pp 5–19.
- Honey, M. A., & Hilton, M. (Eds.). (2010). *Learning science through computer games and simulations*. Washington, DC: National Academies Press.
- Hosein, Anesa; Aczel, James; Clow, Doug and Richardson, John T. E. (2008). *Mathematical thinking of undergraduate students when using three types of software*. In: *The 11th International Congress on Mathematics Education*, 06-13 Jul 2008, Monterrey, Mexico.
- Mislevy, R. (2011). *Evidence-centered design for simulation-based assessment*. Los Angeles, CA: The National Center for Research on Evaluation, Standards, and Student Testing. Google Scholar
- Posov Ilya, Pozdniakov Sergei. *Implementation of Virtual Laboratories for a Sci-entific Distance Game-Competition for Schoolchildren / The 2013 International Conference on Advanced ICT (Information and Communication Technology) for Education (ICAICTE2013)*, September 20-22, 2013, Hainan, China.
- Potkonjaka, Veljko; Gardner, Michael; Callaghan, Victor; Mattilac, Pasi; Guetld, Christian ; Petrović, Vladimir M.; Jovanović. Kosta. *Virtual laboratories for education in science, technology, and engineering: Computers & Education*, Volume 95, April 2016, Pages 309-327
- Pozdniakov S., Posov I, Akimushkin V., Maytattanakhon A. *The bridge from science to school / 10th IFIP World Conference on Computers in Education // WCCE 2013 Torun*, 25 July 2013.
- Pozdniakov, S, Posov, I, Pukhov, A, Tsvetkova I. *Science Popularization by Organizing Training Activities Within the Electronic Game Laboratories/International Journal of Digital Literacy and Digital Competence (IJDLC)*, Volume 3: 2 Issues (2012), p. 17-31.
- Yagunova E., Pozdniakov S, Ryzhova N., Razumovskaia E., Korovkin N. *Evaluation of Difficulty and Complexity of Tasks: Case Study of International On-line Competition "Beaver"*, *International Journal of Engineering Education* Vol. 32, No. 3(A), pp. 1141–1150, 2016

Short Tasks – Big Ideas: Constructive Approach for Learning and Teaching of Informatics Concepts in Primary Education

Valentina Dagienė, valentina.dagiene@mii.vu.lt

Vilnius University Institute of Data Science and Digital Technologies, Lithuania

Gabrielė Stupurienė, gabriele.stupuriene@mii.vu.lt

Vilnius University Institute of Data Science and Digital Technologies, Lithuania

Abstract

Constructionism as a learning paradigm is based on a design, actions and constructing things or solutions by using means via collaboration and the construction of knowledge (Papert & Harel, 1991). The main idea is that pupils can learn by performing activities by making things, or even deeper, by speaking about what they are performing (doing).

Our approach focus on informatics concept-based tasks (problems) as scaffolding activities to introduce informatics concepts to primary schools. The tasks are short and can help pupils to construct their knowledge and mental structures of informatics as a science discipline. In our long experience (more than ten years) hundreds of short tasks on informatics concepts were created.

Resent years many countries, and Lithuania among them, have been developing informatics (computer science, computing), or digital technologies, curriculum for primary school level. Usually the curriculum is based on integration with other subjects, nevertheless the principles and concepts of informatics are needed to be introduced. The key components and competence areas of informatics for primary school education needs to be identified and described (Fig. 1).

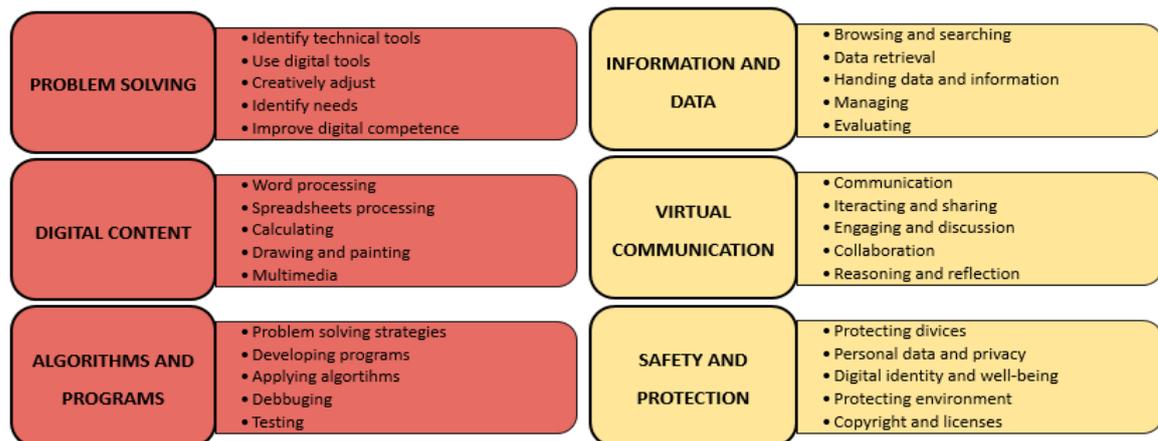


Figure 1. Lithuania identifies the key components of informatics curriculum in 6 *competence* areas

Main educational goal is to provide the different teaching possibilities and environments supporting various activities and to motivate pupils to discover new ideas. Constructionist teaching and learning is one of the important approaches and takes place through conceptually open learning activities in individual as well as group exploration by seeking common knowledge and understanding.

Keywords

Constructionism; informatics education; informatics concepts; short task; concept-based solving; primary education.

Introduction

Informatics (or computer science, or computing) education is emerging area starting from the first level in primary schools. Informatics activities can be included in other subject but not only at the level of using digital technologies. Our ambition is to go beyond technologies and present informatics as a scientific discipline for children. We show that by introducing a methodology for solving informatics concepts-based tasks.

There are many reasons for including informatics education at primary level. One of them is reducing gender inequality in the information technology sphere. Upper school students already have a vision on what is “for girls” and what things are “for boys”. Informatics usually falls into “for boys only” category. This problem might be partly avoided by introducing the course earlier (Fisher, Margolis, 2003).

The main goal of informatics at school is to teach how to think – to solve problems by using different ways including a computer. These different ways can be expressed by computational thinking – a term which become high-profile nowadays. Computational thinking involves many components starting from formulation of task, data collection and analysis, data representation; logical reasoning, abstraction, algorithm design, decomposition, parallelization, automation, pattern generalization, pattern recognition, simulation (Wing, 2011; Selby & Woollard, 2013; Weintrop et al., 2016; Google for Education, 2018)

Thinking computationally draws on the concepts that are fundamental to computer science, and involves systematically and efficiently processing information held in the tasks. Computational thinking involves defining, understanding, and solving problems, reasoning at multiple levels of abstraction, understanding and applying automation, and analysing the appropriateness of the abstractions made (Lee et. al., 2011).

When considering the transfer of knowledge, we need to understand what type of knowledge is being transferred. We need to discover conceptual and procedural knowledge. Conceptual knowledge consists of a connection of networks (chains) and is rich in relationships. Procedural knowledge is close to algorithmically thinking: consists of a series of steps on actions. Conceptual knowledge is related with deep quality of knowledge, while procedural knowledge is related with superficial quality of knowledge (Star & Stylianides, 2013). Our approach is based on obtaining the conceptual knowledge with aim to develop the procedural knowledge.

Two years ago the Ministry of Education and Science of the Republic of Lithuania organised a team of researchers, teachers, education experts and businessmen for developing a framework of informatics as well as digital technologies curricula covering both primary and secondary education. In 2017 a pilot project for introducing informatics to primary education started in ten schools. The aim of the project is to create educational content for primary schools based on the developed curriculum framework and innovative ideas. The framework of informatics curriculum is built on six core competence areas:

1. **Problem solving:** using computers, digital devices, and computer networks – principles of functioning of computers, digital devices, and computer networks; performing calculations and executing programs; using digital tools and technologies to create knowledge and to innovate processes and products; engaging individually and collectively in cognitive processing to understand and resolve conceptual problems and problem situations in digital environments.
2. **Digital content:** creating and editing digital content in different formats, to express oneself through digital means; modifying, refining, improving and integrating information and content into an existing body of knowledge to create new, original and relevant content and knowledge.
3. **Algorithms and programming:** problem solving by using computers and other digital devices – designing and programming algorithms; organizing, searching and sharing information; utilizing computer applications;
4. **Information and data:** understanding and analysis of problems – logical and abstract thinking; algorithmic thinking, algorithms and representation of information;

5. **Virtual communication:** developing social competences especially in virtual environments; project based learning; taking various roles in group projects.
6. **Safety and protection:** observing law and security principles and regulations – respecting privacy of personal information, intellectual property, data security, netiquette, and social norms; positive and negative impact of technology on culture, social life and security.

Related Works

Since 2010, Austria has a project “Informatik erLeben” (experiencing informatics) that aims at attracting students for Informatics as a constructive, technical discipline (Bischof, Sabitzer, 2011). Pupils from primary school up to upper secondary school obtained lectures by university teachers spread over a period of one and a half year. The lessons developed show pupils of all grades selected core-concepts of Informatics/CS in a playful way and at an age-specific level. The prepared lessons cover the topics adequate for primary school pupils. Topics are divided to core-concepts and into several modules that can be composed individually. For example, *Coding* (Morse Game; Creating a Code with Colours; Code trees; Error Detection); *Computer Networks* (Chinese Whispers; Communication Rules; Postman-Game); *Algorithms* (Instructions how to get somewhere); *Sorting* (Binary Search-tree); *Searching* (Blind Search; Searching in a linear Structure) etc.

Depending on the topic they act either as part of the computer, serving as data or as object being manipulated by algorithms, or assuming some role of a program. Out of principle, computers were specifically not used during the lessons. The pupils learned, based on activities, simulations, and animations. Important didactical principals behind the concept are discovery learning and teamwork.

Based on the project reflection there are some useful findings:

- It is very important to start at an early age to broaden the pupils’ image of CS and to create interest.
- While some boys already have been interested in informatics before, all participating girls could be influenced.
- Pupils must have the possibility to attend exciting CS lessons during all grades.
- Because primary school kids are very open and enthusiastic towards new topics and concepts, it is necessary to bring more technical topics in all primary schools.

Duncan and Bell (2015), having established the six general areas covered by existing primary school curricula, have analysed three key English-language computing curricula: the CSTA K-12 Computer Science standards (2011), the English computing curriculum (2014), and the Australian Digital Technologies curriculum (2013). They found some notable features:

- All three curricula introduce programming concepts from the first year (5 or 6 years old), using only sequencing and turtle graphics, which are based on concrete physical motion that students can relate to.
- Selection (branching) is introduced from about 7 years old, and if iteration (repetition) is introduced, it seems to be in the form of simple counted loops. More sophisticated iteration with conditions on the loops, and the introduction of textual (general purpose) languages, seems to be expected around 11 or 12 years old.
- Topics relating to safety and ethics are covered from the very first year, again gradually increasing in sophistication from simple scenarios for young students to more serious issues of identity and privacy as students approach their adolescent years.
- There is some difference in what is taught around “algorithms”, which covers both the design of simple programs, as well as understanding algorithms for standard problems such as searching and sorting. These standard problems serve as examples of clearly defined problems, but also allow students to investigate their performance. The Australian curriculum starts earlier with standard problems, but by 11 years old all three curricula include such algorithms. This will be

another important area to evaluate in studies with students to determine if there is value in starting early with these concepts.

Six core learning areas have been announced in New Zealand curriculum: (1) algorithms, (2) programming, (3) data representation, (4) digital devices and infrastructure, (5) digital applications, and (6) humans and computers. The proposal to relate these areas to the principles of Computational Thinking are made (Duncan, Bell, Atlas, 2017).

Webb et al. (2018) have discussed the evidence that young students, of 7–8 years old can start to develop understanding of important informatics concepts. Students can learn through hands-on experience and gradually begin to link theoretical concepts to their developing practical problem-solving capabilities. Therefore, identifying trajectories in the development of these concepts and devising effective pedagogical approaches which make use of the tools available are important current research challenges. Furthermore, in addition to developing informatics concepts to support the subject per se, it is necessary to define the underlying knowledge base of informatics concepts and crucial skills needed to support digital citizenship.

There are some suggestions about introducing informatics in primary education in Poland: informatics activities need to be included in the same place where kids playing, so no need for a full equipped classroom. Integration of informatics with other subjects during the whole week (1 hour lasts a week). Of course, sometimes a teacher may take pupils to a computer lab. Teacher has access to pupils' results regardless of the place they work, in school or at home (home works). Flipped learning method is suggested to use (Sysło, 2017).

Many countries are integrating digital competencies in primary education and introducing basics of informatics by using various activities.

Learning and Teaching Informatics in Primary Schools through Solving the Short Tasks based on Informatics Concepts

The predominant theories of learning are based on the premise that learning is an epistemological problem involving individual psychological processes that lead to the acquisition of knowledge (Lave, 2008). Thus, an individual constructivist view sees learners as active agents who construct knowledge as their own internal model of 'the world' based on the result of interactions within it. Knowledge changes from being something you acquire to be an ability to act within a community of practice (Fig. 2).

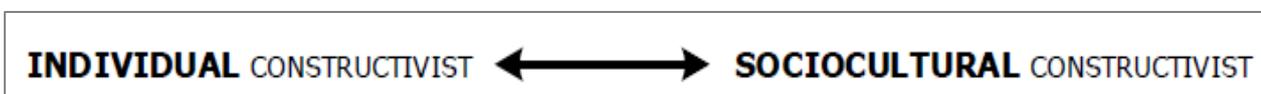


Figure 2. From acquiring knowledge individually to building community of practice

Schools should help pupils to make this transition by scaffolding their constructive work performed individually and in groups. For example, teachers can set pupils work, which were to find solutions of the given short tasks (presented as the set of several task cards). At first pupils should work individually and solve the given tasks. Then they join in groups of 3-4 pupils and discuss the tasks. Each group should discuss and agree on the presented solution. This is one of scenarios developed for teachers. We know that scaffolding increases pupils' ability of deeper understanding knowledge and promotes the acquisition of computational thinking' (Lee et. al., 2011). Thus, pupils' skills and capacities are increasing.



Figure 3. Pupils solving individually (sometimes with help of a teacher) and by groups

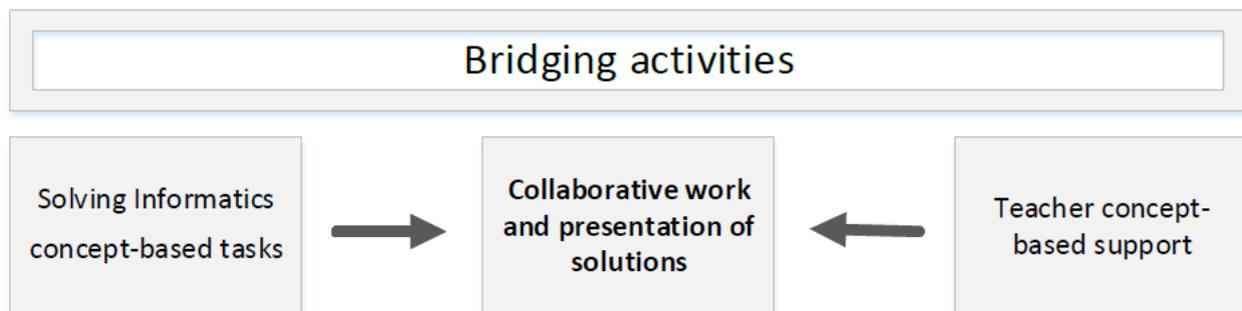


Figure 4. Solving process of tasks based on informatics concepts

Children build their own knowledge structures as they are engaged in constructing things and then these structures are becoming a materialization of ideas and thoughts. Similarly, during the solving process pupils read, think, explore, and analyse the task (text structure) and develop the solution path.

Our practice of pupils' observation during the solving informatics concepts-based tasks showed that this process can support pupils to generate and share meanings about the informatics concepts involved in the tasks and promote the development of computational thinking and understanding of informatics.

Primary school teachers usually have too little informatics background. For them, a deconstructionist process of tasks is very important (Boychev, 2015; Dagiene, Futschek, Stupuriene, 2016). Primary school teachers can improve their informatics competence through analysing, solving, and explaining the essence of the informatics concept-based tasks (Fig. 5).

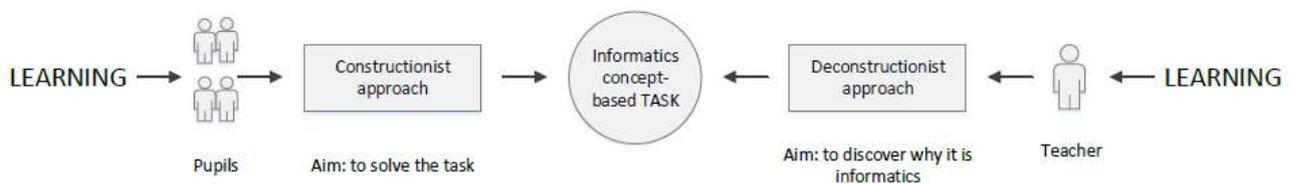


Figure 5. Pupils' and teacher's attitude towards learning by solving tasks

Informatics concepts need to be brought to pupils in attractive way, what is done by establishing Bebras challenge model (www.bebras.com) (Dagiene & Stupuriene, 2016). The Bebras challenge is aimed to promote pupils interest in informatics from the very beginning of school and lead them to develop computational thinking abilities. Main idea is to involve pupils into informatics task solving activities and to use computational thinking and modern technologies more intensively and creatively. At the beginning the Bebras challenge was aimed mainly at junior level, however nowadays more and more countries focus on primary school level as well (Carteli et al., 2010; Dagiene et al., 2017; Izu et al., 2017).

However, we need well developed short tasks which are based on informatics concepts and aimed to develop computational thinking. Since knowledge and understanding are both essential for educational

progress, the informatics curriculum should contain main concepts, and provide a clear structure, a logical progression and open ways for various didactical approaches.

The tasks should be really short, answerable in a few minutes through a computerized interface, and requiring deep-thinking skills in the informatics field. The tasks should be answered without prior knowledge in informatics, and they are clearly related to fundamental informatics concepts. To solve those tasks, pupils are required to think in and about information, discrete structures, computation, data processing, data visualization, and they should use algorithmic as well as programming concepts. Each task can both demonstrate an aspect of informatics and test the participant's ability of understanding informatics fundamentals.

International Bebras community has agreed to develop the informatics tasks according to the following framework (Fig. 6-a). The framework consists of two parts: a task formulation (text, image, question) and metadata (solution, explanation why it belongs to informatics, etc.). Metadata is very important for teachers, also for others tasks developers.

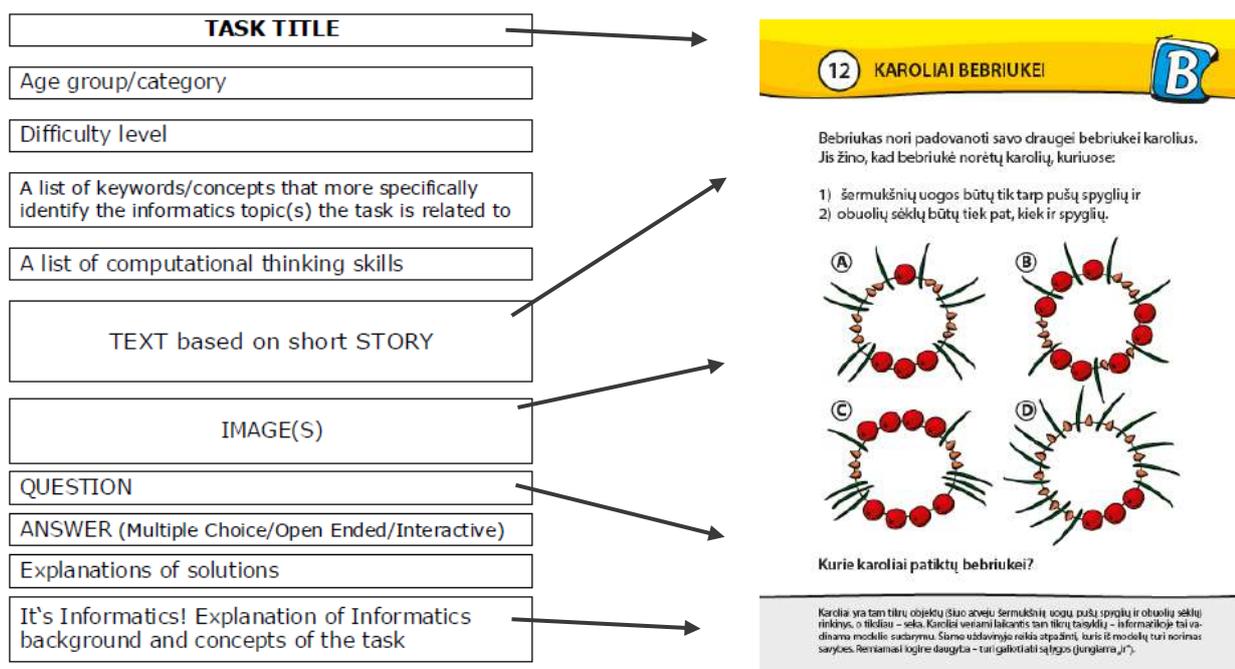


Figure 6. Structure of informatics concept-based task: a) for a Bebras task, b) for a card

Informatics concept-based tasks on cards, or the informatics tasks cards have been originated from the Bebras tasks and supported informatics without computer paradigm (CS Unplugged idea). The tasks on cards are more compact (Fig. 6-b), their metadata is published separately (in additional cards or in the internet).

The informatics concepts-based tasks are based on problem solving process. Problem solving is well known paradigm in education. Our didactical approach to introduce informatics concepts is based on short tasks which involve informatics concepts carefully selected for primary level. Bebras community has a long experience in creating informatics concept-based tasks (more than ten years' practice, more than 50 countries are involved). Solving short informatics tasks can be a transformative teaching methodology and can support an intensive learning atmosphere. Incorporating reasoning and discussions, as well as other active learning strategies is an important didactical component for educators. Many informatics concept-based tasks are developed during the annual Bebras tasks development workshops.

Short informatics concept-based tasks solving is a powerful method that can support a pedagogical shift in the classroom and foster pupils' engagement and motivation to learn. Many publications deal with problem solving methods. Problem solving of the short tasks can be considered as a systematic process

involving pupils into deeper understanding of informatics concepts. The short task solving can be one of the strategies that engage and motivates pupils for deeper learning and fosters the deeper thinking skills.

There are many ways for selecting problems to be solved by pupils in the classroom. For primary education two type of problem solving are usually declared: practical problems which take more time and cover several topics (1), and everyday exercises. Exercises are very common in mathematics and language (grammar) lessons (2). But we suggest the third type of problems - short tasks with double folded aim: to cover informatics concepts and to be solvable in few minutes.

A Pilot Study in Primary Schools: Solving the Small Tasks containing Big Ideas of Informatics

In 2017, the project introducing informatics in primary schools (“Informatika pradiniam ugdymui”⁴²) was initiated by the Ministry of Education and Science of the Republic of Lithuania together with the Education Development Centre. The aim of the project is to create informatics educational content in primary education and test it in ten schools before summer 2018. Ten schools were selected from different locations in Lithuania (<https://informatika.ugdome.lt/en/teams/>). The school teachers are asked to integrate informatics education in their lessons (grades 3 and 4): develop and testing teaching and learning material, sharing their best practices, giving suggestions on informatics educational content and curriculum in primary education.

Various activities and resources have been developed during a year working with these schools. Schools exchange their lessons plans and scenarios using different educational means and technologies, for example, Scratch Juniors, Bee Bots, CS Unplugged activities, Scottie Go, etc. Having a huge experience in informatics gamification and contests (namely, the Bebras challenge) we suggested a set of short tasks based on attractive stories and containing informatics concepts. For supporting pupils’ engagement in learning informatics concepts, we developed a set of tasks and design them as attractive story based task solving card set. Each school get 10-15 set of cards and didactical recommendations.

In January 2018, we visited 5 schools and made observation of 3 and 4 grades pupils’ work on solving short informatics concept-based tasks presented on cards. Some results are described below.

Participants

The study was divided into several 45 min sessions and took place in 5 public primary schools as a school activity – integrated into various school lessons. The participants were 87 pupils, mixed boys and girls, from the third and fourth grades, aged 10-11 years old. They worked collaboratively in small groups of 2-3 pupils in the school’s by using task cards. When discussing of solving process some schools used a projector to show the task. All of the pupils had little previous experience with informatics concepts.

The Cards – Short Tasks

A set of cards with 54 informatics concepts-based tasks was developed (Fig. 7-a).

Also an additional card contains a list of the informatics concepts with link to the tasks (card numbers) and rules example how to use the tasks. During a pilot study we mostly pay attention to solving tasks which based on the following informatics concepts: sorting and grouping, information analyses and search, understanding of an algorithm, program and automation, pattern recognition, selection, repetition, logic, and coordinates.

⁴² <https://informatika.ugdome.lt/en/about-project/>



Figure 7. A tasks card box (a) and some cards with tasks (b)

Data collection and analysis

During the study, we collected qualitative data: photos, pupil’s reactions, answering questions and reasoning of the solving process. We observed both pupils solving process, and presentation of solution. Also we focus on informatics concepts in tasks: are they recognisable by pupils as well as teachers.

Findings

At the end of the study, we got very broad view of the short tasks and their suitability to pupils for learning informatics. The tasks, which pupils have solved during our observations, are briefly presented in Table 1 with a short description and the comments they had.

Table 1: Some findings when observing pupils work by solving informatics concept-based tasks on cards

Informatics concept	Description / Activities	Discovery
Sorting, grouping	<ul style="list-style-type: none"> - sorting buttons, toys, - searching for features - groupings the given items (mentally and physically) 	About haft of the third grade pupils can easy sort items mentally by recognizing different features. 1/3 of pupils have difficulties in mental sorting and need to manipulate with physical items.
Information analyses and search	<ul style="list-style-type: none"> - observing images, texts, - extracting details, - dealing with text and numbers, searching for them 	Pupils are familiar with such type of tasks in language or mathematics lessons, they are comfortable and solve them without questioning. About 1/5 of pupils lack of concentration.
Algorithm	<ul style="list-style-type: none"> - following precisely rules and descriptions, - discovering steps and implementing them. 	Algorithm tasks are very different, from simple ones to quite complex. Then it is not easy to present summary. More than half of pupils understand rules and can follow or implement them. About 1/5 of pupils struggle with attention to follow strictly an algorithm description (rules).
Program, automation	<ul style="list-style-type: none"> - implementation of algorithms using commands, - seeking to determine repetitive parts or components 	Many pupils are familiar with table games and are able to follow rules. They can easy perform set of commands, sequences. However, automation needs more work and more tasks to be solved.
Pattern recognition	<ul style="list-style-type: none"> - dealing with everyday items (decoration, neckless) and searching for structures, - determining and finding patterns, - matching patterns. 	Task solution process is similar to sorting. About haft of third grade pupils can easy recognise simple patterns. 1/3 of pupils have difficulties in reasoning and need to manipulate with physical items. Complex patterns take more time and few pupils in classroom can properly deal with such tasks.
Selection	<ul style="list-style-type: none"> - choosing one item from two under certain condition, - dealing with condition' 	Half of pupils can deal with making decision for selecting items under conditions. Almost all pupils can follow the command's IF-THEN is implementation. Some pupils have

	- understanding statement IF-THEN-ELSE	problems with understanding ELSE. Physical games and activities would help pupils to understand the selection deeper.
Repetition	- recognizing repetitive parts, - following repetitive rules and commands, - applying repetition to everyday life processes.	Pupils like to follow repetitive (iterative) rules. Some pupils due to lack concentration can easily lose track in repetition especially by doing mentally. Pupils enjoy playing repetitive games, mentally and physically.
Logic	- reasoning by logic rules, - excluding alternative, - applying logical operations: AND, OR.	Many pupils solve logical puzzles from magazines, they are familiar with several types of logical games. Mathematical oriented pupils (about 1/4) like logic tasks and solve them very fast. Other pupils need time and support.
Coordinates	- understanding representation, - two dimensional way of thinking.	Pupils know usually tables however they do not pay attention to indicate a cell location. Coordinates introduce to two-dimensional way of thinking and it takes time for understanding. Many tasks should be solved.

In everyday life pupils apply algorithmic concepts to describe different activities and especially game rules. The most common are the sequences of do-this commands, conditional statements (if ... then ... else), logical expressions, and variables in the form of task attributes. Everybody agrees that these concepts are very important for many informatics areas, especially in artificial intelligence, robotics as well as blockchains.

For this paper, following findings from the study are relevant:

- Informatics integration model is used in primary school's education: usually schools dedicate from 15 to 45 min. per week in each grades 3 and 4. The short informatics concepts-based tasks on cards were used as an introductory resource to various topics, mainly integrated with mathematics: data analyses, pattern recognition, repetition, commands, etc.
- We were pleased to learn that all teachers use of the short tasks cards in classes as means for teaching and learning informatics basics.
- The study proved what we had worried about – there are several misconceptions about the informatics concepts. Primary school teachers do not have informatics background and need carefully prepared trainings on informatics topics.
- In general, the reactions of pupils to the short tasks are highly positive. Both teachers and pupils are enjoying by solving the tasks and discussing various topics connected to informatics concepts.
- The short tasks cards inspired the teachers themselves to learn about informatics concepts, to create similar tasks or exercises, and to extend some short tasks by physical tools.

Conclusions

Informatics education is important area of XXI century. Many research studies are aimed at computational thinking as one of the most discussed part of informatics education. In this paper, we explored the informatics concepts through the process of solving short tasks. We presented a small pilot study in which pupils were engaged in solving informatics concept-based tasks on cards. The task solving activities were developed around a set of cards. The results of the data analysis revealed some significant outcomes regarding task solving. First of all, it seems that short and story-based tasks engage pupils in solving and thinking process. Not only pupils but and teachers are interested in this process, because a lot of primary school teachers never study informatics as a science discipline.

Moreover, the analysis showed that the short task solving approach can contribute to guidance for the solving process. This study showed that the solving short tasks has significant potentials as an

educational approach for fostering computational thinking skills. However, more research needs to be done on this field.

Acknowledgments

The authors gratefully acknowledge the support of the Nordic Research Council through the NordPLUS programme of transverse actions, in particular through the funding of the two year project “Culturally Diverse Approaches to Learning Mathematics and Computational Thinking” with project code NPHZ-2018/10063. We also acknowledge the participated Lithuanian schools. The authors wish to thank Nicklas Anttu for proofreading the paper and making suggestions.

References

- Boychev, P. (2015). Constructionism and Deconstructionism. *Constructivist Foundations*, 10(3).
- Bischof, E., & Sabitzer, B. (2011, October). Computer science in primary schools—not possible, but necessary?!. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 94-105). Springer, Berlin, Heidelberg.
- Cartelli, A., Dagiene, V., & Futschek, G. (2010). Bebras contest and digital competence assessment: Analysis of frameworks. *International Journal of Digital Literacy and Digital Competence (IJDLDC)*, 1(1), 24-39.
- Dagiene, V., Futschek, G., & Stupuriene, G. (2016). Teachers’ Constructionist and Deconstructionist Learning by Creating Bebras Tasks. In *Conference Constructionism* (Vol. 16, pp. 257-264).
- Dagiene, V., & Stupuriene, G. (2016). Bebras-a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 15(1), 25.
- Dagiene, V., Sentance, S., Stupurienė, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, Vol. 28, No 1, p. 23-44.
- Duncan, C., Bell, T., & Atlas, J. (2017, January). What Do the Teachers Think?: Introducing Computational Thinking in the Primary School Curriculum. In *Proceedings of the Nineteenth Australasian Computing Education Conference* (pp. 65-74). ACM.
- Duncan, C., & Bell, T. (2015, November). A pilot computer science and programming course for primary school students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39-48). ACM.
- Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. MIT press.
- Google for Education (2018). Exploring computational thinking. Retrieved from <https://edu.google.com/resources/programs/exploring-computational-thinking/#!home>
- Izu, C., Mirolo, C., Settle, A., Mannila, L., & Stupuriene, G. (2017). Exploring Bebras Tasks Content and Performance: A Multinational Study. *Informatics in Education*, 16(1), 39-59.
- Lave, J. (2008). Everyday life and learning. *Knowledge and practice: Representations and identities*, 3-14.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, 2(1), 32-37.
- Papert, S. & Harel, I. (1991). *Constructionism*. New York: Ablex Publishing Corporation.
- Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition, available via internet: <http://eprints.soton.ac.uk/356481>
- Star, J. R., & Stylianides, G. J. (2013). Procedural and conceptual knowledge: exploring the gap between knowledge type and knowledge quality. *Canadian Journal of Science, Mathematics and Technology Education*, 13(2), 169-181.

Sysło, Maciej M. (2017). Implementing Computer Science Curriculum in schools in Poland: issues, challenges, and practice, WCCE, Ireland.

Webb, M. E., Bell, T., Davis, N., Katz, Y. J., Fluck, A., Sysło, M. M., ... & Brinda, T. (2018). Tensions in specifying computing curricula for K-12: Towards a principled approach for objectives. *IT-Information Technology*, 60(2), 59-68.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.

Wing, J. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*, Spring. Carnegie Mellon University, Pittsburgh.

Design Science Research for Computational Thinking in Constructionist Education: A Pragmatistic Perspective

Vladimiras Dolgopolas, *vladimiras.dolgopolovas@mii.vu.lt*

Vilnius University, Institute of Data Science and Digital Technologies, Lithuania

Valentina Dagienė, *valentina.dagiene@mii.vu.lt*

Vilnius University, Institute of Data Science and Digital Technologies, Lithuania

Eglė Jasutė, *egle.jasute@mii.vu.lt*

Vilnius University, Institute of Data Science and Digital Technologies, Lithuania

Tatjana Jevsikova, *tatjana.jevsikova@mii.vu.lt*

Vilnius University, Institute of Data Science and Digital Technologies, Lithuania

Abstract

Constructionist educational environments have provided solutions for effective learning process. What are the criteria of the effectiveness of using such environments? The question is even more difficult if we are seeking computer enhanced constructionist solutions. One more problem is how to train the computational thinking (CT) skills and especially to develop their assessment. If we consider the level of understanding of the relevant scientific knowledge to be transferred to the learner as the effectiveness criteria, like done in such traditionally positivistic fields like physics or mathematics, how it corresponds to the computational thinking skills and abilities? This is an even more difficult question to answer in the case of multi-paradigmatic fields like engineering or informatics. The nature of knowledge in these disciplines is non-positivistic in general, such research areas usually include not only technical but also social domains and the research methods in these fields are mainly qualitative.

The article makes an attempt to start a discussion on the above presented problems. This research aims at a detailed analysis and study of the possibility to arrange a constructionist platform and the relevant software enhanced educational environment in an effective way as it could be seen from the point of view of acquisition of CT skills. The possible solution is based on Design Science Research (DSR) methodology. The DSR methodology is developed under the pragmatistic research paradigm and could provide a framework for unification of educational solutions, especially in such educational fields like informatics and Science, Technology, Engineering, and Mathematics (STEM).

Research questions as related to software enhanced constructionist educational platforms are formulated as follows: (RQ1) What are the relations between inquiry-based learning (IBL) and CT? How can IBL help to develop CT? (RQ2) What are the advantages and possible disadvantages of inductive-deductive reasoning in connection with IBL? What is the role of abductive reasoning as related to IBL and CT? (RQ3) How to effectively (as related to CT) incorporate DSR into IBL? The article focuses on cognitive and epistemological aspects of the presented approach and analyses the possible connection of DSR to the inquiry-based educational process. Another important aspect is the possible approach to IBL in the paradigmatically non-positivistic areas of scientific research. Pragmatistic aspects of DSR, based on the implementation of abductive and circumscriptive cognitive features, allow implementation of an inquiry-based educational process that focuses on CT in various inter-disciplinary areas like, for example, informatics.

Keywords

Computational thinking, constructionist education using technology, design science research, inquiry-based education, pragmatism.

Introduction

Modern educational environments are usually based on technology. Appropriate software and hardware platforms are an integral part of modern teaching and learning. There are two main directions in the design and application of educational software: (1) the development of educational software, including software platform, and (2) development of software-based learning objects. Examples of educational software may include various educational software platforms, such as virtual laboratories (Scheckler, 2003) or serious games (Ritterfeld et al., 2009). Software-based learning objects promote the educational use of software in a form of programming code and can be used in engineering and computing educational environments, for example, in robotics (Štuikys, 2015), introductory stochastics (Dolgopolas et al., 2014), or programming education (Dolgopolas et al., 2015). Computers affect and at the same time provide requirements for the skills that students must acquire. One of these skills is computational thinking. These skills are one of the most important competencies in the 21st century (Kurilovas & Dagiene, 2016).

In this respect, the main questions to be studied are: What is the "proper" list of skills, and if it is already provided, is it still relevant for current business and society requirements? How to organize learning process that allows us to develop or improve the student's computing skills? How to integrate computers in the way that improves the learning process or, conversely, the use of computers will require additional efforts and skills of the student, and this will make the process of achieving educational goals even more difficult and less effective? How to organize a "personally significant" environment, taking into account aspects of social communication that naturally arise in the classroom and are promoted by modern means of telecommunications? Another type of questions is related to evaluation. How to evaluate progress in acquiring skills and assessing cognition skills, such as thinking skills? What is the relationship between mental abilities and cognitive abilities and how this to be assessed?

Another group of questions focuses on the possible relationship between formal requirements for the results of the educational process, such as a clearly formulated list of professional skills and knowledge, or rather abstract requirements for mental skills that need to be improved. How to organize the learning process, which will correspond to the student's acquisition of both practical knowledge and "proper" thinking skills? How, in this aspect, is possible to teach different scientific and technical disciplines that probably relate to even different research paradigms? We will try to discuss at least some of these issues and tasks with an emphasis on philosophical, epistemic and cognitive aspects related to constructionist education in general and constructivist aspects of engineering education in particular.

The history of the concept of "Computational Thinking Skills" originates from the definition given and further developed by Jeannette M Wing: "Computational thinking is taking an approach to solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing" (Wing, 2008). Computational thinking skills incorporate analytical thinking, engineering thinking, and scientific thinking thus they could be positioned as a kind of universal skills for the modern student, and this is especially true for engineering and STEM education. Wing wrote: "Computational thinking is a kind of analytical thinking. It shares with mathematical thinking in the general ways in which we might approach solving a problem. It shares with engineering thinking in the general ways in which we might approach designing and evaluating a large, complex system that operates within the constraints of the real world. It shares with scientific thinking in the general ways in which we might approach understanding computability, intelligence, the mind and human behavior" (Wing, 2008). How to arrange a proper educational environment which enables computational thinking skills to be systematically developed? There is no doubt that an integrated approach is needed. We need to provide solutions for the navigation and motivation of students, including different educational environments for schoolchildren. (Dagiene & Sentance, 2016). A more difficult problem is how to evaluate the acquisition of such skills (Dolgopolas et al., 2016)? It seems that the software enhanced environments could provide proper solutions. The logic is straightforward: computers and skills of using computer improve the skills of computational thinking, because computational thinking "...draws on concepts fundamental to computing" (Wing, 2008). However, this is not that simple. There are many contradictory examples provided in literature (Coughlan, 2015; Richtel, 2011). In many educational cases, computers and software enhanced environment do not work as expected. Instead of improving the abilities and skills

of students, the skills and abilities of students are degrading, including the skills associated with computational thinking. In order to solve this problem, we do not recommend avoiding the use of computers in educational environments. Instead, we promote the proper organization of the "computer friendly" educational process. Such an educational process must be based on a properly designed constructionist environment. Such a "proper" environment should include the appropriate "cognitive interface", provided to the student. This cognitive interface will serve as an intermediate link between the extended educational environment, which includes a computer with the appropriate programming interface, on the one hand, and the cognitive processes and epistemological actions of students on the other, and all this allows to correctly direct the "flow" of the constructionist learning process.

The article examines the modern and computer-based educational environment, including an analysis of the relevant types of educational software and its application. Also detailed analyses on the requirements and a specification of the previously described cognitive interface are provided. Although the main emphasis is on cognitive and physiological aspects, we present a description of some practical tools for computational thinking enabling modern constructionist educational environment. Since we study the computer enhanced educational environment, the presented analytical material and developed solutions are aimed at education with computers. But in connection with the general principles of constructionist education focused on computational thinking, the proposed solutions can in any case be generalized, which makes it possible to create a computer-free environment and corresponding educational solutions. The universalizing paradigm here is pragmatism, viewed as a philosophical assumption. By designing and creating a pragmatic educational environment, one can find a common way of organizing computational thinking enabling constructionist educational solutions. We will discuss this pragmatistic approach in the following sections.

Inquiry-based education and computational thinking skills in constructionist settings

In this section we discuss the educational aspects of a scientific inquiry. Then we create a link to computational thinking skills, which are acquired by students during the learning process. First, we need to answer the following question: what is the scientific or engineering inquiry and why is it important for all levels of education? There is a strong opinion (Council, 2000b; Jadrach, 2011) that students should not only learn science and scientific methods, but also should have the opportunity to "... do science. This vision for science teaching stems directly from the educational imperative to develop scientifically literate students" (Jadrach, 2011). Students can become scientifically literate if they participate in practical scientific activities. Moreover, this is true for students at all levels of educational programs in schools, as well as in universities. The practical implementation of educational technology based on scientific inquiry is a complex and challenging task (Flick & Lederman, 2004; Jadrach, 2011). Learning of scientific content corresponds to the highest levels Bloom's taxonomy. "Consequently, learning to think and act like a scientist is much more difficult to do than just learning about scientific content" (Jadrach, 2011). How could scientific inquiry be defined?

First, what constitutes scientific inquiry? Several approaches could be presented: scientific inquiry (1) begins with a scientific question; (2) is a hands-on activity; (3) is a set of specific methods and practices used by scientists; (4) is a set of reasoning strategies or skills needed while driving a scientific process. The main common feature of the presented approaches to the definition of scientific inquiry is that all these definitions are process oriented, as they attempt to define scientific inquiry, describing the activities of scientists. Another solution is to define scientific inquiry using a result-oriented approach (Jadrach, 2011). What is the primary goal of scientific activity? We share the opinion (Jadrach, 2011; Nersessian, 2010; Windschitl et al., 2008) that the primary goal of science are scientific models and the aim of any scientific work is to develop, test, and modify the scientific model of the subject of study. Generally, we could name the process of development, testing, evaluating and modification of a model as simulation. The reason for this is the following. In any case, such operations with the model should take place within time, therefore it can be described as a simulative modelling process or simply a simulation. Why do we focus on this? The reason will be clear after a closer look at the nature of the simulation. Scientific activity in the design (development, testing, evaluation and modification) of simulations as artefacts is closely related to the cognitive activity of constructing mental simulations and

simulative reasoning (Nersessian, 2010). Summing up, scientific inquiry can be defined as an activity in the design of scientific simulations, and the goal of scientific work is the design of model-based scientific simulations.

Why are simulations so important? The process of designing simulations provides a part of the cognitive interface enabling the learner's relevant cognitive activities of constructing and grounding of mental models. We will discuss this in more detail. The definition of a model-based simulation is based on the following meanings (Landriscina, 2013): (1) The meaning of a system. A system is a collection of different elements whose combination yields results that are unobtainable by the elements alone. Therefore, the system is more than a sum of its parts; (2) The definition of a model. A model is a simplified representation of a real or imagined system; (3) A simulation could be defined based on (1) and (2): a simulation is an interactive representation of the system to be studied based on a model of the system. This definition has a wider meaning than the traditional view of simulations as a dynamic set of interactive representations. Basically model-based simulations are associated with cognitive activity of students and with the process of constructing appropriate mental models (Landriscina, 2013). This approach to simulations also improves learning related cognitive processes, facilitates modification, construction or replacement of appropriate cognitive structures (Mayer, 2009). These processes include enhancement of "... cognitive processes that are crucial to learning, such as: selecting key information; organizing this information into a cognitive structure; integrating this new information into previous knowledge; accessing and creating appropriate analogies and metaphors; generating inferences; reorganizing cognitive structures" (Jadrich, 2011). One should mention an alternative – a cybernetic approach based on engineering traditions. This approach has roots in the technique of representing any system in the form of a "black box" with a certain behaviour. Typically, the aim of simulation is to observe the dynamic behaviour of the model of a real system. Thus, a person, especially with engineering experience, could view modelling as something superfluous, complex and not essential, especially for educational needs. The model can be considered more important than its simulation, and one can confine himself to the learning task only with the use of modelling. At the same time, educational simulations are usually seen as an example of the use of information and communication technology (ICT) tools, so the effectiveness of such a "traditionally understandable" educational process based on simulations can be questionable. This article focuses on simulation *making* rather than simulation *using* activities, although the role of simulation as an ICT tool is not completely rejected.

The constructionist approach focuses on the creation of physical or mental "things" (artefacts) during instruction and student interaction with these artefacts in order to facilitate knowledge (Papert, 1980). Focusing on simulations making, simulating using activities can be an alternative in some cases depending on the available learning environment. Implementing the engineering point of view, modelling and simulations can be defined more generally as follows (Raczynski, 2014): modelling as a link between real systems and models and simulation as the relationship between models and computers. In fact, the simulation is based on the model. From the engineering and pragmatistic positions, the model can be defined as: "A model is a description of some system intended to predict what happens if certain actions are taken" (Bratley et al., 2011) and extended in (Zeigler, 1975). First, you need to specify a set of model components. Each component is described by a set of input, output, and state variables. Then the experimental frame is defined as the set of all descriptive variables. Depending on the chosen level of simplification, various experimental frames can be determined (Zeigler, 2014).

Another important aspect is the aspect of simulative scientific reasoning. A model-based scientific reasoning based on simulations of mental models was developed (Nersessian, 2010). As a rule, the mental model corresponds to the conceptual model, which is built on the first stage implementing model-based simulation activities. A properly designed learning process should implement a unique mapping between model-based cognitive simulations and computer simulations based on models that must be designed during the educational activity. This mapping can be provided by the teacher also in the form of co-mediated teaching. Presented ideas of intermediate conceptual models and simulation making educational activities provide a clear bridge to the environment aimed at acquiring of computational thinking skills. For example, there is clear evidence of conceptual support of acquisition of computational thinking skills practically applied in the contest-based educational environment (Dagiene

& Stupuriene, 2016). As will be discussed in the following sections, an approach that focuses on developing learner's conceptual models as intermediaries for his or her cognitive grounding will provide a solution to the computational thinking skills to be acquired. In this aspect, the set of computational thinking skills can serve as a kind of criterion for evaluating the effectiveness of the educational process, with an emphasis on epistemological and cognitive educational aspects. Such an assessment is aimed to root the process of justification that takes place in the process of the student's abductive reasoning. We will discuss this in more detail in the following sections.

Inductive-deductive reasoning scheme and educational aspects

Computational pedagogy is an example of one possible approach to arrange the educational environment in the constructionist manner and is especially relevant for the introductory level education. The approach is based on didactic schemes, general computational thinking skills and practical abilities to conduct simulations. Traditionally, a deductive approach is considered as the most applicable for teaching scientific topics. Such an approach is considered as demotivating for students, especially for topics where sufficient theoretical background and a large amount of pre-knowledge are required (Yasar & Maliekal, 2014; Prince & Felder, 2007). On the contrary, the inductive approach is promoted as such an alternative, which can improve the positive attitude of students towards the learning of science (Council, 2000a). Inquiry-based learning, problem-based learning, and project-based learning, all relate to forms of inductive instruction (Prince & Felder, 2006). To overcome the described problems of the deductive approach, modelling and simulation-based computational pedagogy is proposed. Such pedagogy allows "cycle back and forth between the inductive and deductive approaches to learning" (Yasar & Maliekal, 2014) with the assistance of modelling and simulation tools.

The theoretical foundations of computational pedagogy are based on the concept of cognitive retrieval, presented by P. C. Brown, H. L. Roediger III, and M. A. McDaniel (Brown et al., 2014). This so-called interleaved retrieval practice creates a cognitive basis for the interdisciplinary computational pedagogical content knowledge. "Interleaving retrieval practices by weaving together multi-disciplinary features around a common topic (i.e., interdisciplinary education) have great advantages for gaining deep and lasting knowledge" (Yaşar, 2013). The process of model-based reasoning is presented as an inductive/deductive cycle of modelling and retrieval of models. This process "is consistent with the dual deductive and inductive process of computational modelling and simulation" (Yaşar, 2016). Using modelling and simulation, the learner can receive feedback from the modelling and simulation environment, thereby promoting his or her construction of knowledge.

The presented approach consists of two main principles: (1) the principle of designing educational environment based on using modelling and simulations and (2) the idea of scientific reductionism. There are several important aspects for discussion. First, and this has already been mentioned in the previous section, from the educational perspectives, there is a great difference between simulation *making* and simulation *using* activities. Moreover, there are many doubts about the effectiveness of simulation-using tools aimed at increasing the learner's motivation and his conceptual understanding of scientific topics (Council, 2011). Usually, the practical implementation of simulation using predefined software can be considered as a sample of a pre-designed cognitive artefact.

Application of the artefact in the educational environment can be studied from different points of view: the system view and the personal view. The effectiveness of educational software is a key motivating factor for educators in favour of including such solutions in the educational process. However, from the learner's point of view, such a cognitive artefact is positioned as another tool with a personal view of its effectiveness (Fig. 1 and Fig. 2 are adapted from Norman, 1991, p. 3). D. C. Norman provides the following illustrative example: consider a to-do list (1991, p. 3). Such a checklist, for example, developed for aircraft pilots, enhances the cognitive abilities of pilots and improves their memory. Therefore, from the point of view of the system, it is a memory enhancer. From the point of view of the pilot or from a personal view, using the list is just another task requiring different type of activity. Without a check list, a person should remember all the to-do tasks. To use the list, you must perform the following task: (1) building a list (in the described case is done beforehand by the third person); (2) do not forget to read the list; (3) reading and interpreting the items of the list items.

To sum up, if from the point of view of the system (read the "teacher") the cognitive abilities are improved, from the personal (student's) view the person participates only in (2) and (3) activities, because instead of trying to memorize the elements of the list, the person now should remember only to consult this list. This type of memory degradation is clearly manifested in the daily practice of education. Perhaps this is one of the reasons for the current popular movement towards educational technology without a computer (Richtel, 2011). With regard to the use of educational software as an advanced educational tool for constructionists educational platforms, such educational platforms, viewed as cognitive artefacts from the user's point of view, can be very harmful and demotivating. Obviously, during the process of using software there is some shift in tasks and cognitive processes, therefore educational software platforms must be carefully designed and tested. Another obstacle is the problem of bricolage (Papert & Harel, 1991). By focusing on using the provided educational platforms, the learner will improve his / her skills in using the software, rather than constructing knowledge for the learning topic. The bricolage leads to the "endless debugging of the 'try-it-and-see-what-happens' variety" (Ben-Ari, 2001). As can be seen, the described aspects are interrelated. Next, the relevance of the use of the ideas of scientific reductionism for the theoretical grounding. To argue this, we first provide a description of inquiry-based educational process: "... Inquiry also refers to the activities of students in which they develop knowledge and understanding of scientific ideas, as well as an understanding of how scientists study the natural world" (Colburn, 2000). As can be seen from the above description, it would at least be questionable to reduce scientific methods to positivistic and basically quantitative, as scientific reductionism does. Other forms of research and the paradigm of research are also important. This is especially true for interdisciplinary engineering areas, such as, for example, research of information systems, where interpretivist and pragmatist paradigms and methods of qualitative research are of paramount importance (Goldkuhl, 2012; Walsham, 2006). As for cognitive and epistemological aspects, to reduce the scheme of reasoning only to inductive-deductive reasoning is doubtful. This can lead to the disabling of the proper grounding process for the student's cognitive models, and we will discuss this aspect in the next section.

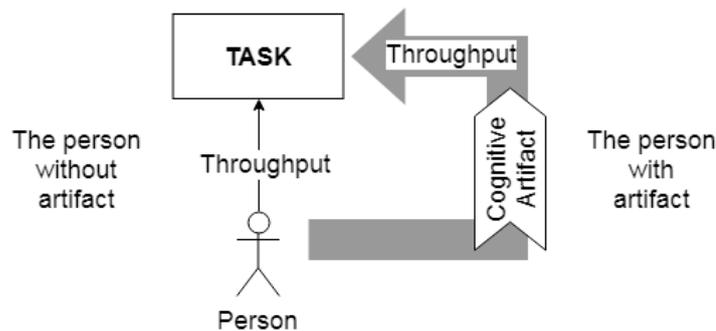


Fig. 1. System view of a cognitive artefact. Adapted from (Norman, 1991, p. 3)

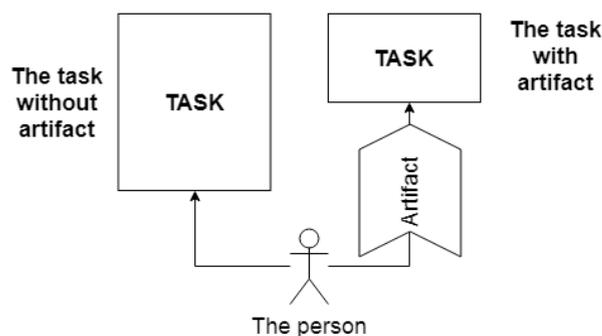


Fig. 2. Personal view of a cognitive artefact. Adapted from (Norman, 1991, p. 3)

Pragmatistic approach as the key to inquiry-based education

The psychological aspects of situated cognition give recommendations on how the learning environment should be designed and constructed. Obviously, there is a clear distinction “between natural environments which afford the learning of ‘percepts’ in everyday life, and unnatural environments” (Laurillard, 2002). Viewing from pragmatic positions, modelling and simulation is considered as a tool for grounding a student to such an artificial environment in terms of grounded cognition (Barsalou, 2008, 2010). Using computer simulation in a constructivist way could be considered as a specific type of grounding through situated simulations (Barsalou, 2008; Pezzulo et al., 2013). How can this grounding be practically achieved? Simulation-based learning can be described from the standpoint of the progression of mental models (Landriscina, 2013): “... beginning with a student’s initial model of an examined system and developing into a target conceptual model—presumably the same one underlying the simulation’s computational model. Moreover, to arrive at the target model, students must first develop their own intermediate conceptual models, which are mental models expressed as cognitive artifacts”. The presented approach focuses on the conceptual model, as an intermediate for grounding. How can such conceptual models be developed? Following is noteworthy. Conceptual models are based on mental models, so first one has to ask the next relevant question: how does the mental model develop by the human brain? The approach developed by P. Thagard considers “mental models as representations consisting of patterns of activation in populations of neurons” (Thagard, 2010). This cognitive model-based approach overcomes the limitations of sentential models of theoretical abduction (Magnani, 1999; Magnani, Casadio & Magnani, 2016) and expands C. S. Peirce ideas of how mental models can “contribute explanatory reasoning” (Peirce, 1992) going beyond verbal information and including “visual, olfactory, tactile, auditory, gustatory, and even kinesthetic representations” (Magnani, 1999). Considering mental representations as patterns of firing in neural populations, the process of constructing of mental models could be presented as a chain of patterns developing by the process of causal correlations (Magnani, 1999). Such an approach could be used to provide explanations of how abduction could generate new ideas. The neural model of abduction presented by P. Thagard and T. Stewart (2011) implements a fully multimodal convolutional model of “creative conceptual combination” describing “many kinds of creativity and innovation, including scientific discovery, technological invention, social innovation, and artistic imagination” (Thagard, 2010). The human brain is adapted to powerful learning mechanisms: “One of these learning mechanisms is an abductive inference, which leads people to respond to surprising observations with a search for hypotheses that can explain them. Like all cognitive processes, this search must be constrained by contextual factors such as triggering conditions that cut down the number of new conceptual combinations that are performed” (Thagard, 2010). It is obvious that such triggering conditions use circumscription (McCarthy, 1981) in the form of a previous experience to eliminate inapplicable transactions.

How could the abductive reasoning, viewed from pragmatistic positions, practically improve the software-based constructionist educational environment towards acquisition and improving of computational thinking skills? First, we should discuss the pragmatistic meaning of abduction (Peirce, 1992). What are the differences between abduction and the well-known hypothetical-deductive method (Lawson, 2015; Magnani, 2009)? The essence of abduction or abductive reasoning, as can be seen from the pragmatistic perspective, is to provide a way to generate a “clear” hypothesis. How to describe

the definition of "clarity"? First, according to Pierce, the clear idea is that it is possible to experience in practice. As he reminds us by the example of the concept of hardness "there is absolutely no difference between a hard thing and a soft thing so long as they are not brought to the test" (Magnani, 2009; Peirce, 1992). Regarding the research issue, the learner can test the hypothesis by implementing the simulation and modelling process. Further, C. S. Pierce's idea of "pragmatic" truth as a result of inquiry (Peirce, 1992). In this respect, abduction can be considered "as inference to the best explanation, that also evaluates hypotheses by induction." (Magnani, 2009). The criteria for choosing the "best" hypothesis among the possible ones are presented in the form of a list of requirements for computational thinking skills that must be acquired and is provided during the co-mediated teaching process. The description of computational thinking (Barr et al., 2011) could be positioned as a pragmatic criterion for evaluating the effectiveness of the process of abductive reasoning.

Enhancing educational process by circumscription and abductive reasoning

To formalize the presented approach, a model of modelling (Justi & Gilbert, 2002), which describes how students produce their models as mental and as expressed ones, will be used. The presented approach considers modelling as "non-linear creative process comprised of multiple and complex stages mainly concerning with acquiring information about the entity that is being modelled (from empirical observations and/or from previous knowledge); producing a mental model of it; expressing that model in an adequate mode of representation, testing it (through mental and empirical experimentation) and evaluating its scope and limitations" (Justi, 2009). From the point of view of cognitive reasoning, the presented "model of modelling" could be generalized as follows. First, the propositional phase is needed to generalize existing information by inductive reasoning. Further, the process of producing mental models based on existing information requires a kind of hypothetical model-based reasoning that should be involved, therefore, as previously described, a kind of abduction (or grounded abduction in this particular case) is required to be implemented in such a case. The following steps are based on such a classical method of reasoning as a deduction, requiring some form of conceptualization through an empirical design process. Finally, all these processes are based on existing knowledge and skills, which provide some sort of limitations in the form of circumscriptive reasoning.

This practical model, related to modelling and simulation in education, is generalized by Franco Landriscina (Landriscina, 2013). The processes of abduction and circumscription are extremely important, since they provide a plane dividing simulation using educational methods from simulation making educational methods. At the same time, the presented approach provides a clear picture of the general practice of eliminating abduction and circumscription from the educational process. This is a kind of common practice based on complete reliance on software tools, like a magic wand. Thus, you can completely defocus your educational goals, focusing not on teaching inquiry and scientific reasoning, but simply training you additional skills in using software tools. Obviously, the practical implementation of the described approach is not a trivial task. This requires additional efforts for pre- and post-training of teachers, as well as to develop additional educational programs. Moreover, the lack of practical examples for such activities stimulates a strong movement for education without computers, like presented by (Bell et al., 2009), that is, behind-the-scenes movement for the introduction of abduction and circumscription in pedagogical practice. Another reason for such popular movement of learning without computer can be a strong influence of instructional techniques, a sort of Instructionism vs Constructionism as it was defined by S. Papert and I. Harel (1991).

Design Science Research cognitive aspects

Previously described cognitive and epistemological aspects require additional efforts to improve the software driven educational environment, which allows to develop inquiry-based constructionist approaches to computational thinking skills. Correct application of educational software tools allows us to organize the learning context in a constructionist manner, that is, to develop a context that is personally significant for the learner. In this regard, the following important remark should be made. We are studying a complex environment that includes both technical and social factors. Technical factors,

in addition to hardware, also include software products in the form of educational software or software learning objects. This environment can be characterized as socio-technical. When studying, developing or implementing such educational environment, it is necessary to take into account not only the technical aspects of the system, but also social aspects and interactions. Such environment: has a type of a socio-technical system; includes participants (teachers, students, educational authorities, community, other stakeholders), as well as educational technology, instructional design methods, educational tools; educational tools are (mostly) artefacts, including cognitive artefacts in the form of educational software or software-based learning objects (software as a learning object). We could continue to analyse the previously described aspects from an epistemological standpoint. In general, science can be divided into formal science, such as logic or classical mathematics and factual science, which describes, explains and predicts phenomena and is tested when it gives empirical data. Factual science is divided into natural and social sciences. Natural sciences are interested in objects or phenomena, and the main research activity is to analyse their nature and the reasons for their occurrences (Dresch et al., 2014). Social science describes and reflects the society and individuals. Research conducted in social science is usually question-based and it is focused on the researchers' view on the problem in study, therefore it is subjective in its nature (Dresch et al., 2014; Romme, 2003). Social science could focus on descriptions with attention to a quantitative approach. Another focus, for example in management science, is on solutions to given problems or on artefacts creation (Dresch et al., 2014). The concept of Design Science as science of artificial was first introduced by H. A. Simon (Simon, 1996). The Design Science is focusing on practical solutions and artefacts. The motivational cause of any research can range from research focused on solving theoretical problem and without any or less concern with practical applications, or applied research focused on practical solutions (Saunders et al., 2009). Generally, design means creation (or invention) of some new artefacts and its implementation into the area of application. This could be done under existing or non-existing (innovative design) theoretical backgrounds (Vaishnavi & Kuechler, 2004). If Design Research focuses on the question of how to design artefacts, Design Science Research (DSR) focuses on the problem of using design as a research method (Vaishnavi & Kuechler, 2004). Therefore, DSR could be positioned as a well-formalized teaching technique, which implements learning through building an educational paradigm and inquiry-based educational methods. Considering inquiry-based educational process, DSR could provide a set of formalizations for implementing in the practical process of instruction. The main focus of the application of DSR methodology in education is "to teach research" (Vaishnavi & Kuechler, 2004). To go further, the artefact is the highlight of our interest, which gives us a target for the design process. Therefore, the design can be described as relating to the artefact, its internal structure and the "crafting" process of the outer environment (Vaishnavi & Kuechler, 2004). The next question should be answered: can design be research? The answer is affirmative, if the learning process is built as an artefact-oriented, enabling research through the process of designing an artefact. A model for DSR process focuses on the contribution of new knowledge to be produced. Fig. 3 (adapted from Vaishnavi & Kuechler, 2004) presents the cognitive aspects of the DSR process.

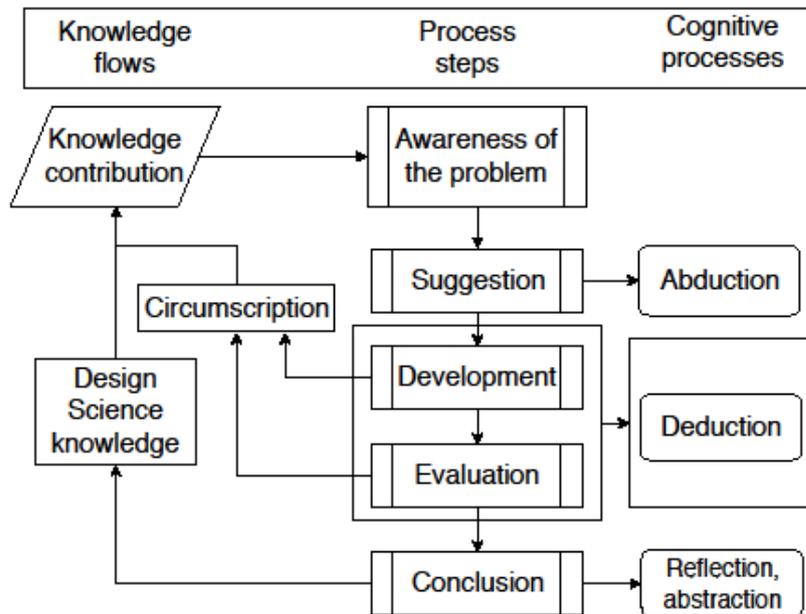


Fig. 3. The cognitive aspects of the DSR process. Adapted from (Vaishnavi & Kuechler, 2004)

Towards computational thinking by using Design Science Research approach

The DSR methodology provides a way for the formalization of an inquiry-centered approach based on the development of model-based scientific simulations. This methodology provides a set of analytical techniques based on circumscription and abduction reasoning. The formal structure of research based on DSR, very clearly corresponds to educational objectives. This correspondence can provide a set of formalisms for practical implementations of DSR as a set of educational tools. The purpose of DSR as an educational tool is twofold: (1) First, formalize the process of design of model-based simulations as cognitive artefacts. For this educational purpose, the set of DSR techniques is truncated and adapted for the educational needs. The learner is immersed in a kind of “quasi” scientific research environment, satisfying inquiry by designing a set of simulations based on provided models of one or another type. Therefore, a practical algorithm could be provided for such a case. The most important remark is the following: the learner should get a clear idea of the meaning of “rigor” (with reference to this educational scientific research environment and possible future “real” scientific research). In this educational case, the word rigor strict adherence to the steps of the algorithm and the instructions of the teacher; (2) Next, to introduce DSR as a practical design tool for the future scientific activities of the students. For this purpose, an example of one of the DSR formalisms can be introduced to the students. Such an introduction will provide a clear understanding of the method and its possible future (research) and present (educational tool) applications.

Educational process could be arranged as follows. The research question is provided by an educator in the form of a project or a problem to be studied in detail (Bell, 2010; Mills & Treagust, 2003; Solomon, 2003). The learner should start with developing some definitions and generalizations and specify how the system to be modelled should be defined (from reality to a system, identification of the problem). This identification should be provided in the form of definite and clear answers to the presented educational research questions (Dresch et al., 2014). Then, a formal review-based on the previous study should be provided and approved by an educator. The fourth step is to provide the generalization (factorization) of the previous studies in the next form: what class of similar problems could be named? The final step (implementation) is to propose a practically implementable solution for the problem: the learner starts from proposition for the simulation solution (from system to model, the representation process): how to implement a simulation solution for the previously formulated problem? After, the design process for a specific cognitive artefact for the presented problem has to be solved: what are possible models of the system? Then, the implementation phase follows (from model to simulation, the

exploration process): what is the practical solution for simulation? Evaluation phase follows the next. In this phase, the learner runs the simulation on the computer and evaluates the results. After, the clarification of the problem could be done and the steps are repeated if needed. Finally, the learner summarizes the problem and provides a kind of generalizations in the form of the final report. All these steps provide a formal basis for inquiry-based research in the form of developing of model-based scientific simulations.

The presented approach provides a practical educational methodology for the constructionist inquiry-based learning. Such well-developed approaches as a model-based approach to instruction and DSR are implemented in the form of inquiry-centered pedagogy. The model-based approach provides a solid foundation for teaching via development of model-based scientific simulations, enhancing scientific inquiry and project-based constructionist teaching methods. Design Science Research provides clear formalizations in the form of universal educational techniques. Using these considerations, students can act within an interdisciplinary group as quasi-researchers, generating hypotheses, designing simulations, and evaluating results using DSR techniques. Teachers should provide a quasi-research environment by predesigned multifaceted models, educational instructions, and seamless theoretical backgrounds.

Conclusions

The provided discussion analyses the pragmatistic aspects of the constructionist inquiry-based educational environment. The focus is on the cognitive and epistemological features of the learning process, including descriptions and studies of applications of such important meanings as circumscription and abductive reasoning. Abductive reasoning is considered both from the point of view of the formation of hypotheses, and from the point of view of evaluating hypotheses (Magnani, 2009). This provides foundations for a model of grounded cognition and allows and inquiry-based educational activities to be organized in a "personally significant" way. At the same time, the formalities of the DSR provide a framework for the teacher to design the inquiry-based educational environment that is aimed at acquiring of computational thinking skills while developing new knowledge. The cognitive aspects of DSR clearly correspond to previously described epistemological considerations, and at the same time this approach can be viewed as a kind of universal approach that could be used for various educational topics and in various educational environments, including computer-based STEM or engineering education. The list of computational thinking skills provides criteria for assessing of the effectiveness of the process of the student's abductive reasoning. The methodological paradigm for this approach is clearly not positivistic, and, as discussed in the article, the pragmatistic nature of the presented approach provides a way of practical implementation of the study results.

References

- Barr, David, Harrison, John, & Conery, Leslie. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, 38(6), 20-23.
- Barsalou, Lawrence W. (2008). Grounded cognition. *Annu. Rev. Psychol.*, 59, 617-645.
- Barsalou, Lawrence W. (2010). Grounded cognition: Past, present, and future. *Topics in cognitive science*, 2(4), 716-724.
- Bell, Stephanie. (2010). Project-based learning for the 21st century: Skills for the future. *The Clearing House*, 83(2), 39-43.
- Bell, Tim, Alexander, Jason, Freeman, Isaac, & Grimley, Mick. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13(1), 20-29.
- Ben-Ari, Mordechai. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45-73 %@ 0731-9258.
- Bratley, Paul, Fox, Bennet L, & Schrage, Linus E. (2011). *A guide to simulation*: Springer Science & Business Media.

- Brown, Peter C, Roediger III, Henry L, & McDaniel, Mark A. (2014). *Make it stick*: Harvard University Press.
- Colburn, Alan. (2000). An inquiry primer. *Science scope*, 23(6), 42-44.
- Coughlan, Sean. (2015). Computers 'do not improve' pupil results, says OECD. *BBC News*, 15.
- Council, National Research. (2000a). *How people learn: Brain, mind, experience, and school: Expanded edition*: National Academies Press.
- Council, National Research. (2000b). *Inquiry and the national science education standards: A guide for teaching and learning*: National Academies Press.
- Council, National Research. (2011). *Learning science through computer games and simulations*: National Academies Press.
- Dagienė, Valentina, & Sentance, Sue. (2016). *It's computational thinking! Bebras tasks in the curriculum*. Paper presented at the International Conference on Informatics in Schools: Situation, Evolution, and Perspectives.
- Dagiene, Valentina, & Stupuriene, Gabriele. (2016). Bebras--A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in Education*, 15(1), 25-44.
- Dolgopolas, Vladimiras, Jevsikova, Tatjana, Dagiene, Valentina, & Savulioniene, Loreta. (2016). Exploration of Computational Thinking of Software Engineering Novice Students Based on Solving Computer Science Tasks. *International Journal of Engineering Education*, 32(3), 1-10.
- Dresch, Aline, Lacerda, Daniel Pacheco, & Antunes Jr, José Antônio Valle. (2014). *Design science research: A method for science and technology advancement*: Springer.
- Flick, Lary, Lederman, Norman G. (2004). Scientific inquiry and nature of science. *Contemporary Trends and Issues in Science Education*.
- Goldkuhl, Göran. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European journal of information systems*, 21(2), 135-146.
- Yaşar, Osman. (2013). Teaching science through computation. *generations*, 13, 15.
- Yaşar, Osman. (2016). Cognitive Aspects of Computational Modeling and Simulation in Teaching and Learning. *J. Computational Science Education*, 7(1).
- Yasar, Osman, & Maliekal, Jose. (2014). Computational Pedagogy: A Modeling and Simulation Approach. *Computing in Science & Engineering*, 16(3), 78-88.
- Jadrich, James. (2011). *Learning & teaching scientific inquiry: Research and applications*: NSTA press.
- Justi, Rosária. (2009). Learning how to model in science classroom: Key teacher's role in supporting the development of students' modelling skills. *Educación química*, 20(1), 32-40.
- Justi, Rosária S, & Gilbert, John K. (2002). Modelling, teachers' views on the nature of modelling, and implications for the education of modellers. *International Journal of Science Education*, 24(4), 369-387.
- Kurilovas, Eugenijus, & Dagiene, Valentina. (2016). Computational thinking skills and adaptation quality of virtual learning environments for learning informatics. *International Journal of Engineering Education*, 32(4), 1596-1603.
- Landriscina, Franco. (2013). *Simulation and learning*: Springer.
- Laurillard, Diana. (2002). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*: Routledge.
- Lawson, Anton E. (2015). Hypothetico-deductive method *Encyclopedia of Science Education* (pp. 471-472): Springer.
- Magnani, Lorenzo. (1999). Model-based creative abduction *Model-based reasoning in scientific discovery* (pp. 219-238): Springer.
- Magnani, Lorenzo. (2009). *Abductive cognition: The epistemological and eco-cognitive dimensions of hypothetical reasoning* (Vol. 3): Springer Science & Business Media.
- Magnani, Lorenzo, Casadio, Claudia, & Magnani. (2016). *Model-based reasoning in science and technology*: Springer.

- Mayer, Richard E. (2009). *Multimedia Learning*. Cambridge University Press.
- McCarthy, John. (1981). Circumscription—a form of non-monotonic reasoning *Readings in Artificial Intelligence* (pp. 466-472): Elsevier.
- Mills, Julie E, & Treagust, David F. (2003). Engineering education—Is problem-based or project-based learning the answer. *Australasian journal of engineering education*, 3(2), 2-16.
- Nersessian, Nancy J. (2010). *Creating scientific concepts*: MIT press.
- Norman, Donald A. (1991). Cognitive artifacts. *Designing interaction: Psychology at the human-computer interface*, 1, 17-38.
- Papert, Seymour. (1980). *Mindstorms: Children, computers, and powerful ideas*: Basic Books, Inc.
- Papert, Seymour, & Harel, Idit. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Peirce, Charles Sanders. (1992). *The essential Peirce: selected philosophical writings*, 2: Indiana University Press.
- Pezzulo, Giovanni, Barsalou, Lawrence W, Cangelosi, Angelo, Fischer, Martin H, McRae, Ken, & Spivey, Michael. (2013). Computational grounded cognition: a new alliance between grounded cognition and computational modeling. *Frontiers in psychology*, 3, 612.
- Prince, Michael, & Felder, Richard. (2007). The many faces of inductive teaching and learning. *Journal of college science teaching*, 36(5), 14.
- Prince, Michael J, & Felder, Richard M. (2006). Inductive teaching and learning methods: Definitions, comparisons, and research bases. *Journal of engineering education*, 95(2), 123-138.
- Raczynski, Stanislaw. (2014). *Modeling and simulation: the computer science of illusion*: John Wiley & Sons.
- Richtel, Matt. (2011). A Silicon Valley school that doesn't compute. *The New York Times*, 22.
- Romme, A Georges L. (2003). Making a difference: Organization as design. *Organization science*, 14(5), 558-573.
- Saunders, Mark, Lewis, Philip, & Thornhill, Adrian. (2009). *Research methods for business students*: Pearson education.
- Simon, Herbert A. (1996). *The sciences of the artificial*: MIT press.
- Solomon, Gwen. (2003). Project-based learning: A primer. *Technology and learning*, 23(6), 20.
- Thagard, Paul. (2010). How brains make mental models *Model-based reasoning in science and technology* (pp. 447-461): Springer.
- Thagard, Paul, & Stewart, Terrence C. (2011). The AHA! experience: Creativity through emergent binding in neural networks. *Cognitive science*, 35(1), 1-33.
- Vaishnavi, Vijay, & Kuechler, William. (2004). Design research in information systems.
- Walsham, Geoff. (2006). Doing interpretive research. *European journal of information systems*, 15(3), 320-330.
- Windschitl, Mark, Thompson, Jessica, & Braaten, Melissa. (2008). Beyond the scientific method: Model-based inquiry as a new paradigm of preference for school science investigations. *Science education*, 92(5), 941-967.
- Wing, Jeannette M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, 366(1881), 3717-3725.
- Zeigler, Bernard P. (1975). Simulation based structural complexity of models. *International Journal of General System*, 2(1), 217-223.
- Zeigler, Bernard P. (2014). *Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems*: Academic press.

Computational Thinking in Teacher Professional Development Programs

Xiaoxue Du, xd2164@tc.columbia.edu
Columbia University, New York, NY, USA

Kay Chioma Igwe, kci2104@columbia.edu
Columbia University, New York, NY, USA

Abstract

The integration of computational thinking, CT, through technology, into curriculum strongly depend on teacher openness to compliance, of which is strongly correlated to awareness and technological skill level. Here, technological skill refers to ability to manipulate hardware, such as using problem solving to build a simple car robot, and/or software creation and implementation. In this pilot study, we create a project-based technology based workshop for teachers that introduce principles of computational thinking and analyze their abilities and thoughts about adding these learning tools into their curriculum. Six pre and in-service teachers in New York City were recruited to participate in this forty five minute workshop at separate times. The results suggest that technologically driven project-based learning design principles have the potential to increase participants' confidence and as a consequence motivate them to consider applying computational thinking related pedagogical practices in their classroom especially in subjects relating to math and science. Although CT pedagogical practice is rather technical and new to the majority of the participants, technology driven workshops may be necessary to increase teacher participation in CT classroom CT practices. This paper outlines the process by which computational thinking, through technology and problem solving, can be embedded in a professional training program for teachers. The results suggest that teachers need to have an understanding of technology and ways to implement computational thinking in order for it to be introduced in curriculum.

Keywords

computational thinking; project-based learning; teacher professional development; pedagogical practice; curriculum; workshop; program

In a global atmosphere where digital technology plays an important role, it is critical for individuals to have the education, knowledge, and skills to recognize their rights and responsibilities in this interconnected digital world (Standard 1, Empowered Learner, ISTE), and to develop strategies for understanding and solving problems (Standard 5, Computational Thinker, 2014 ISTE). Czerkawski and Lyman (2015) argues the knowledge that individuals have that corresponds to 21st century skills has to go beyond knowledge acquisition; i.e., individuals are expected to understand the basics of computer systems and practices. Students need to develop skills in critical thinking, communication, creativity and collaboration. In the 1970s, numerous computer scientists researched the field of Computational Thinking (CT) and defined it as "procedure thinking" (Papert, 1981). Several findings suggested that key CT constructs, including algorithms, abstraction and automation, can move students from merely being "technology-literate" to using "computational tools" to solve problems (Yadav, Hong, & Stephenson, 2016).

The question is how educators prepare for students to develop relevant concepts and skills in order to cope with the technological changes of the 21 century. This educational goal can be achieved by preparing teachers by increasing their technological and computational problem solving literacy. Thus, these educators will be more equipped to teach students certain CT principles such as how computers make decisions based on human being behaviors in order to solve the real-life problem.

The CS4All policy initiative makes a concerted effort to increase the level of computer science education and provide equitable access to the students in the New York City. However, building educator knowledge and capacity in CT is a complex issue that involves legislative, administrative, political and

educational challenges. Specifically, there are two major educational challenges, which are i) understanding the body of knowledge teachers need in order to be able to embed the CT framework into the regular class practices; ii) how to introduce the concepts and skills of CT to build teachers competency considering the common core standards in the Next Generation Science Standards (NGSS), Math and English language learning (ELA).

This paper begins with the discussion of challenges and changes that schools are facing in the digital age, and the rationale to introduce CT focused professional development to pre-service and in-service teachers. Next, this paper introduces a hands-on project based professional development program to increase educator CT related knowledge and skills. This paper concludes with a discussion of the study design, the challenges of incorporating CT pedagogical practices into the classroom, and the potential for a larger scale research study that looks into more ways of designing educator programs for increased probability of computational thinking classroom integration.

Literature review

Computational Thinking

The importance of the integration of computational thinking into curriculum is based on the idea that knowledge and skills derived from the application of computer science principles can be beneficial to all the learners, especially in areas where problem solving is imperative (Weintrop, Holbert, Horn & Wilensky, 2016). Computational thinking is necessary for building learner skills in areas such as algorithmic and critical thinking, of which both increase learner ability to tackle tough decisions. This skill set can be applied to all subject areas such as music, english, and are not limited to science, technology, engineering, and mathematics, STEM, type classes. The increased accessibility of the computer facilities and the development of the national education technology plan has further validated the need for the application of CT in the K-12 education.

Central to the skill of CT is “the ability to encode ideas into a form that can be interpreted and executed by a computational device ” (Weintrop et al., 2016). Recently, Wing (2006) suggested that CT involves solving problems, designing systems and understanding human behaviors by drawing on the concepts fundamental to computer science.

The prospective limitations including a definition of computational thinking and computer science has been discussed in the research fields and has not reached consensus. Research on CT and its relevance to students’ problem-solving skills was pioneered by Seymour Papert. He explained, in *Mindstorm*, that CT was closely related to “procedural thinking”, including developing, representing, testing and debugging procedures. When people design a program, it is important for them to use procedural thinking and plan their programs in terms of a sequence of what happens next, before, or until another action (Pea & Kurland, 1984). Programming is a valuable educational tool that provides the cognitive artifacts necessary for the human mind to build a representation of the world with which it interacts (Papert, 1981).

Numerous scholars argue that computational thinking refers to a cross-disciplinary set of mental skills that entail the basic elements of problem representation, abstraction, decomposition, simulation, verification and prediction. Hu (2011) argues that CT is a hybrid paradigm that accommodates different thinking modes with similar practices. Denning (2017) argues how that CT is a part of mental skills that can automate. It is not clear whether CT should be categorized under computer science, or is a subject that is able to connect the abstract knowledge representation in the mind, such as “connecting explicit knowledge into a concrete form” to student (NRC, 2011). Research suggests that professional development training workshops on the use of CT can help learners develop better understanding of CT and adopt relevant skills to participate in the digital world (Denning, 2017).

Systematic Changes and CT Classroom Practices

A systematic change that educators are facing to is how to support teachers in developing meaningful approaches to implementing CT practices in the K-12 classrooms. First, a few teacher education programs focus on training pre-service teachers to incorporate CT into K-12 classrooms (Yadav,

Gretter, Good, & McLean, 2017). The challenge has multiple aspects that include; how CT benefits students' understanding of fundamental disciplinary concepts and cross-cutting concepts found in the Common Core State Standards and the Next Generation Science Standards (Stephenson et al., 2016); how to identify promising pathways and pedagogical strategies that help preservice and in-service teachers infusing CT in their curricula; and how to embed CT in the disciplinary knowledge of each subject area in K-12 teaching to avoid the situation by simply "adding it on" to what they are already doing—not substantively changing what they do (Cuban, 2001; Means, Roschelle, Penuel, Sabelli, & Haertel, 2004; Sandholtz, Ringstaff, & Dwyer, 1997).

Secondly, assessments of CT remain underdeveloped and under-researched (Yadav et al., 2017) and this issue has been called out as a key computer science education research imperative (Cooper, Grover, Guzdial, & Simon, 2014). The few research efforts that have specifically targeted tackling the issue of CT assessment—especially in the context of activities involving programming—suggest that assessing the learning of computational concepts and constructs in popular programming environments is a challenge (e.g., Fields, Searle, Kafai, & Min, 2012; Koh, Nickerson, Basawapatna, & Repenning, 2014; Meerbaum-Salant, Armoni, & Ben-Ari, 2010; Werner, Denner, Campe, & Kawamoto, 2012; Werner, Denner, & Campe, 2015).

Thirdly, a few efforts have looked at the issue of transferring computational thinking skills beyond the common core curriculum. Transferring of learning is an aspect of assessment that deserves attention since computational experiences at various levels of K-12 aim to serve as bridges to future computational work. New approaches to transfer such as Preparation for Future Learning (PFL; Bransford & Schwartz, 1999; Schwartz, Bransford, & Sears, 2005) have shown promise in the context of science and mathematics learning at the secondary level (Zakaria, Chin & Daud, 2010; Dede, 2010; Schwartz & Martin, 2004).

Integrating CT into K-12 system poses systematic changes in schools. The personnel and organizational changes are complex for any groups, but perhaps particularly challenging for educators because of the complexity of the schooling process and the number of forces competing for the time and attention of teachers and leaders (Meier, 2015). As Fullen mentioned (2007), educational change depends on what teachers do and think - it is as simple and as complex as that (p.129). One key to help teachers and leaders understand the required changes of CT envision in the K-12 classroom is to build the active professional learning communities. An effective teacher professional development workshop needs to be developed for teachers to make ongoing improvements (Fullen, 2007). There is a need for the ongoing research work to foster the professional training communities, support teachers to model applicable steps to develop CT skills, and apply the cross-cutting concepts through the process of prototype building, creating and designing the prototype. The backward design process positions teachers as designers and helps them move from covering content to uncovering the big ideas (Meier & Richards, 2016, Wiggins & McTighe, 2005) in face of curriculum planning.

Project-based Learning (PBL) and Backward Design Principles

To gauge interest in the field of computational thinking, numerous studies have been accomplished to introduced educators to the field of computational thinking. Constructionism as a paradigm for teaching and learning provides the theoretical framework of project-based learning and backward design models. As a form of situated learning (Lave & Wenger, 1991), PBL is based on the constructivist finding that students gain a deeper understanding of material when they actively construct their understanding by working with and using ideas in real world contexts.

Project-based learning (PBL) starts with a driving question in a real-world situation that learners find meaningful and important (Krajcik, 2015; Nahum, Mamlok-Naaman, Hofstein & Krajcik, 2007). First, the driving question in PBL project design helps the participant focus on the main goal. Second, while engaging in the practice of PBL, participants are introduced to learning goals and technologies that support the inquiry process which is normally non-existent in the regular in-class projects: a process also known as scaffolding learning. Third, learners who develop these new skills can explore the driving question by participating in scientific practices in collaborative activities that are central to expert performance in the discipline. Fourth, participants can create a set of tangible products that address the

problems that become shared artifacts, a publicly accessible external representation of the class's learning (Lave & Wenger, 1991).

The pedagogical framework Understanding by Design (Wiggins & McTighe, 2005) as a backward design approach has played a crucial role in the educator professional development. UbD focuses on guiding educators through the process of designing curriculum units, performance assessments and classroom instructions, that integrate computational thinking. Research has shown that the project-based learning principle can maximize the chances that students be exposed to big ideas specified in the learning goal (Darling-Hammond, 2008; Larmer & Mergendoller. 2015). Also, engaging teachers in the design-centered approach and the backward design approach (Meier, 2005; Wiggins & McTighe, 2005) can deepen the content knowledge and pedagogical practices in CT .

Project-based learning hands-on projects can be used to model the integration of CT practices into classroom settings, of which the applicability of CT concepts in real world situations, as well as the application of tools and techniques from computing to understand natural, social and artificial systems and processes, is realized. The project-based learning design principle can be a powerful tool for allowing participants to transform computational thinking principles into real world problem solving skills.

Research questions

Research Question I: How can a technologically based CT program help educators increase their confidence and motivation for integrating CT concepts into curricula;

Research Question II: Is the project-based learning, hands-on project framework an effective model for integrating CT concepts into pedagogical practices.

Study design

This plot study focused on introducing participants to the applicability of computational thinking pedagogical practices into the regular classroom practices, and to create the understanding of how computers can function in different ways to solve real-world problems. First, the principal investigator (e.g., facilitator) conducted a pre-interview questionnaire to get information about participants' attitudes towards computational thinking and prior experience in the field of computer science and hardware design. Some of the questions included were, "What is computational thinking?", "What terms come to mind when you think of computational thinking?", "Do you think it is realistic to integrate computational thinking practices into regular classroom teaching environments?".

Next, the participant completed a short survey to indicate the applicability of computational thinking in the subject areas of English Language Arts, Math and Science by way of crosscutting concepts. Each question is rated on a scale from 1 to 5, (1 = the least applicable, 5 = the most applicable). The big ideas in ELA are key ideas and details, craft and structure, and integration of knowledge and ideas. Some of the key ideas in mathematics and science under the cross-cutting paradigm are; geometry and relationships, patterns and relationships, cause and effect, energy and matter, etc.

Next, the principal investigator walked participants through a hands-on robotic car assembly workshop while discussing concepts such as speed, velocity, acceleration that could be reinforced through the robot assembly. The principal investigator told the participants to imagine they were the designers and to develop an autonomous driving car of the future with the materials in front of them. The participants used the material provided to assemble the car without fixed instructions. The material consisted of two car wheels, two motors, two line follower sensors, one raspberry pi hardware, multiple female and male wires, and screws.

Lastly, the principal investigator guided participants in writing a computer program that simulated the car driving process in order to model how the computer processes signals. Participants used the Python turtle graphics programming environment to simulate the moving forward and backward procedures. The purpose of the variables and functions as well as how the computer execute the specific commands were to be explained.

To engage participants in scientific, engineering and mathematics practices, the workshop facilitator encouraged participants to develop a collaborative learning atmosphere, and to ask questions. Some of the questions asked included the mechanism of the self-driving cars, why sensor matters in the autonomous driving, and how to use Raspberry Pi could control the car's navigation. The inquiry and guiding process allowed participants to use their knowledge to explore emerging technologies.

Pre-Data Collection

In designing the program, the principal investigator surveyed 50 participants to collect the preference of the introductory projects to learn computer science related concepts at New York City. Eighty percent of participants indicated that they did not have any previous hardware design experience (i.e., using Arduino or other types of microcontrollers). Only 38% participants showed they do not have previous programming experience. When asked about participants' levels of interests in the following project (0 = the least interested, 4 = the most interested), a majority of the participants selected the Line Following Robot project.

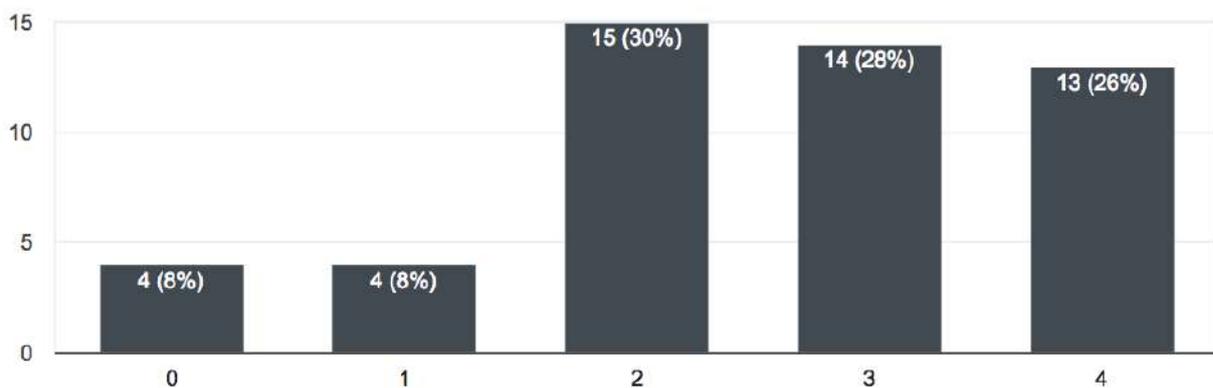


Figure 2. The level of interests in the project Line Following Robot . The vertical axis is the amount of participants, and horizontal axis is the level of interests (0 = the least interested, 4 = the most interested)

Methods

Participants

This research study used the mixed methods research methodology to collect qualitative and quantitative data from six participants, (three males and three females), who represent a group of pre and in service teachers in New York City. The workshop is approximately 45 minutes, and is videotaped to capture the entire process. There were no restrictions relating to teaching discipline of the participants. Participation in the study was completely voluntary. Convenience sampling was used to recruit participants who are affiliated with educational institutions of this research.

Procedures

The study followed a four-stage, iterative protocol. Step 1, participants discussed their understanding about computational thinking concepts and skills. Step 2, the participants were given the brief survey to rate the applicability of computational thinking in different areas. Step 3, the participants were asked to assemble a robotic car, without formal instructions, while discussing the relevant big idea concepts that could be applied to hands-on classroom exercises. Step 4, the participants were asked to use Turtle to simulate the car driving process. This interactive interview-play design allowed us to understand how the participant made sense of the hands-on project and the correlation of CT concepts. Along with recording the interview, we collected and analyzed the strategies that participants used in assembling the car and their programming command structure to support findings from our computational analysis.

Results

At the beginning of the workshop, participants believed that computational thinking was more applicable in science and math subject areas. The following table shows the average number that participants marked for the applicability of CT in the big ideas learning areas.

English Language Arts		Math		Cross Cutting Concepts	
Big ideas	Applicability	Big ideas	Applicability	Big ideas	Applicability
Integration of knowledge and ideas	2.3	Ratios & Proportional	2.6	Interdependence of Science, Engineering, and Technology	4.3
Craft and structure	3	Relationship	1.3	Cause and Effect	3
Key Ideas and Details	2	The Number System	2.3	Scale, Proportion, and Quantity	3.6
		Expressions & Equations	2.3	Systems and System Models	4
		Geometry	3.7	Energy and Matter	2.3
		Statistics & Probability	3.4	Structure and Functions	4
				Patterns	3.3

Figure 3. The participants rated from 1 to 5 in each category (1= the least applicable, 5= the most applicable) to integrate computational thinking into the listed content areas.

Computational thinking is a fairly new term to the participants, some of the definitions included

“decisions through mathematical computation.....”, and some of the participants believed that computational thinking was similar to the systematic thinking. It is similar to programming and is very technical as well as procedural. Also, participants expressed “ logic”, “ deduction” and “calculation” are the terms related to the concept of computational thinking.

After the car assembly exercise, a majority of the participants indicated their willingness and interest in integrating CT relevant pedagogical practices into their curricula. Below are some of the responses from the participants.

Level 1	Level 2	Level 3	Level 4
Have no confidence on CT integration; Reject CT curricula without rational reasons	Have less confidence on CT integration; Reject CT curricula with rational reasons	Have confidence on CT integration; Accept CT curricula without rational reasons	Highly confidence on CT integration; Accept CT curricula with rational reasons

<p>The teachers reject and did not accept to integrate CT into the classroom teaching.</p> <p>The teachers display a minimal understanding of how computational thinking can be applicable into the classroom setting - and little knowledge of their varied approach to integrate CT knowledge, skills into classroom practice.</p> <p>The teachers do not feel motivated about using project-based learning methods in the classroom teaching.</p>	<p>The teachers reject with rational reasons that they cannot implement CT curricula in the classroom.</p> <p>The teachers display generally accurate knowledge of how computational thinking can be applicable into the classroom setting, yet not to apply the knowledge into the classroom teaching and learning,</p> <p>The teachers give rational reasons why they do not feel motivated about using project-based learning methods in the classroom teaching.</p>	<p>The teachers accept the CT curricula and would like to implement CT curricula in the classroom.</p> <p>The teachers understand the nature of the CT, and purposefully to think about the how to design CT curricula into the classroom setting. However, the CT curricula is not authentic and fit to the school learning environment.</p> <p>The teachers show the willingness to integrate CT into the classroom currently or in the future without the concrete plan and rational reasons.</p>	<p>The teachers accept the CT curricula and actively brainstorm ideas to implement CT curricula in the classroom, schools or districts.</p> <p>The teachers not only understand the nature of CT, and also systematically acquire knowledge from several sources about how to design project based learning model to put CT curricula into practices.</p> <p>The teachers are highly motivated to brainstorm ideas to integrate CT curricula into the classroom with rational reasons and concrete plans.</p>
<p>“ There is no use to integrate CT in my classroom...”</p> <p>“ There is no applicability to integrate CT curricula into my daily classroom practice or student teaching work.....”</p> <p>“Computational thinking concepts are confusing and the hands-on practice is tedious....</p> <p>“ I did not see the implementable potentials of the project into my classroom.....”</p>	<p>“ I see the values to integrate CT curricula into my class, however, I cannot accommodate my daily teaching reality considering my school environment.”</p> <p>“ I see the applicability to integrate CT curricula into my daily classroom practice, however, I have to accommodate other initiatives to our leadership team at school.”</p> <p>“ I understand the concepts of CT and values of project-based learning, however, my teaching styles did not fit with the hands-on workshop deliverable methods.”</p>	<p>“ I see the values to integrate CT curricula into my class, and I would like to give a try! Although I am not sure how to achieve my goal.”</p> <p>“ There is a great potential to integrate CT curricula into my daily classroom teaching, however, I feel CT is about science and math, and has less focus on writing and reading skills.”</p> <p>“I see the values of project-based learning into my classroom. I am not sure how to design the project-based learning unit with the emphasized concepts of computational thinking.”</p>	<p>“There is a great potential to integrate CT curricula into the classroom, and I am ready to design the curricula with my colleagues in our maker space.”</p> <p>“I fully appreciate the workshop study that allows me to think about how to integrate CT curricula into my class. CT is more about the general skills about pattern recognition, problem solving and critical thinking. I would like to collaborate with my colleagues in the school to design the project-based learning focused unit.”</p> <p>“ I am excited to implement project-based learning hands-on workshop into my classroom. Also, I would like to modify the workshop, and think about how computational thinking concepts can connect to my own content teaching areas”.</p>

Figure 4. Rubrics to evaluate participants acceptances of the research study and CT curriculum implementation based on project-based learning model.

After this hands-on exercise, the participants showed more confidence and willingness to use CT concepts and hardware material in their educational environments. Some of the participants expressed

their willingness to utilize the modeled examples in the hands-on projects and to integrate into the kindergarten teaching practices. One participant explained the application to integrate CT into their language learning programs.

	Pre-interview	Hands-on Workshop	Post-Interview
No.1	Level 2	Level 3	Level 4
No.2	Level 2	Level 2	Level 3
No.3	Level 1	Level 2	Level 3
No.4	Level 2	Level 3	Level 3
No.5	Level 3	Level 4	Level 4
No.6	Level 1	Level 3	Level 3

Figure 5. Workshop participation result analysis

By the end of the workshop, participants changed the previous beliefs on computational thinking and realized that it does not have to be strongly correlated with high technical skill. It is a way of teaching that incorporates algorithmic, or procedural thinking, critical thinking and hands-on experiences that facilitate learning retention and better prepare students for the real world experiences.

Discussion and implication

Although computational thinking pedagogical practice are a bit more technical and new to a majority of the participants, technology driven programs may be necessary to increase teacher participation in CT practices. More research work needs to be accomplished to design a more robust hands-on project to guide educators towards that goal. To ensure participants were not intimidated by the proposed projects, the facilitator told the participant that complete assembly of the robotic car was not required. The principal investigator also gauged the stress level of participants when they believed they were to build the robot to completion. A workshop design that includes how to guide participants in building the robotic car to completion that minimizes stress level, needs to be explored. Some variables that need to be tested are the difficulty level of assembly with and without formal instruction and whether that plays into stress levels. Also, whether stress is necessary for completing the task.

Some of the critiques from participants are the car assembling project tend to be mechanical with less flexibility and does not foster participant creativity. A future hands-on project can allow participants to use their own imagination and create technology prototypes of their own choosing. Ideas about computational thinking stemmed from participant level of experience with technological ideas and concepts. Although user technology experience data was not collected at the beginning of the workshop, their experience level could be implied based on their attitudes towards their computational integration during the workshop. Without the prior experience and expertise in computer science, participants can not realize the direct connections between the computational thinking and content areas in ELA, science and Math. In the designed study, there should be the specific sections to offer some examples of the big area to the participants to familiarize their understanding at the beginning of the workshop.

More research needs to be done to capture ways of introducing technical concepts to educators in order to change perspectives adding computational thinking principles in K-12 education. Helping educators gain the experience and knowledge to integrate CT into the regular classroom settings is an ongoing process and requires an excitement and acknowledgement of necessity. Research supports the need to facilitate this change-in-agency work in the individual classroom of each teacher and the individual ethos of each school (Bull, Spector, Persichitte, Meier, 2016). This pilot study needs to be scaled to include a larger cohort of pre and in service educators.

To conclude, this study showed that teachers need to have an understanding of technology and ways to implement computational thinking in order for it to be introduced in curricula. Also, the technologically driven project-based learning design principles have the potential to increase participants' confidence and in consequence motivates them to consider applying computational thinking related pedagogical practices in their classroom.

References

- Bransford, J. D., & Schwartz, D. L. (1999). Chapter 3: Rethinking transfer: A simple proposal with multiple implications. *Review of research in education*, 24(1), 61-100.
- Bull, G., Spector, J. M., Persichitte, K., Meier, E. (2017). Reflections on preparing educators to evaluate the efficacy of educational technology: An interview with Joseph South. *Contemporary Issues in Technology and Teacher Education*, 17(1).
- Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). A future for computing education research. *Communications of the ACM*, 57(11), 34-36.
- Cuban, L. (2001). *Oversold and underused*. Cambridge, MA: Harvard University Press.
- Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends*, 59(2), 57-65.
- Darling-Hammond, L. (2008). Teacher learning that supports student learning. *Teaching for intelligence*, 2(1), 91-100.
- Dede, C. (2010). Comparing frameworks for 21st century skills. *21st century skills: Rethinking how students learn*, 20, 51-76.
- Denning, P. J. (2017). Computational Thinking in Science. *American Scientist*, 105(1), 13.
- Fields, D. A., Searle, K. A., Kafai, Y. B., & Min, H. S. (2012, February). Debuggems to assess student learning in e-textiles. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 699-699). ACM.
- Fullen, M. (2007). *The new meaning of educational change*. Routledge.
- Hu, C. (2011, June). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 223-227). ACM.
- International Society for Technology in Education. International Association for Computing in Education International Council for Computers in Education. (2000). *ISTE national educational technology standards (NETS)*. Eugene, OR :International Society for Technology in Education,
- Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge university press.
- Larmer, J., Mergendoller, J., & Boss, S. (2015). *Setting the standard for project based learning*. ASCD.
- Koh, K. H., Nickerson, H., Basawapatna, A., & Repenning, A. (2014, June). Early validation of computational thinking pattern analysis. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 213-218). ACM.
- Krajcik, J. (2015). Project-based science: Engaging students in three-dimensional learning. *The science teacher*, 82(1), 25.
- Means, B., Roschelle, J., Penuel, W., Sabelli, N., & Haertel, G. (2004). Technology's contribution to teaching and policy: Efficiency, standardization, or transformation? In R. Floden (Ed.), *Review of research in education* (vol. 27, pp. 159-183). Washington, DC: American Educational Research Association.

Meerbaum-Salant, O. (2010). Armoni, Michal a Ben-Ari, Mordechai (Moti). In Learning computer science concepts with Scratch. Proceedings of the Sixth international workshop on Computing education research. Aarhus, Denmark: ACM.

Meier, E. (2005). Situating technology professional development in urban schools. *Journal of Educational Computing Research*, 32(4), 395-407.

Meier, E. (2015). Beyond a digital status quo: re-conceptualizing online learning opportunities. Bank Street Occasional Paper Series 34. Retrieved from <https://www.bankstreet.edu/occasional-paper-series/>

Nahum, T. L., Mamlok-Naaman, R., Hofstein, A., & Krajcik, J. (2007). Developing a new teaching approach for the chemical bonding concept aligned with current scientific and pedagogical knowledge. *Science Education*, 91(4), 579-603.

National Research Council. (2011). Report of a workshop on the pedagogical aspects of computational thinking. National Academies Press.

Pea, R. D., & Kurland, D. M. (1984). Logo Programming and the Development of Planning Skills. Technical Report No. 16.

Richards, R., Meier, E. (2016). Leveraging Mobile Devices for Qualitative Formative Assessment. in D. Mentor, (Ed.), *Handbook of Research on Mobile Learning in Contemporary Classrooms* (pp. 94-115). Hershey, PA: IGI Global Publishing Company.

Sandholtz, J., Ringstaff, C., & Dwyer, D. (1997). *Teaching with technology*. New York: Teachers College Press.

Schwartz, D. L., & Martin, T. (2004). Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. *Cognition and Instruction*, 22(2), 129-184.

Schwartz, D. L., Bransford, J. D., & Sears, D. (2005). Efficiency and innovation in transfer. *Transfer of learning from a modern multidisciplinary perspective*, 1-51.

Seymour Papert, 1981, *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning (IJGBL)*, 6(1), 1-17.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). ACM.

Werner, L., Denner, J., & Campe, S. (2015). Children programming games: a strategy for measuring computational learning. *ACM Transactions on Computing Education (TOCE)*, 14(4), 24.

Wiggins, G. P., & McTighe, J. (2005). *Understanding by design*. Ascd.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6), 565-568. <https://doi.org/10.1007/s11528-016-0087-7>

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 205-220). Springer, Cham.

Zakaria, E., Chin, L. C., & Daud, M. Y. (2010). The effects of cooperative learning on students' mathematics achievement and attitude towards mathematics. *Journal of social sciences*, 6(2), 272-275.

What is Constructionism? Views from a Thai Perspective

Deborah A. Fields, deborah.fields@usu.edu
Utah State University, USA

Paulo Blikstein, paulob@stanford.edu
Stanford University, USA

Abstract

This paper investigates the interpretation and implementation of constructionist principles in Thailand. Interviewing key founders of the movement as well as teachers, village leaders, business people, and others vested for many years in the constructionism movement in Thailand, we ask what constructionism means within the diverse Thai community where it has been applied.

Keywords

Constructionism; developing countries; educational reform; Thai perspective

Introduction

In a 1991 paper, Seymour Papert faced the difficult challenge of defining constructionism, an educational philosophy that prided itself in not telling people what things are—after all, he was a constructivist who advocated that knowledge should be constructed and not simply transmitted. At the same time, pressure was mounting on Papert and his collaborators to offer a better definition of constructionism, which, alongside the explosive growth of the Logo language, was become increasingly popular and influential. It was one thing to define constructionism within the confines of Seymour's weekly seminars at the Massachusetts Institute of Technology (the "Loud Thinking" meetings), but defining it for a broader audience, in an academic article, was an entirely different business. Papert's response was that,

"If one eschews pipeline models of transmitting knowledge in talking among ourselves as well as in theorizing about classrooms, then one must expect that I will not be able to tell you my idea of constructionism. Doing so is bound to trivialize it. Instead, I must confine myself to engage you in experiences (including verbal ones) liable to encourage your own personal construction of something in some sense like it." (p. 1).

If Papert succeeded in steering away from contradicting his own principles by offering a set-in-stone definition of constructionism, intentionally or not he opened up a wide and complex set of possibilities for implementing and thinking about constructionism. In traditional academic circles, whenever a new theory or methodology is created, the proponents spend the rest of their lives making sure that their ideas are not "distorted" or wrongfully appropriated. By allowing his readers to "personally construct" their own definitions, Papert was doing the opposite. It sounds like a good idea in principle, but what happens when people outside of Papert's close circle of students and colleagues try to implement constructionism? What happens when people have indeed the license to appropriate it in very personal and culturally-aware ways?

This paper investigates one such case: the interpretation and implementation of constructionist principles in Thailand. In 1996 a group of MIT graduates from Thailand started a foundation to support what they hoped would be a transformative initiative to reshape education in their country. They brought Papert and later many other researchers from MIT, including his then student Cavallo, to visit Thailand and begin a project to introduce constructionism to the country, starting first with teachers then reaching out through non-formal education to people in rural communities (e.g., Cavallo, 2000). After a few years, the collaboration with MIT phased out and the movement was left largely to its own devices, with relatively little contact with U.S. institutions or researchers. The foundation, led by a few visionary figures

and several educators and leaders, with some support from the ministry of education and various corporations, continued to figure out by on their own how to implement constructionism in Thailand. Intriguingly the Thai leadership and people who joined the constructionist community applied the philosophy not just in K-12 education but also in industry (from chemical companies to banks), remote farming villages, technical colleges, and non-formal education. Interviewing key founders of the movement as well as teachers, village leaders, business people, and others vested for many years in the constructionism movement in Thailand, we ask what constructionism means in Thailand within the diverse community in which it has been applied.

Background

In order to understand how constructionism has been framed and practiced in Thailand, we must situate those meanings against the background of how constructionism has been defined and applied in other areas, namely the developed countries in the West (in the context of this paper, we will refer to the West as mostly North America and Europe, from which most of the constructionist research originates, but we are aware that there are active communities in Latin Brazil, Mexico, Senegal, Australia, and many other countries.) Because of Papert's intention to not overly define constructionism and to convey what it is through stories and interventions, it can be difficult to pin down what it is and how it has been applied. Yet given the substantial literature on the topic, we certainly *can* frame some key foci of the movement. Perhaps the most common definition of constructionism comes from succinct statements, such as that the following one by Papert himself (Papert & Harel, 1991):

“Constructionism--the N word as opposed to the V word--shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe” (p. 3).

Later, Ackermann (2001) tried to clarify the definition in broader terms, saying that Papert's constructionism “focuses more on the art of learning, or 'learning to learn'.” She further emphasized Papert's interest in learners engaging in conversation with their and others' artifacts and “how these conversations boost self-directed learning, and ultimately facilitate the construction of new knowledge” (p. 1).

These two sets of statements embody several themes in constructionist literature, including the roles of creating things/artifacts, engaging with others about these public artifacts, learning in a self-directed way, and developing new knowledge. Ostensibly constructionism could be applied to any human being in any context creating anything of relevance to them and others. Yet within these very general themes the *application* of constructionism has tended to be far narrower. For instance, it has not often been used to study the construction of knowledge related to sand castles or cosmological theories. At the risk of grossly overgeneralizing a rich movement in just a few paragraphs, below we suggest four overarching ways to understand constructionism in the West. We consider the foci of people, institutions, activity, and mechanisms of building constructionism. Our goal is not to oversimplify a philosophy that has been explored in multiple books, but to situate the simultaneously familiar and unique 20-year implementation of constructionism in Thailand.

People Focus

Given its roots in developmental psychology, published literature of constructionism has focused almost exclusively on children. In the early years of constructionist research, in the 1980s and 1990s there was practically no work on adult education, higher education, or workforce development. Though constructionists worked considerably with teachers, they were seen as a vehicle to ultimately reach the children. There was a strong child-centered focus both in terms of research and implementation, but also in terms of the philosophy of the movement. On occasion, Papert would state that technology was giving children the opportunity to “learn on their own” or make some of the teaching redundant (for

example, in a video-debate between Papert and Freire recorded in the 1990s⁴³). In contrast, in Thailand the constructionist movement has been broadly applied to people of a wide age span: children, youth, young adults, adults, the elderly.

Institutional Focus

Deriving from the focus on children, educational institutions have been the main focus of constructionism in the West. In 1985, MIT started the well-known project in the Hennigan School in Boston which implemented Logo across the curricula (Tabor, 1990), and in the 1980s and early 1990s, schools in tens of countries started to incorporate Logo in their curricula. Much of the work was bottom-up with teachers bringing it on their own to schools, but given Papert's and MIT's centrality and international reputation, they were approached by many big school systems and ministries of education to implement constructionism in entire countries or cities (e.g., Costa Rica, Thailand, Senegal, etc.). Eventually, with the "Logo crisis," of the late 1990s, there was a bigger focus on out of school implementations: extracurricular time, summer programs, and after-school workshops. Over the past two decades, especially with the emergence of games-for-learning and FabLabs, libraries, community spaces, and museums have also begun to play a strong role in constructionist education for children. Still, all of these are generally *educational* institutions. In the Thailand constructionist movement, there has been a triple-focus on schools, businesses, and rural villages, laying the groundwork for some potentially unusual applications of constructionism.

Activity Focus

The initial Logo implementations focused considerably on project-based learning—students were writing computer programs based on personal interest. Papert defined the terms "project" and "problem" in very particular way: "projects are primary, problems come up in the course of projects" (Papert, 1996). "Projects" were larger-scale personal endeavors, and within them students would find "problems" to solve. This was a specific reaction to the "problem" based approach in traditional math instruction, in which students are given different math *problems* to solve without a connection to a broader project. The definition of these terms is relevant in our context because "problem" and "project" are used in Thailand in slightly different ways. This focus on making things has continued in recent years, visible in the number of titles in Constructionism conference proceedings that focus on learning programming, making games, creating animations, developing system models, and so forth.

Mechanisms of Dissemination

It is impossible to deny the role that digital tools have played in disseminating constructionism throughout the world. Since Papert introduced turtle geometry and Microworlds, there has been an almost inevitable focus on designing tools that children can use as much as designing the learning environments those tools are used in. Indeed there are many constructionist tools and "construction kits" since the early days, including Lego/Logo, NetLogo, StarLogo, Scratch, electronic textiles, FabLabs, and digital games. Even though many of those tools were not originally designed as stand-alone education materials, they have been undeniably a primary means of disseminating constructionism to educational settings. Programs, workshops, and curricula are built around the use of a tool, albeit with reflection, social expression, and sharing with a broader audience as familiar attributes of these learning environments. These are some of the primary (though certainly not the only) ways that constructionism has been disseminated, along with some academic artifacts such as books, publications, and conferences.

Methods

Context and Participants

This is part of a larger study to investigate the historical development of the constructionist movement in Thailand. Constructionism was formally introduced in Thailand in 1997 and has continued there

⁴³ <https://www.youtube.com/watch?v=FnVCyL9BwS8>

through the present time. What started as a small movement with a foundation established in 1996, some government and business sponsors, and initial training and workshops led by Seymour Papert and David Cavallo has grown substantially. It would take a full article or perhaps a book to wholly describe the breadth and history of that movement, but here we seek to provide a little context to provide a backdrop within which to situate what constructionism means in Thailand. The Thai community has three overlapping domains of implementing constructionism: education (K-16 schools and non-formal education), business (chemical, agricultural, tech-driven, and financial institutions), and rural villages. It has spread through both word of mouth (ground-up) and by top-down support from business, village, and school leaders.

Implementing constructionism has involved many types of activities for the different individuals and communities involved. Many teachers have sought to apply constructionism in their classrooms, often expanding to a school-wide approach if school leadership has been supportive. A number of large corporations in Thailand have used workshops to train employees and subsequently supported these workers in identifying and working through problems in manufacturing, human resources, or community outreach. Villages have identified problems such as water management, sustainable agriculture, managing finances, improving family education, and so forth as areas to work with constructionism. Some of these areas are described in more detail by the Thai community itself (e.g., Israsena et al., 2014). None of these areas are without contestation. When a principal at a school or the head of a business changes often constructionist efforts have been thoroughly disrupted or ended altogether. As we discuss below, it can take months, even years, for people to accept a constructionist mindset. Yet the community has persevered, largely isolated from out-of-country resources, for more than two decades, raising the opportunity to study what constructionism means in Thailand and how it has developed there.

Data and Participants

During a two-week visit to Thailand in December 2017 one of the co-authors (Fields), with a translator as needed, conducted 1-2-hour semi-structured interviews with 22 participants nominated by leaders of the community with the goal of sampling from people long- and/or heavily-involved in one or more of three main domains of constructionism in Thailand: education (9 participants), business (8), and villages (5). An interview protocol and a list of interviewees was developed by the authors with collaboration from some other researchers familiar with the history of Constructionism in Thailand. With a few exceptions, most participants had more than 10 years of experience working with constructionism, including some who were involved from the very early years of the project. A few people were relatively new to constructionism but were highly involved in growing the constructionism project over the past 2-3 years. Geography and time limited the number of people who could be interviewed during this first visit of the larger study. Interviews focused on participants' histories with constructionism, key characteristics they thought important to constructionism, and positive and negative examples of how they had seen constructionism in practice. All interviews were transcribed in the main language the interviewee used—English or Thai depending in participant preference. Thai interviews were then translated into English.

Analysis and Limitations

Multiple rounds of grounded, comparative analysis (see Charmaz, 2002) were conducted on the interviews with the focus of identifying key attributes of constructionism in Thailand. We initially developed 16 codes grounded in the participants' descriptions that were applied to the entirety of the data. In a second round we compared, condensed, and reorganized the codes to better reflect the dominant themes emerging from the analysis. We identified four overarching themes spoken of by almost everyone interviewed, with several subthemes that provided further clarification. We do not claim that any of these themes are universally shared by all members of the constructionist community in Thailand. Findings in this paper emerged from discussion with a particular subset of very experienced leaders in the community and are likely not reflective of the thousands of individuals who have experienced and applied constructionism overall. Further, we as authors recognize that we are foreign to the Thai context and though we have been involved in varying levels with the constructionism community in Thailand for several years (primarily during 1-2 week visits, often as workshop leaders or

observers), it has been largely as outsiders. Member checking with two leaders of the constructionist community at multiple stages of analysis and writing has helped us check our understanding against that of participants.

Findings

Given that the Thailand community applies constructionism across a wide range of contexts, from K-12 schools to non-formal education, and from rural villages to large industrial companies, it should not be surprising that there was a wide range in what people valued and prioritized in their explanations and descriptions of constructionism. Yet there were several core themes that emerged across people's accounts. Intriguingly, what might be considered one of the core aspects of constructionism in the West, the idea that learning happens when creating something was mentioned by only a few people, namely those with fluent English who had an advanced degree from a U.S. university, and had read Papert's original works. So if the idea of "learning by constructing and sharing objects" is not the core of Thai constructionism, what is?

Control over One's Own Learning

One thing that nearly all participants spoke about in regard to constructionism was that learners were *in control* of their own learning and that this was a transformative attitude compared to traditional Thai culture. Boonkong⁴⁴, a businessperson who had spent extensive time studying and observing constructionism in Thailand, described this as a powerful transformation in one village that took up constructionism across the past 18 years: "[M]ost of the villagers in Chuenchit, they control their lives. Before their life was controlled by someone else. Now they control their life. They say that everyday they wake up, they feel happy, not because they're rich, because they can control their life" (p. 4). Participants reported that without constructionism, when people in Thailand encounter a problem, they either wait to be told what to do or report a problem to higher authorities rather than trying to figure out what to do oneself, whether in a business or a village. Similarly, in schools, children are told what to do and are in continual search for what the "right" answer is, something that many participants said was still a tendency in their adulthood. In contrast, taking control over one's own learning meant having the power to deal with issues that came up in one's life and actively doing something about it.

Implied in the idea of being in control of one's own learning was the notion that learning should be *interest-driven*, which came with both a sense of freedom and responsibility. In other words, people should get to choose what to do, whether in class, at work, or in community life. For instance, Saijai, a leader in her village, described that constructionism is "the learning process, that the learner is the center of the learning" (p. 3). Or as Punya, a long-time teacher, explained, students should have the freedom to "do whatever they want to complete [their] project[s]" (p. 3). Learning should be driven by learners, not by others in authority. At the same time, there was an emphasis on responsibility, on doing things oneself rather than having others do a project or solve a problem for them, though support was always available in the form of mentorship. As an example, Saijai described how she and other members in her village helped others begin to apply constructionism: "We didn't do a project for them, they have to do it by themselves with our support" (p. 5). Thus applying constructionism meant taking responsibility and ownership over one's life, one's problems, one's projects.

This belief that people should have ownership of their own learning had empowering implications for how people thought of themselves and others. As Pinit, a business leader and constructionist facilitator, spelled out, "I've changed from a person who lacks confidence, afraid to try new things, to have more confidence. I now know the beauty of learning," (p. 2). Further, just as participants like Pinit took more empowered views of themselves, they took similar views of the students they worked with, whether those students were children in schools, family members, or colleagues in companies, or fellow members of a village. Many teachers described the transformation they experienced as they witnessed constructionism in practice with their students. Students became more engaged, happier, spoke up more, presented with confidence, and demonstrated changed mindsets even years later. Thus not only

⁴⁴ All names are pseudonyms. Quotes are cited by page number of the interview.

did participants describe greater confidence in themselves, they expressed similarly strong confidence in the capabilities of the students or learners that they worked with.

Notably participants described learning this attribute of Thai constructionism in *practice*, not by a definition read in a book or heard in a lecture. Most participants first encountered constructionism through a one-week workshop on Microworlds Logo. Designed to put learners in an unfamiliar situation, participants often felt very uncomfortable at the beginning because they were not told exactly what to make or how to make it. Yet through the weeklong experience they *learned how to learn* and grasped that learning could be driven by their interests and learned through practice not through lecturing or being told what the right answer was. Not only this, but as participants went on to apply constructionism in their own jobs and families (many expressed applying this with family members), the support they received further emphasized the interest-driven and learner-centered characteristics of constructionist learning. For instance, one teacher, Samorn, explained that when she and her fellow teachers tried to figure out how to implement constructionism at their own school, their director “didn’t try to control us ‘how’ to do it, we managed ourselves,” figuring out how to design constructionist learning environments in their classrooms and grade levels (p. 2). This was emphasized again and again in participants’ interviews; they were never told how to implement constructionism, something that was occasionally frustrating but overall fulfilling. Instead, it was up to them to try it out, learn through a process of trial, error, observation, and trying again.

Real-World-Problem-Focused and Process-Driven

In the Thailand constructionist community, solving real-world problems and the processes of solving them are at the center of doing constructionism⁴⁵. Whereas in the West the types of projects that are most common are related to personal interests of the individual child, in Thailand the context of constructionism involved working on a *real-world problem*, usually one of relevance to a local community. In businesses, employees need to be able to identify what a problem is when it arises and find a solution to solve the problem themselves. In villages, “Rural people need deep understanding on their own problem[s],” finding the root cause of an issue, collecting data and community knowledge about that, and solving the bigger, underlying issue (Worawech, p. 17). Finally in schools, students can identify problems in their class or in their community and try to work on a solution to that. In all of this, people work toward bettering their local community, be it a school, neighborhood, village, business, or Thailand itself. In other words, the interest-driven emphasis of constructionism in Thailand is based on broader collective interests rather than personal intellectual inclinations or hobbies.

Solving these problems involves applying a *process*. Almost everyone interviewed explicitly mentioned the word “process” in discussing constructionism, and those who did not mention the actual word described a type of process or set of steps that they used to solve problems in a constructionist manner. There was variation to the processes described, but in general it involved identifying a problem, doing something about it, reflecting on the result, and adapting to continue to fix the problem. One senior facilitator of constructionism, Boonmee, summarized one process as follows: “Think, make, reflect are the core principles... Then rethink, remake, and reflect.” (p. 2). Perhaps one reason why *process* is so important in Thai constructionism is that it is a set of abstracted principles that can be applied to a wide variety of situations. As village leader Pana Pong explained, “In the past, everything is scattered and unorganized, not systematic, but when we learn things we can get into clear processes” (pp. 1-2). Like Pana Pong, many participants emphasized that the process of doing constructionism gave them power to deal with problems, work through mistakes, and persevere when things did not work out the first time. Anurak explained that the idea of a process may work well with common cultural ways of thinking in Thailand: “If you look at all the different interpretations of the learning process in Thailand, all of it is a cycle. They might have five steps, six steps, seven steps, eight steps, but it’s all a cycle” (pp. 4-5). Internalizing a process of constructionism allowed people to use it again and again.

Although people had slightly different expressions of what a process of constructionism involved, several overarching themes emerged across the interviews. First, the process involved *community*.

⁴⁵ Note that many constructionists would consider the term “problems” as an equivalent to “textbooks problems,” so here we used “real-world problems” to differentiate.

Identifying, understanding, and working through a problem was best done with others. Manit described this insight as transformative. Instead of facing a problem by himself and when often he could not find a way to solve it, “by sharing information with the other people, other workers, there are many solutions to solve the problems” (p. 3). Many like Manit shared that constructionism brought people together, whether in a classroom, village, or workplace. Teachers intentionally supported peer pedagogy, villages thought about issues together in ways that built community, and workplaces experienced teamwork. Pinit shared that constructionism “made us understand or empathize with others’ perspectives and ideas” (p. 4). He explained that before technicians and engineers did not share their expertise in working through issues, but that constructionism helped them to listen to each other, share ideas, and work with “cross-disciplinary knowledge.” It would be all too easy to stereotype a community focus as relating to broad ideas about collectivist versus individualist cultures. But that idea does not explain why groups across the spectrum of Thai culture had previously failed to collaborate on solving problems before they were introduced to constructionism. Constructionism may have opened up deeper, more collaborative communities than were able to exist before.

About half of the participants mentioned managing *emotions*, often through *meditation*, in their descriptions of a constructionist process. They spoke about this as a means to concentrate, stay focused, understand oneself, and be receptive to constructive criticism. As Boonmee explained, “If you can’t control your mindfulness, you will get angry and you can’t think thoroughly about the cause of the problem you made” (p. 2). He shared that this was very personal to himself. Before constructionism he had a very short temper and was especially unreceptive when people criticized his teaching. But after learning to apply constructionism he was calmer and think about the other person’s perspective when they gave criticism. For many, managing their emotions, especially anger, allowed them to listen deeply to others and also helped them to be more humble. This all helped them to broaden their perspectives on the problem or project they were working on and to get help from others—to be more open to their community in productive ways.

Finally, about half of the participants also cited *reflection* as a core attribute of the constructionist process. This goes back to the “think, make reflect... rethink, remake, reflect” process that Boonmee described. It was important to take time to think about what one had done, evaluating what went well and what to change. Pimchan highlighted the importance of reflection in the changes her village experienced once they began to apply constructionism:

“Because normally, in the normal village, when they have activities or the things that they have to do, after those activities end, then it is finished. But our village has applied constructionism to the working process. And when we finish one activity, we have a reflection time asking whether or not this one is good? This one is bad? Are we going to continue doing this?” (pp. 6-7).

Like Pimchan, for those participants who mentioned it, reflection was tremendously powerful to their learning experiences. It gave them the ability to improve, to persevere through initial failures or imperfections, and to “crystallize” their knowledge. Those who were teachers saw a further value to reflection in that it enabled them to see their students’ thinking. Samorn, a retired teacher of 3rd graders explained that when she changed to constructionism, her students can “express into words so I know that they can think!” (p. 3). Reflection allowed students to externalize their thoughts so that they could be shared and seen by others.

Many of the characteristics shared here should be familiar to constructionist audiences, though some may be a surprise. It is not a far stretch to shift from a project focus (in the classical sense used in the West of personally-meaningful endeavors) to a problem focus (i.e., community problems in the real world,) especially since the types of problems solved in the Thai community often involve projects to fix them (e.g., water management projects, industrial design machines, agricultural plans). Further, in terms of an emphasis on process, the ideas that Thai leaders shared here are not far from processes such as “design thinking” that involve idea-generation, creation, testing, and evaluation common to many implementations of constructionism. Although reflection is far from a universal characteristic of constructionist literature, it has had an explicit part of some early constructionist work, such as the design notebooks that children in Harel’s software and game design environments used to keep track

of changes they made in their projects every day (Harel & Papert, 1990). In terms of community, Papert himself suggested some “social criteria” for the learning environments he promoted, drawing on the very social Samba School to compare it to the traditional “lonely, impersonal” mathematics environments. Managing emotions and using meditation may seem the most unusual characteristics of constructionism to western mindsets, but if we note that this is to facilitate focus and concentration in problem solving and to be more open to criticism on one’s work, then it may be something we can learn from.

Contested/Conflicting Roles of Digital Technology

While digital technology has played a key role in many, if not most, reports of constructionism in the west, it has a contested role in constructionism in Thailand. On the one hand, nearly all participants were first introduced to constructionism through a technology-centric, weeklong workshop on Microworlds, Gogo Boards, or photojournalism. This workshop structure was inherited from the initial work by Papert and Cavallo when they introduced constructionism in Thailand (see Cavallo, 2000). Since then local leaders started to create their own version of a trio of weeklong Microworlds, photojournalism, and Gogo Board workshops. The emphasis in these workshops was creating a situation where attendees could “learn how to learn” in a way they were unused to. These tech-driven workshops allowed participants to learn teamwork, meditation, reflection, the constructionist process. As the director of a technical school explained, “Tools are the first step—used to encourage learners so they can understand and see the process of learning” (p. 1). Then learners could figure out how to apply constructionism in their own locales. This training was common for people in all contexts - industry, villages, and education.

A few people continued to view technology as core to their implementation of constructionism and sometimes as essential to constructionism more broadly. One teacher who first experienced a Microworlds workshop 18 years ago immediately implemented it in her grade 6 classroom and continued to use technology as her primary way to share constructionism with her students, shifting to Scratch and Crickets (an early platform for robotics) as she learned of newer, more contemporary tools that worked with the computing systems available. A former teacher and now school leader, Nisa, prioritized introducing various digital technology in the school where she has worked for many years: starting with Microworlds and then shifting to Scratch, Gogo boards, and now FabLabs as ways to engage children. She spoke of a desire to emphasize the computational side of constructionism more in Thailand and her school. Another participant, Anurak, is a computer engineer and focuses most of his constructionist work on “building things to build other things with,” designing new technologies that are affordable for everyday Thai people to design and think with. However, outside of a few members like these, most participants did not necessarily focus on digital technology in how they put constructionism to practice.

Some participants, especially those in areas of industry and rural outreach, expressed concern that “digital technology is not essential for doing constructionism” (Worawech, p. 10) and that it could actually be a “block” to people in industry and rural areas if they became too focused on a particular technology (Boonkong, p. 7). After all, most people in workplaces do not use Microworlds or Gogo boards as professional tools. These participants hinted at a misunderstanding by workshop participants that constructionism was only relevant in the context of digital tools. Boonkong, an engineer at a technical firm sought deeper understanding of the key elements and outcomes of the now-traditional introductory constructionism workshops: “[W]hen we know those key elements, there’s no need of Microworlds, photojournalism, or Gogo Boards anymore. We can all create our own tools,” (p. 8). Similarly, one of the primary leaders of the introductory workshops thought that the community should not be stuck on particular technologies. Instead it might be better to shift to broader “project-based or work-place learning” (Boonmee, p. 5) so that participants could see the potential for using constructionism in their areas of employment or everyday life. The range of opinions and concerns about the role and importance of digital technology in constructionism raises it as an issue in the broader field worldwide. Constructionism was introduced in the context of digital tools, but Papert clearly drew insight from and saw applications in other areas, such as Samba Schools (Papert, 1980) and in sand castles (Papert, 1991). Yet in Thailand there is concern about over-associating constructionist with a particular tool. It raises the question of what *is* the role of digital technology in constructionism, historically and in contemporary times?

Constructionism as a Life-Altering and Lifelong Pursuit

One very striking element that participants spoke of was what they saw as the lifelong, transformative aspect of constructionism. Learning to think and act in a constructionist way involved taking up a new mindset that affected all aspects of life, both present and future. Teachers saw helping students (from elementary to vocational schools) to learn constructionism as valuable not just for particular academic-based thinking skills (e.g., computing) but as something that they would use for a lifetime. “The most important key of constructionism is to develop people with lifelong learning skills,” (Mana, p. 3). Taking on a constructionist mindset meant being able to learn and continue to improve for one’s entire life, regardless of problems one might need to face. Teachers reported that students came back years after their education was over to say how much they had learned from the experiences in the constructionist-based classes. That they had the ability to make decisions, to learn how to learn, to take an active approach to their own learning. Participants regarded training people in a constructionist manner as something that would enable them to improve their lives and contribute to their communities.

Participants saw a changed constructionist mindset and thus a trajectory of lifelong learning as key to transforming their local communities, workplaces, Thailand, and even the world and society as a whole. This is nowhere more evident than in the ways that people took constructionism from the contexts in which they first used it and applied it to broader areas, often in their volunteer time or in their retirement. Having received so much themselves from particular mentors over the years, many expressed the desire to give back, to help people have “a happy learning experience and hope they can do more for our nation in the future” (Pinit, p. 4). A transformed mindset ready to tackle problems through a thoughtful, iterative and reflective process within a shared community was important for the individual, community, society, Thailand, and the world.

At the same time participants spoke clearly about how difficult it was to help people develop such a changed mindset. “Constructionism is like growing a tree. It takes time,” (Mana, p. 4). Working for over a decade with students in a vocational school program, Mana explained that it took six months to two years for students to understand and adopt constructionism as a mindset with changed habits, and even then 10% of the students still did not change. It often took longer for some teachers at the school, but when they saw the evidence in the form of how students thought and acted (especially through a trial program where some students were taught in a constructionist program and others were not), they were willing to shift. Complicating this was that often results of constructionist projects were slow, especially if they dealt with challenging problems in a community. Anurak said that some projects in elementary schools he worked with took 1-3 years to finish. Pana Pong, familiar with nearly 18 years of educational, agricultural and financial changes in his village, spoke of a 4-5-year timespan to see the results of constructionism. Yet everyone thought the investment worthwhile. Constructionism took a “lifetime” of learning but was a “process that can pass on to different generations, young or elders or even kids” (Pana Pong, p. 4). In the Thailand constructionist community, this was about transforming their society over the course of individual lives and across generations within society.

Conclusion and Discussion

Papert taught the philosophy of constructionism in a way that prioritized people learning it in a constructionist manner themselves, through stories and personal experiences. One could argue that the Thailand constructionist community’s 20-year process of learning and growth well illustrates the affordances of this approach as they built their movement largely in isolation from the broader worldwide community. In doing so they extended constructionism to a number of settings that are not often considered in the literature of the broader constructionism community, namely businesses, non-formal education, and rural villages. The values and practices that they ascribe to constructionism show both cultural integration and transformation. They brought culturally relevant practices validating process, emotions, and community to their implementations in ways that, at least for them, transform how they think, live, and relate. The way they conceptualize constructionism also raises a number of important questions for us to consider as we explore what this means for the broader constructionist community.

One question this study raises is what the *starting point* for constructionism can be. There are many examples in constructionist literature of the starting point for an educational intervention being a

particular digital technology (or a set of technologies, i.e., in a makerspace), albeit one with openness for learners to design something of personal relevance to them. However, in Thailand the starting point is more often a problem of relevance to a local community, which might lead people to a range of different solutions which may or may not involve particular tools.

A related question is what the role of technology should be in constructionism. Despite Papert's insistence that constructionism can involve making sand castles or cosmological theories, using technology to do things that can't be done (at all or as effectively) with paper and pencil is one of the central foci of western constructionism (for a discussion see Kafai, 2006). One of the motivators for this may be a prioritization to introduce computational literacy, but it may be helpful to tease apart the more domain-centric drive to support computation and more broadly applicable philosophy of constructionism. Perhaps it is time to revisit epistemological considerations (e.g., syntonicity, epistemological pluralism, powerful ideas) or pedagogical strategies (e.g., audience collaboration, reflection) that have received far less attention in constructionist literature.

Constructionists have always had a strong identification with the work of Paulo Freire and other scholars in critical pedagogy, due to their agreement on issues of student-centeredness, real life relevance, and student empowerment. However, Freirean scholars had a much stronger stance on issues of power within the classroom ("dialogical education") and on the broader role of education in society (e.g., education as a way towards personal "emancipation"). In many school systems and countries we might anticipate that even though policy makers are enthusiastic about constructionist principles, they might not have anticipated that, if fully implemented, those principles will upend some of the canons of traditional educational policy: being able to predict, test, and track students, the ability create environments that are predictable and stable, the respect for authority in classrooms or schools, and the lowering of cost. Some interviewees raised some of those questions, and we believe that, if constructionism keeps expanding in Thailand, at some point these larger societal issues will come to the fore of the discussion.

Acknowledgments

This work was supported by the Suksapattana Foundation and the Lemann Center for Entrepreneurship and Educational Innovation in Brazil. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Suksapattana Foundation, Utah State University, or Stanford University. Special thanks to Nalin Tutiya-phuengprasert and Arnan Sipitakiat for substantive constructive feedback and earlier reviews of this paper.

References

- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*, 5(3).
- Cavallo, D. (2000). Emergent design and learning environments: Building on indigenous knowledge. *IBM Systems Journal*, 39(3.4), 768-781.
- Charmaz, K. (2002). Stories and silences: Disclosures and self in chronic illness. *Qualitative Inquiry*, 8(3), 302-328.
- Harel, I., & Papert, S (1990). Software design as a learning environment. *Interactive Learning Environments*, 1(1), 1-32.
- Harel, I., & Papert, S. (1991). *Constructionism*. Norwood, NJ: Ablex Publishing Corporation.
- Israsena, P., Wongviriyawong, C., Sipitakiat, A., Tutiya-phuengprasert, N., Tantikul, T., Limpiti, N., Rattanathavorn, I. & Cheamsawat, S. (2014). Constructionism in Thailand and its transformative effect on the lifelong learning process. In *Proceedings of Constructionism*, Vienna, Austria. Retrieved online at http://constructionism2014.ifs.tuwien.ac.at/papers/2.1_2-8590.pdf on March 30, 2018.

Kafai, Y. B. (2006). Constructionism. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 35–46). New York: Cambridge University Press.

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, 39(3.4), 720-729.

Papert, S. (1996) An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, Vol. 1(1), pp. 95-123.

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex

Tabor, M. B. (1990). A Lab to Re-invent Communication. *The New York Times*. Retrieved online at <https://www.nytimes.com/1990/12/26/news/a-lab-to-re-invent-communication.html> March 30, 2018.

Personal Learning Journeys: Reflective Portfolios as “Objects-to-Learn-With” in an E-textiles High School Class

Deborah A. Fields, *deborah.fields@usu.edu*
Utah State University, Logan, UT, USA

Mia S. Shaw, *mshaw12@gse.upenn.edu*
University of Pennsylvania, Philadelphia, PA, USA

Yasmin B. Kafai, *kafai@upenn.edu*
University of Pennsylvania, Philadelphia, PA, USA

Abstract

Much attention in constructionism has focused on the design of learning tools and support for students building artifacts. Far less attention has been placed on reflection and reflective artifacts that let students consider their own learning. The design of their electronic textile projects during an eight-week curricular unit in an *Exploring Computer Science* class, an introductory computing course for high school (secondary) students in the United States. We examine the portfolios as sites of student self-authorship: places where students showed agency in positioning themselves in relation to how they made their e-textile projects and to computer science more generally. In the discussion we consider the implications of reflective portfolios as “objects-to-learn-with” for educational implementation and constructionist pedagogy.

Keywords

Electronic textiles; computer science education; identity; portfolios; objects-to-think-with

Introduction

“Studying one’s own learning process...can be a powerful method of enhancing learning.”

(Papert, 2001, p. 85)

A lot of attention in constructionism has centered on how how learners can construct knowledge through the design and interaction with artifacts ranging from Logo programs, video games, robots, and animations to 3D-printed and laser-cut objects (Blikstein, 2013; Harel, 1990; Kafai, 1995). To support young designers in constructing objects on and off the screen, hundreds of tools have been designed, including programming environments like Logo, Scratch, Squeak, Blockly, Snap!, and Alice in addition to construction kits such as Lego Mindstorms, LilyPad Arduino, MaKey MaKey, Chibitronics, and many more, working with materials such as Lego bricks, textiles and paper. A major emphasis has been on what kids can make and what they learn by making through “objects-to-think-with” that engage them with powerful ideas (Papert, 1980) and through “objects-to-share-with” that foster interactions with others (Kafai & Burke, 2014). An emphasis in this work is highlighting children’s capabilities to make and create when they are provided with powerful tools such as computers. In these situations, children and youth can make interest-driven, student-centered projects that are highly engaging and can promote rich learning through the process of creating these objects.

Far less attention has been placed on how learners can make their own learning an equally important focus as Papert (2001) suggested (see quote above) or formally reflect on their own learning with what we would like to call “objects-to-learn-with.” Creating artifacts and reflecting on both the process of making them and the products themselves engages students not only with disciplinary learning, but just as importantly supports students’ authorship of their own identities or senses of self within a particular discipline, i.e., a “disciplinary identity” (Van Horne & Bell, 2017). Being able to author one’s own

disciplinary identity may be especially important in a field such as computer science (CS), a field that is well-documented as historically exclusive and in which many students often struggle to develop a positive sense of self or sense of belonging (Cheryan, Plaut, Handon & Hudson, 2013; Yardi & Bruckman, 2007). To help students articulate a clear sense of self or self-narrative in relation to a discipline presents a critical dimension of the learning process (Carlone, 2017) and would suggest involving learners in documenting their own development in design notebooks or portfolios that could capture their reflections on their learning processes and outcomes. Such reflective artifacts could engage students in making their own learning an object-to-learn-with—just as much as the actual design of games, robots or software already does—and uncover how they see themselves establishing (or not) personal connections with academic disciplines.

In this paper we share the design and analysis of learner-generated reflective portfolios that accompanied the development of a series of electronic textile (hereafter e-textile) projects that high school students completed during an eight-week curricular unit in an introductory *Exploring Computer Science* (ECS) class (Goode, Chapman & Margolis, 2012). E-textiles are hybrid designs, consciously combining traditionally “masculine” activities such as engineering and computing with traditionally “feminine” activities such as crafting and sewing (Buechley, Peppler, Eisenberg & Kafai, 2013). Portfolios have been extensively used as a means of assessment, for instance, in studying how students reflect on their academic learning by documenting computational thinking concepts and practices learned (Chang et al., 2015; Lui, Jayathirtha, Fields, Shaw, & Kafai, 2018). However, here we wanted to focus on another equally important aspect of constructionist learning—that of personal expression and connection to the academic discipline—which could uncover critical aspects of students’ identities as learners in the field of CS. This led us to consider the portfolios as sites of student self-authorship, places where students could show agency in positioning themselves in relation to how they made their e-textile projects and to CS more generally. In this paper we ask, how do students use reflective portfolios to express their own voices and self-authorship in relation to computer science? In the discussion we consider the implications of reflective portfolios as “objects-to-learn-with” in constructionist learning environments where students can personally express themselves as well as the ramifications for educational implementations and constructionist pedagogy more generally.

Background

Early constructionist activities such as the instructional software design (Harel, 1990) or game design projects (Kafai, 1995) included reflection in the form of design notebooks. For instance, when children designed Logo software and games, they also wrote out daily reflections on the “problems and changes they made each day” (Harel & Papert, 1991, p. 6). These journals or design notebooks encouraged students to reflect on what they were learning and supported self-awareness of the process of creating as students noted how their projects changed as they moved through the process of making them. While notebook entries captured changes in students’ thinking and designs, such reflections tend to be under-analyzed in comparison with the actual programmed artifacts (e.g., games, software) that were created by learners (Reynolds & Caperton, 2011). A further limitation is that even the learners considered the notebooks transitional or process artifacts, as they ultimately did not become part of students’ final artifacts that were shared with a wider audience.

In contrast, portfolios have begun to emerge as potential culminating artifacts in some classrooms to showcase students’ work, accompanying the actual project(s) that students have made. For instance, portfolios featuring the best projects or series of projects students have made, as in a professional art portfolio, highlight students’ accumulated competency and skill (Býrgýn & Baký, 2007). Portfolios can also be process-based, showing students’ progress through a series of projects or even within a project (Chang, et al., 2015). Here students can narrate how they have grown throughout the creation of a single project or across multiple projects, allowing students to author their own pathways of learning through reflection. While portfolios are more common in arts education, more recently they have appeared in CS education. For example, in the Advanced Placement Computer Science Principles class, students submit both a project and a portfolio that explicates the intention behind their projects as well as documents how they were made (College Board, 2017). Thus the portfolio supplements the project itself and can be used as a type of learning assessment for academic content.

In addition to serving as a learning assessment or a demonstration of learning progress, the type of narration that portfolios encourage holds equal potential to be a key resource in supporting students' authorship of "disciplinary identity" (Van Horne & Bell, 2017). Identity is a broad concept that, from a sociocultural perspective, includes how people act in particular situations (practice-based identities), how people think about themselves (self-narratives), and how other people perceive someone (others'-narratives) (Fields, 2010). Though creating personalized artifacts linked to core disciplinary content can support students' identities in practice-based ways (Van Horne & Bell, 2017), the design process alone may not necessarily help students articulate a clear sense of self (i.e., self-narrative) in relation to a discipline. In this paper we are primarily concerned with the aspect of identity that relates to students' self-narrative—how they think about themselves (Fields, 2010). In discussing identity development, Nasir and Cooks (2009) argue for the importance of "ideational resources" in developing and establishing one's identity within a particular learning space. Ideational resources are the "ideas about oneself and one's relationship to and place in the practice and the world, as well as ideas about what is valued and what is good" (p. 44). Ideational resources might include specific lessons or sayings about how to manage one's emotions in a challenging scenario or how to name oneself in relation to others, for instance as a core participant in a community (i.e., a "hurdler" or a "jumper" in track-and-field or a "problem solver" in a computer science class). Having a sense of self as a type of computer scientist or as having characteristics of capable computer scientists can situate students on an inbound trajectory of participation in a discipline.

In turning to portfolios for capturing students' identities as learners, we suggest that they can become an ideational resource in helping students express who they are in relation to an academic discipline or field, namely computer science. By combining the process features of the early design notebooks of constructionist projects (Harel, 1990) with the product features of portfolios found more in arts education, we can turn reflective portfolios into meta-artifacts, or "objects-to-think-with", in which students express what, why, and how they have made a computational artifact and thus begin to narrate, in their own personalized ways, who they are within the field of CS. Below we describe the type of portfolio that students made in the e-textile curricular unit, a portfolio that was both project-based (featuring a series of artifacts students made) and process-based (highlighting challenges and revisions made in the process of making the projects). Then we explore the ways that students expressed and authored themselves through the personalized portfolios that they created, ending with a discussion about the implications for educational implementation and constructionist pedagogy.

Methods

Curriculum & Portfolio

Over the past two years we have co-developed an e-textiles unit for the *Exploring Computer Science* curriculum, a year-long course providing an introduction to computing with equity-focused and inquiry-based teaching (Goode, et al., 2012). The e-textiles unit took place over eight weeks and consisted of a series of four projects: 1) a paper-card using a simple circuit, 2) a wristband with three LEDs in parallel, 3) a classroom-wide mural project where pairs of students created portions that each incorporated two switches to computationally create four lighting patterns, and 4) a "human sensor" project that used two aluminum foil conductive patches that when squeezed generated a range of data to be used as conditions for lighting effects. Each project allowed increasing flexibility in design and personalization, and the human sensor projects reflected this in the diversity of students' projects: stuffed animals, paper cranes, wearable shirts or hoodies, handbags, and gifts for family members.

In the second year of implementation, with the help of two teachers, we revised the unit, adding reflective portfolios as a capstone to accompany the final project of the unit. The portfolio that students created was co-developed with Ben, the teacher of the ECS class that is the subject of this paper. The portfolio was both project- and process-based, showing the series of projects students made as well as reflections about their processes of making them. The portfolio consisted of a set of at least four Google Slides for each project, with students adding on slides until the portfolio contained reflections on all four projects. For each project, the requirements included: 1) an initial drawing of the project and a reflection on changes made to the project, 2) at least one challenge they faced and an explanation of how they

dealt with it (Figures 1 and 3), 3) how they had “grown as a computer scientist” accompanied by at least one picture or video of their work in progress (Figure 2), and 4) a picture/video of the final project with an explanation of what it did plus a reflection about what they learned and how the project fits into their identities as computer scientists. The portfolio ended with one final reflection on students’ learning across the entire unit. Figures 1, 2, and 3 provide examples of how students chose to illustrate the different parts of the portfolio assignment, as well as how students personalized their portfolios. All but one student personalized their portfolios by using specialized fonts and backgrounds, adding title pages for each project, creating digital representations of their projects, and incorporating selfies or images that exhibited their own styles and interests.

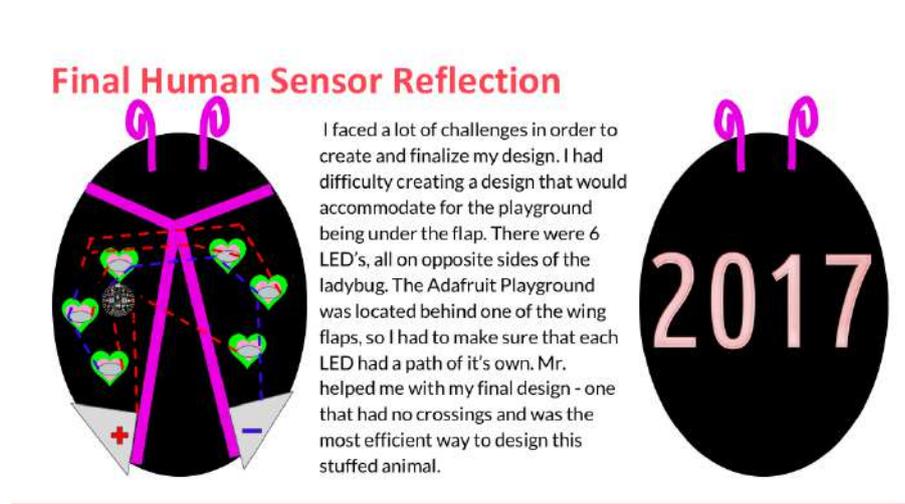


Figure 1. Ashley's portfolio page on challenges in the human sensor project.

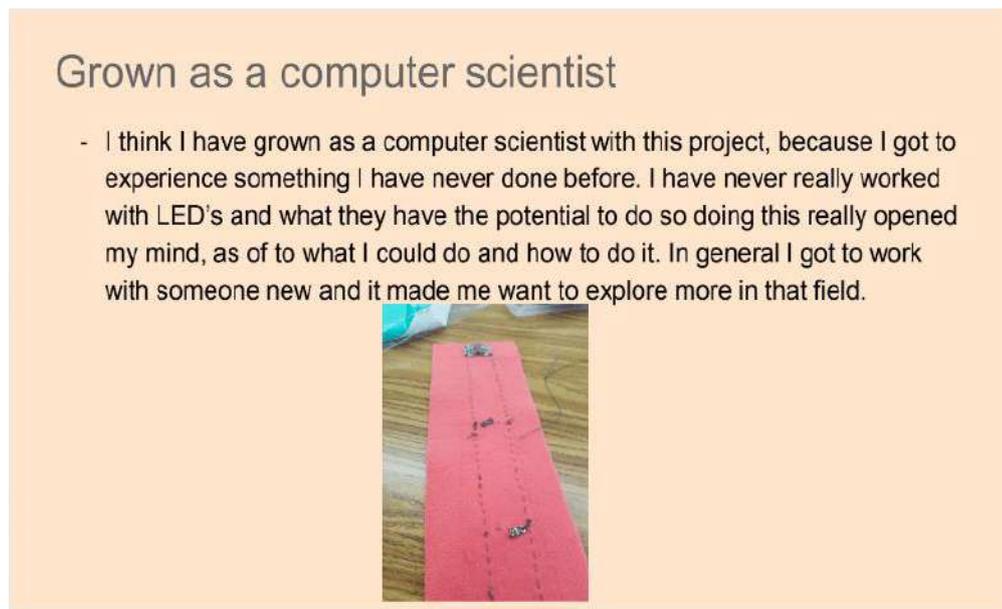


Figure 2. Alejandra's portfolio page on growing as a computer scientist after the wristband project.

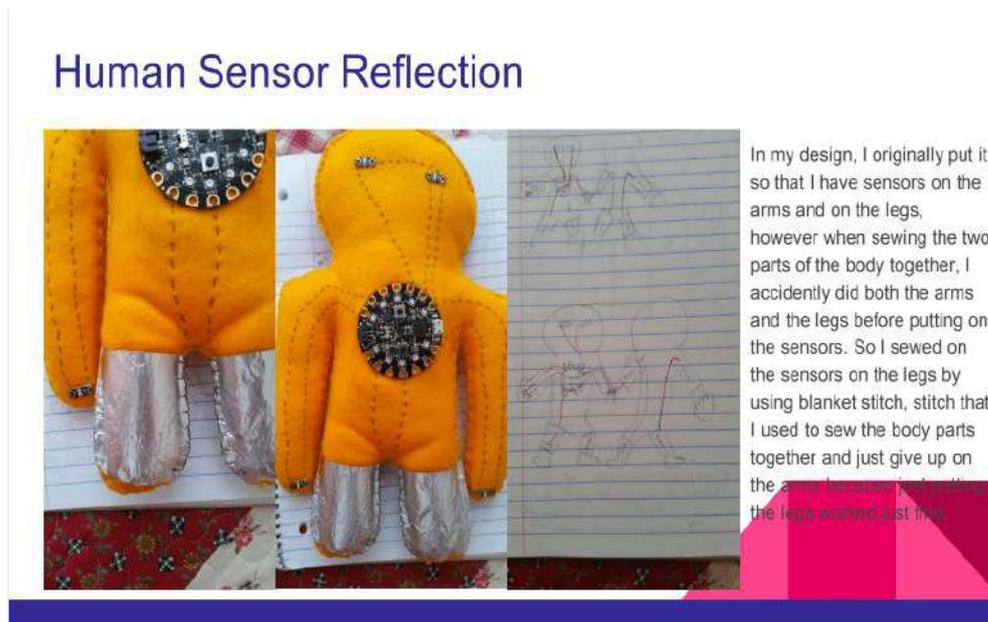


Figure 3. Louis's portfolio page on challenges in the human sensor project.

Ben piloted the e-textiles unit in Spring 2017 in his ECS class with 35 students (13 girls and 22 boys), with 32 consenting to participate in the research project. Most students were in 9th grade (14-15 years old); one student was in 12th grade (17 years old). At Ben's school, 54% of students were identified as socioeconomically disadvantaged as defined by the State of California and included the following demographics: 4% African American, 18% Asian, 10% Filipino, 40% Hispanic or Latino, 25% White, 1% as two or more races, and 2% race not reported). The school had a three-year trajectory of elective (i.e., optional) computer science courses, with ECS being the introductory course.

Data and Analysis

The data for this project included the digital portfolios from all 32 consenting students in Ben's class, and we sought to develop a framework to analyze how students utilized the portfolio assignment to author themselves in relation to CS. We conducted multiple rounds of grounded, comparative analysis (Charmaz, 2002). We initially developed 11 codes stemming from the identity statements students generated as answers to the question "How have I grown as a computer scientist?" However, we found these codes limiting, as we noticed that students related to and framed CS in more subtle ways outside the identity statements. In a second review of the portfolios, we expanded our analysis to the entire portfolio, developing several new categories in addition to the original ones. We created a dictionary of codes with examples for clarity, then two researchers independently coded a portion of the portfolios until they reached uniform agreement. They then proceeded to separately code the remaining portfolios, soliciting second opinions for borderline areas as needed.

In further analysis we compared and contrasted the categories of codes, looking at areas of commonality and areas of exception. Finally, we analyzed student narrative statements that did not fit any of our six categories to consider what they revealed about the portfolios, reflection, and student narratives about CS. Together these stages of analysis helped unveil how the portfolios afforded students the space to articulate how they identify with CS. In the end we framed our research in the six broad categories below:

Self as a Computer Science (CS) Person

- **Computational:** Students author themselves as problem-solvers and relate this to doing computer science or being a type of computer scientist.
- **Personal:** Students explicitly link growth as a computer scientist with specific skills they learned, such as coding or making circuits.

Self and CS as Personal

- **Socioemotional:** Students describe how doing the project or the project itself demonstrates personal characteristics like dedication, perseverance, patience, getting out of one's comfort zone, making mistakes, or collaborating with others.
- **Relational:** Students express a relationship with a friend, family member, or teacher that either provided feedback on the e-textile project, involved collaboration, or made a project intended for someone else.

Self in Relation to the Larger Field of CS

- **Aspirational:** Students discuss themselves in a future tense in relation to computer science in the context of applications outside the classroom such as future jobs or in other projects.
- **Projectional:** Students describe new realizations of what computer science is or what it can include.

Findings

Throughout our analysis of Ben's students' portfolios, we discovered that in addition to serving as a tool to assess student learning of computer science content (Lui et al., 2018), the portfolios afforded students the space to author themselves in relation to CS. In constructing their portfolios, students not only articulated who they were in relation to CS but identified the various resources, skills, and personal qualities that helped them construct their artifacts. In addition, the portfolios allowed students to author new and expanded understandings of CS as a field. They also allowed students to narrate who they could be in the future in relation to CS, articulating what lessons they learned that could help them succeed in CS beyond this unit.

Identifying as Types of Computer Scientists

Answering the question "How have I grown as a computer scientist?" in their portfolios afforded students the opportunity to author themselves explicitly as computer scientists. Many students did so by articulating *specific functional skills* they exhibited throughout the unit that they believed aligned with being a computer scientist. In total, 20 students said that they had grown as a computer scientist through developing specific skills: 14 of these students listed specific coding skills, 12 cited circuitry skills, and 7 mentioned crafting skills (some students cited multiple skills across their portfolios). For instance, Leon claimed that the project helped him become a better computer scientist by his ability to code and design light functions. Other students linked being a computer scientist with things such as making LEDs light up, learning how to program a microcontroller, and even how to create a fabric that contained LEDs. The frequency of students linking being a computer scientist with particular skills is striking and shows the association they saw between being a computer scientist and the things that they learned in class while creating projects—they saw a direct link between what they were doing and who they were as computer scientists.

This direct link between doing and being is also visible in the ways that students associated being a computer scientist with problem solving. In all, 17 out of 32 students authored themselves as *problem solvers* specifically in relation to computer science. Jarvis expressed this the most obviously by writing that the project helped him as a computer scientist because in his words, a lot of computer science is problem solving. In this, Jarvis made a direct statement that computer science is problem solving, and since he did some problem solving in his work, he therefore was "helped" as a computer scientist in the making. Other students explained this connection between being a computer scientist and problem solving. For example, Jeevan "realized his mistakes so he would not make the same mistakes the next time, Anita shared that the project experience allowed her to solve problems on her own and work until she figured out a solution (Anita), and Ana expressed that her final project illustrates that her identity of being a computer scientist is she understands what to do but messes up a lot. This connection between problem solving and identity is intriguing because problem solving is a more general skill or way of thinking than particular forms of coding, crafting, or making circuits discussed above. Students' highlighting how to problem solve or work through challenges reflects broader, intentional teaching practices in the classroom community valuing mistakes and failure as legitimate means of learning (Fields, Kafai, Nakajima, Goode & Margolis, in press). It also reflects the values behind the requirements

of the design notebook and the portfolio, showing that students respected challenges and the processes of working through them.

In positioning themselves as having new skills in computer science as well as abilities in problem solving, these high school students challenged the common notion of not being “smart enough” to participate in computer science (Yardi & Bruckman, 2007). In other words, as opposed to being told what computer scientists do or what kinds of people they are (which can act as a barrier for participation in computer science), the students personally authored what they did throughout the e-textiles unit and how that confirmed their identities as computer scientists.

Relating to Computer Science as Personal and Social

Intriguingly, beyond listing particular skills and practices, 12 students also linked demonstrating socioemotional characteristics to being computer scientists. Consider Ashley’s explanation about being creative and taking risks, in which she shared that the project fits into her identity of a computer scientist because it allows her to go out of her comfort zone and create something new. She further explains that creating doesn’t only mean putting everything together, but imagining, discussing, evaluating, and understanding. Ashley explored an expanding view of creativity with her willingness to step out of her comfort zone as important personal attributes she associated with being a computer scientist. Other students expressed how they felt “more capable” (Heidi), how their project work improved their “patience” (Anita) or how the process of making e-textiles demonstrated “hard work” that developed through “constant practice” (Adeep). For these students, computer science was more than just learning specific skills or coding; they linked CS to involving personal attributes that they valued.

In addition, students particularly challenged the stereotypical perception of computer science being antisocial (Yardi & Bruckman, 2007). In total, 18 students acknowledged *relationships* as serving a role in the construction of their artifacts, whether it was by working with a partner on the mural project, gaining help from their peers or the teacher, or eliciting feedback from peers or family members. Heidi, for example, came up with her project idea by eliciting feedback from both her mother and her peers on whether to design a bag or pants and a corgi or flowers. In other words, Heidi valued people’s opinions in her design process, truly conceptualizing her e-textile project as an “object-to-share-with” (Kafai & Burke, 2014). Four other students specifically designed their e-textiles artifacts as gifts for family members, integrating relationships in the purpose of their designs. For example, both Jeremy and Ana designed their electronic cards for their mothers, while Ashley and Sara designed their human sensor projects (a ladybug and a wolf, respectively) for their little sisters. Designing their projects for family members and leveraging those relationships created additional meaning and relevance for students in considering the role computer science plays in their lives.

What is interesting about students noting how relationships played a role in their projects rests on the fact that most of their projects (aside from the mural project) and their portfolios were individual assignments. However, in reflecting on their learning through their portfolios, over half of the students recognized how others played a role in their being able to construct their projects, demonstrating how without explicit prompting they were able to author an expanded understanding of computing as a social field. By acknowledging the social aspects of computer science, students who normally view the field as antisocial can develop an increased interest and sense of belonging, such as Alejandra who expressed that she got to work with someone new, which made her want to explore more in the field.

Situating Selves with Computer Science as a Field

The portfolios also afforded students the space to author new and expanded understandings of computer science as a field. In analyzing all of the portfolios, we found that 14 students described new realizations of what computer science is outside the classroom. For example, Kevin learned that coding can be found in everything when he came to realize that lights required a software program to turn them on and off. Other students shared discovering new ways that computer science is done. For instance, as opposed to computer science being a highly regimented field with limited ways to participate, Adeep reflected on how the project helped him grow as a computer scientist by showing him that computer science does not have a handbook. It should be noted that students were not required to reflect on computer science as a field. However, in reflecting on their identities as computer scientists, students

positioned themselves in the field by broadening its criteria in ways that validated their participation. In addition, they reflected on how what they did in completing their projects related to how computer science is done in the real world, which speaks to how they were able to create meaning and relevance from their projects.

Other students, 12 in all, authored themselves as participating in e-textiles projects in the future, including Shona who expressed interest in doing both of the projects again due to the wonderful experiences she had. In addition to wanting to make e-textiles in the future, students like Leon and Ashley identified specific lessons they learned from the projects that they would apply for future e-textiles projects, such as listening more carefully in order to make fewer mistakes (Leon) or creating an initial plan before designing (Ashley). By identifying these lessons, there were given the opportunity to reflect on how they could improve upon their initial designs and experiences, increasing their chances of success. Other students like Jesse expressed interest in exploring computer science past the e-textiles class, particularly in college. Interestingly enough, one student, Mario, explicitly stated that he did not want to be a computer scientist in the future but acknowledged that it could help him get a job, become a skill, or help his family with computer problems. Even though Mario had expressed in his portfolio that he did not want to be a computer scientist, he still named meaningful ways he could apply what he learned in the e-textiles unit to his future.

It is fascinating how students used the space of their portfolios to author these future selves in relation to computer science. This reflects how portfolios can serve as ideational resources that allow students to construct identities of themselves participating in computer science in the future in ways that are personally relevant. Doing so affords students the opportunity to develop positive self-narratives that situate them on an inbound trajectory of participation in CS.

Discussion

When Papert (2001) wrote about valuing the study of one own's learning, he was reviewing his own journey—the various experiences, activities, and observations of learning—that contributed to his conceptualizing learning as a multi-faceted process of connecting with the world. Students engaged in very much the same approach in reviewing their learning as they created their portfolios in addition to and in review of their e-textile artifacts; both served as powerful ways to reflect on their own learning process and identities as e-textile designers. Not only did students apply computer science in real-world contexts through constructing their e-textiles artifacts but constructing their portfolios provided them with the opportunity to author new understandings of computer science in ways that were meaningful to them as well as increased their interest in the field. In other words, as opposed to holding negative perceptions of computer science, students were able to reframe their perceptions of computer science in more nuanced ways as a more relevant, sociable, and engaging field.

From “Objects-to-Think-With” to “Objects-to-Learn-With”

We started out with the observation that constructionist theory and practice have emphasized the construction of artifacts as “objects-to-think-with”, on and off the screen, as a primary vehicle for learners' knowledge reformulation and personal expression. Adding to this is the importance in constructionism of creating objects in a supportive social environment. In creating shareable artifacts, learners' knowledge construction becomes not only an individual but also as a social process. One could argue that by designing for an audience, learners have an opportunity to share the understanding they have gained from making something and in the process revisit their learning. We have called this dimension “objects-to-share-with” (Kafai & Burke, 2014). For instance, in Harel's (1990) instructional software design project learners developed software to teach others. However, in this instance the point of reflection was always another person—customizing software for the user—not the designer. The reflective portfolios studied in this paper move the examination of learning back to the learner him or herself. While “objects-to-think-with” have student-created artifacts as the focal point, reflective portfolios become “objects-to-learn-with”, places where students can trace their path of learning, record what they have discovered, and situate themselves as learners within a larger disciplinary community.

Identifying with Process and Mistakes

The portfolios also served as ideational resources (Nasir & Cooks, 2009) that supported students' authorship of who they were (i.e., students' expressions of identity) in relation to computer science. Most students in Ben's class clearly articulated themselves in relation to computer science, whether expressing that by describing the skills they learned, the types of problem-solving and persevering people they had become, or the friends, family and teachers who were part of their learning experiences. It is further evident from students' reflections that many felt able to clarify the *type* of computer scientist they were: as someone who "messes up", fixes things, thinks out of the box, works collaboratively with others, or "solves problems on my own." Here students were not limited to culturally dominant stereotypes about the types of people who do computer science (Yardi & Bruckman, 2007). Instead they linked their personal interests and experiences in authoring *new* ways of participating in the field of CS.

There are interesting parallels and links between the reflective portfolios students wrote and the e-textiles artifacts they made. Both are interest-driven and student-centered; students molded projects in relation to their own personal interests and shaped their portfolios in similar ways. Both had constraints that helped shape what students made. In the e-textiles projects, students were constrained by using certain tools or creating a number of lighting patterns; in the portfolios students had to fill in specific sections with writing and evidence. Both allowed for personal aesthetics. E-textiles projects varied based on students' interests and showed visible links to family, schools, or popular culture; e-textiles portfolios differed in fonts, backgrounds, digital images, and discourses of writing. In this we argue that portfolios like the ones students created in Ben's class are a type of secondary artifact, one that can accompany the type of personalized object usually focused on in constructionist-based learning environments. Yet while the primary artifact (i.e., an e-textiles or another project) only shows the end product, portfolios complement this by sharing the stories behind the e-textiles artifacts' creation: troubles encountered, mistakes made, revisions that took place, and people who influenced the design. While the primary artifact hints at students' identities in the skills exhibited or interests made visible, portfolios allow students to explain in explicit ways what type of people they are and how they relate to the broader field in which they are participating.

Constructing Spaces for Learning Journeys

Just as we design tools and communities for supporting students' constructions we need to attend to the design of tools for reflection. Some key affordances of the portfolios in this study could be applied to portfolios or similarly reflective artifacts in other contexts. One requirement of the portfolio in this study was to write about challenges faced and changes made in the projects. This constraint may have had a direct influence on students' prioritizing qualities such as problem solving, troubleshooting, persevering, and having patience. Another helpful requirement was specifically designed by the teacher: explaining "how you have grown as a computer scientist." Ben created this requirement to explicitly support students' identity development, and his success is evident in the frequency with which students associated certain skills or personal qualities with being a computer scientist or doing computer science. In future research we are investigating variations of these design features as well as other formats and contexts to understand how to facilitate students' reflections.

For instance, in our case, the portfolios were created in Google Slides format, very much resembling PowerPoint presentations. Yet there is an implicit contradiction in using a presentation format that is often introduced as visual medium with as few words as possible for a reflection journal that asked students to elaborate in writing about their creative process. While the visual component is invited in presentation format, the textual component is not. In the next iteration of the portfolio assignment which is being implemented in more than a dozen schools, teachers are choosing different formats for their classes, including blogs, websites, documents, and, of course, slides (though those could be oriented as a landscape or portrait format in ways that make them more or less like a traditional document). We plan to analyze to what degree different formats support students' creativity and self-expression in their reflections. Further, we are maintaining and adding to core constraints for the portfolios, while allowing teachers to have the flexibility to add to them. For instance, the new portfolio assignment has doubled the emphasis on challenges and revisions (each student must share at least two instances of facing a challenge or revision) but has no requirement to express how students have grown as computer

scientists. We will study to what degree students continue to author themselves explicitly in relation to computer science with and without that obvious prompt.

In this study, we focused on understanding how the construction of a personal and shareable artifact itself *with* the accompanying construction of a reflective portfolio could support students' learning journeys. Portfolios as meta-artifacts or objects-to-think-with deserve their own place in thinking about the philosophy and pedagogy of constructionism and how the combined artifacts can contribute to students' growing self-narratives in a broader disciplinary way.

Acknowledgements

This work was supported by a grant #1509245 from the National Science Foundation to Yasmin Kafai, Jane Margolis, and Joanna Goode. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, the University of Pennsylvania, or Utah State University. Special thanks to Tomoko Nakajima, Debora Lui, Justice Walker, and Gayithri Jayathirtha for their help with data collection, portfolio design, and constructive feedback.

References

- Blikstein, P. (2013). Digital fabrication and 'making' in education: The democratization of invention. In J. Walter-Herrmann & C. Büching (Eds.), *FabLabs: Of machines, makers and inventors* (pp. 1-21). Bielefeld: Transcript Publishers.
- Buechley, L., Peppler, K., Eisenberg, M. & Kafai, Y. (Eds.) (2013). *Textile messages: Dispatches from the world of e-textiles and education*. New York, NY: Peter Lang Publishing Group.
- Býrgýn, O., & Baký, A. (2007). The use of portfolio to assess students' performance. *Journal of Turkish Science Education, 4*(2), 75.
- Carlone, H. B. (2017). Disciplinary identity as analytic construct and design goal: Making learning sciences matter. *Journal of the Learning Sciences, 26*(3), 525-531.
- Chang, S., Keune, A., Peppler, K., Maltese, A., McKay, C. & Regalla, L. (2015). *Open Portfolios: Maker Education Initiative Full Research Brief Series*. Retrieved from: <http://makered.org/opp/publications/>
- Charmaz, K. (2002). Stories and silences: Disclosures and self in chronic illness. *Qualitative Inquiry, 8*(3), 302-328.
- Cheryan, A., Plaut, V. C., Handon, C., & Hudson, L. (2013). The stereotypical computer scientist: Gendered media representations as a barrier to inclusion for women. *Sex Roles, 69*(1-2), 58–71.
- College Board (2017). Advanced Placement Computer Science Principles Course Guide. Retrieved from <https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf>
- Fields, D. (2010). *Trajectories of identification across social spaces: Intersections between home, school, and everyday settings*. Unpublished dissertation. University of California, Los Angeles.
- Fields, D., Kafai, Y., Nakajima, T., Goode, J., & Margolis, J. (in press). Putting making into high school computer science classrooms: Promoting equity in teaching and learning with electronic textiles in *Exploring Computer Science. Equity, Excellence, and Education*.
- Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: The *Exploring Computer Science* program. *ACM Inroads, 3*(2), 47-53.
- Harel, I., & Papert, S (1990). Software design as a learning environment. *Interactive Learning Environments, 1*(1), 1-32
- Harel, I., & Papert, S. (1991). *Constructionism*. Norwood, NJ: Ablex Publishing Corporation.

Kafai, Y. (1995). *Minds in play: Computer game design as a context for children's learning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kafai, Y. & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: The MIT Press.

Lui, D., Jayathirtha, G., Fields, D., Shaw, M., & Kafai, Y. (2018). Design considerations for capturing computational thinking practices in high school students' electronic textile portfolios. In the proceedings of the *International Conference of the Learning Sciences*, London, UK.

Nasir, N., & Cooks, J. (2009). Becoming a hurdler: How learning settings afford identities. *Anthropology & Education Quarterly*, 40(1), 41-61.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Papert, S. (2001). Personal thinking. In C. Paechter, R. Edwards & P. Twining (Eds.), *Learning, space and identity* (pp. 78-86). London, UK: Sage Publishing.

Reynolds, R., & Caperton, I. (2011). Contrasts in student engagement, meaning-making, dislikes, and challenges in a discovery-based program of game design learning. *Educational Technology Research and Development*, 59(2), 267-289.

Van Horne, K., & Bell, P. (2017). Youth disciplinary identification during participation in contemporary project-based science investigations in school. *Journal of the Learning Sciences*, 26(3), 437-476.

Yardi, S. & Bruckman, A. (2007). What is computing? Bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the 3rd International Workshop on Computing Education Research* (pp. 39-50). New York, NY: ACM.

Constructionism and De-Constructionism as Complementary Pedagogies

Jean M. Griffin, jean.griffin@temple.edu
 Temple University, USA

Abstract

Constructionism is advantageous for learners of all ages but underutilized in “serious” formal education settings. It can be challenging for teachers to allocate enough time for students to construct their own designs if the students are expected to master numerous concepts and skills. Also, some teachers lack experience with evaluating creative constructions. Students often want to abandon their designs when they encounter obstacles or bugs. Constructionism might be more widely adopted if teachers knew how to pair it with a pedagogy that ensures coverage of key concepts, provides effective practice with core skills, and helps students gain confidence with troubleshooting. This paper presents such a pedagogy, de-constructionism, a learning-by-taking-apart approach grounded in reverse engineering and practice theory. An experiment is described where students learning to program solve practice problems designed with a de-constructionist approach. Students’ attitudes about the problems are reported. Suggestions for balancing constructionist activities with de-constructionist ones are discussed.

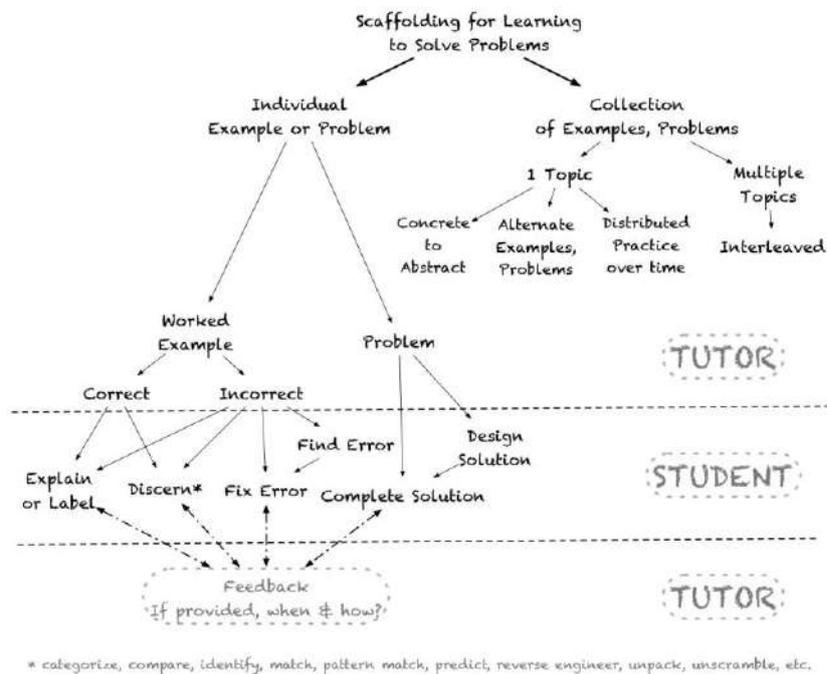


Figure 1. Model for De-Construction

Keywords

Constructionism; De-Constructionism, code comprehension, bugs

Introduction

Constructionism is a pedagogy and learning theory conceived by Seymour Papert that encourages learners to create personally meaningful computational artifacts and share them with others (Papert, 1980, 1987). Constructionism enjoyed a wave of popularity in the late 1990s when numerous education initiatives engaged young children with the Logo programming language to explore computing, math,

animation, storytelling, and other topics. By the early 2000s Logo's popularity had waned but new constructionist technologies emerged that attracted an even larger number of participants from a wider age range. Examples include Scratch, MIT App Inventor, LilyPad Arduino, and littleBits.

There are a few well-known college-level courses that use constructionist technologies and approaches at the beginning of a "serious" programming course (Malan & Leitner, 2007) or throughout an introductory computer science (CS) course that is also suitable for high school (Guzdial, 2003; "Mobile CSP," n.d., "The Beauty and Joy of Computing," n.d.). Despite such examples, constructionism is often perceived as a pre-college "just for fun" pedagogy. Although there is a constructionist component to the new American advanced placement (AP) *CS Principles* course, which requires student to design and create a computational artifact, teachers are typically not informed that constructionism is a pedagogy relevant to the course ("AP Computer Science Principles," n.d.; Kick & Trees, 2015).

It would be helpful if teachers of such courses, and teachers of traditional CS courses with a problem-solving focus, knew about constructionism – its history, guiding principles, and varied techniques (for designing, managing, and evaluating constructionist projects) – because students of all ages are motivated by making their own computational creations. For teachers who already take a constructionist approach, it would be helpful to know how to counter-balance its focus on design with a pedagogy that focuses on analysis and skill building. The Background section of this paper outlines the theoretical framework for such a pedagogy, de-constructionism. The Method and Results sections describe an experiment with a university Python programming course where the lab exercises were designed with a de-constructionist approach. The Discussion section considers the experiment within a constructionist/de-constructionist dialectic. It discusses future work, suggests ways to incorporate student collaboration, and invites others to apply de-constructionism to topics other than programming.

Background

The Background section discusses guiding principles for de-constructionism: learning from taking apart well-built examples, learning from taking apart well-built examples with intentional errors, and learning through effective practice. Note that de-constructionism is used here to mean a pedagogy; this is distinct from uses of the term deconstruction relative to literary analysis, social theory, architecture, or as a general decomposition strategy, e.g. (Boytchev, 2015; Self, 1997).

Learning From Taking Apart Well-Built Examples

How do people become mechanical engineers? Car mechanics? Fashion designers? In disciplines that involve physical objects, there is a long tradition of learning-by-taking-apart. Future mechanics and mechanical engineers are often drawn to take apart and fix cars, appliances, gadgets, and toys. Future fashion designers often take apart and then copy others' garment designs while developing their own unique style. While constructionism encourages this practice to some extent through remixing, a remixing experience (which involves changing an existing artifact) is often serendipitous. Students may choose to remix only an object's surface features, and either avoid or be unaware of its more complex features.

This section reviews three learning-by-taking-apart pedagogies. Two are used for hands-on engineering education: mechanical dissection and Tod Phod Jod. The third, *worked examples*, has been researched extensively for mathematics education and to some extent for computing education. Also discussed are a few additional learning-by-taking-apart techniques used in computing education. These approaches are compared and contrasted to find best practices applicable to a general pedagogy of de-construction.

Mechanical Dissection

In job training and vocational education settings, it is common for apprentices to disassemble, analyze, and assemble (DAA) vehicles, appliances, and other machines. This is useful for developing not only mechanics, but mechanical engineers and designers (Seabrook, 2010; Wu, 2008). For college/university mechanical engineering departments, this presents challenges in terms of workspaces, staffing, and storage. Thus, many offer introductory courses that are theoretical. An

exception is a Stanford University course that implements a hands-on DAA approach called mechanical dissection (Sheppard, 1992).

Mechanical dissection is similar to the medical school practice of dissecting cadavers, but instead students disassemble (and re-assemble) mechanical things such as bicycles, fishing reels, and drills. Later courses engage students with design activities. Guiding questions for the mechanical dissection course are: “How did others solve a particular problem?” and “Why does the solution work?” An important goal is for students to become familiar with the terminology of mechanical engineering. This is accomplished by reading instructions, conversing in a structured team setting, answering questions, and writing reflections. Students are guided to study product diagrams, and to notice labels and categories of parts and systems. Students also explain what they learned, and generate their own labels and categories in written reports and reflections. Names, labels, and categories all serve as metacognitive cues (Bransford, Brown, Cocking, Donovan, & Pellegrino, 2000). Researchers who observed participants of a Toy Dissection module report “We observed most groups generating a causal chain, in which they traced how a sequence of events propagates from one part of the device to the next” (Roschelle & Linde, 1996).

Some mechanical dissection activities have students analyze multiple items from the same product family, e.g. coffee machines made by different companies. Students compare them and evaluate design tradeoffs. Comparing and generalizing helps students learn design templates, develop design strategies, and think abstractly. Sheppard says: “The reality is that very little design is actually new design. Very good designers have this catalog in their brains of stuff – of mechanisms, of devices, of machine elements” (Wu, 2008, p. 59). Over thirty institutions have implemented mechanical dissection (Agogino, Sheppard, & Oladipupo, 1992; Regan & Sheppard, 1996; Roschelle & Linde, 1996; Wood & Agogino, 1996). It has been adapted to teach about products that aren’t solely mechanical; this is called product dissection.

Tod Phod Jod

An innovative approach to hands-on engineering education in India is *Tod Phod Jod*. It is similar to mechanical dissection in many respects but offered to children through a series of out-of-school workshops, each several hours long. *Tod Phod Jod* roughly translates to “break and make.” In the first phase, middle school aged boys and girls deconstruct and reconstruct objects such as ceiling fans, clocks, and irons. Later they take apart more complicated things and fashion new things (Jods). Like mechanical dissection, *Tod Phod Jod* encourages students to work hands-on with one or more teammates, and ask questions such as: “How does it work?,” “What’s inside?,” “Who made this?,” “Why?,” “How?” Also similar is the anticipation that discoveries participants make during *Tod Phod* (deconstruction) will come in handy later on for *Jod* (design). Yet another similarity is the goal of learning relevant terminology, by “deconstructing the scientific jargon that is usually learned by rote” (Vishnoi, 2012). *Tod Phod Jod* builds on the lower levels of Bloom’s taxonomy (remembering and understanding) to engage youth with higher levels – applying, analyzing, evaluating, and creating (“*Tod Phod Jod: To Encourage Students to Discover, Experiment, Innovate,*” 2012). Unlike mechanical dissection, participants of *Tod Phod Jod* do not fill out detailed worksheets and reports. Instead the emphasis is on having a fun and motivating experience. India’s national initiative to promote youth interest in STEM education recommends *Tod Phod Jod* for hands-on, activity-based learning (“*Rashtriya Avishkar Abhiyan,*” n.d.).

Both mechanical dissection and *Tod Phod Jod* promote learning through reverse engineering, given this definition: “Reverse engineering is the process of analyzing a subject system to identify the system’s components and their interrelationships and create representations of the system in another form or at a higher level of abstraction” (Chikofsky & Cross, 1990, p. 15). Both help students develop mental abstract representations, which can later be draw upon for design. There are other approaches for learning by taking apart physical things, e.g. un-crafting (Murer, Fuchsberger, & Tscheligi, 2017).

Worked Examples

Another approach to learning from analyzing well-made examples is the use of worked examples. John Sweller introduced this pedagogy, where students learn from studying problems that are completely

worked out as an expert would solve them (Sweller, 1988). Contrary to the prevailing belief that students should spend a lot of time solving problems, Sweller's research showed that learning can be more efficient if students study worked examples while gradually learning to solve problems on their own. This is known as the worked example effect (Sweller, 2006). According to cognitive load theory, which Sweller developed as the theoretical foundation for this research, humans can learn and store many schemas (patterns of knowledge) in long-term memory, but have a quite limited amount of working memory that can make sense of new information. Learning can be efficient if the cognitive load is managed and the learner doesn't experience cognitive overload due to excessive or confusing information. Scaffolding (instructional support) is needed to manage cognitive load. Worked examples can provide such support (Sweller, 1988; Sweller & Cooper, 1985).

Mathematics education researchers have conducted numerous large-scale experiments with worked examples to measure students' learning gains. After decades of research, experts recommend that students spend roughly the same amount of time studying worked examples as they do solving problems on their own (Booth, McGinn, Young, & Barbieri, 2015). It is important to consider prior knowledge; students with high prior knowledge may lose expertise if required to study too many rudimentary examples (Kalyuga, 2007). Empirical research attests to the benefits of several techniques used in conjunction with worked examples. These include self explanations, where students explain in their own words or choose an explanation (Chi, De Leeuw, Chiu, & Lavancher, 1994), comparisons (Rittle-Johnson & Star, 2007), sub-goal labels (Catrambone, 1998), and feedback (Conati & VanLehn, 2000). These same techniques show promise with worked examples for computing education. There is recent empirical research on students explaining programs and algorithms (Margulieux, Morrison, Guzdial, & Catrambone, 2016; Sudol-DeLyser, Stehlik, & Carver, 2012), comparing algorithms (Patitsas, Craig, & Easterbrook, 2013), interacting with sub-goal labels (Margulieux, Catrambone, & Guzdial, 2016; Morrison, Margulieux, & Guzdial, 2015), and getting feedback via intelligent tutoring systems (Di Eugenio et al., 2015; Harsley & Morgan, 2015; Sudol-DeLyser et al., 2012) and electronic books (Ericson, Guzdial, & Morrison, 2015).

Best Practices from Mechanical Dissection, Tod Phod Jod, Worked Examples, CS Education

Mechanical dissection, Tod Phod Jod, and worked examples all guide students to deconstruct well-constructed examples. Students explain how the examples work, identify parts, learn terminology, and compare items. Participants of mechanical dissection and Tod Phod Jod get hands-on feedback, while students that interact with worked examples can get computerized feedback. They all help students learn exemplary design patterns and problem solving strategies. This prepares students both intellectually and psychologically to master design challenges and problem solving challenges.

A few additional learning-by-taking-apart techniques that are used for teaching programming are noteworthy. These emphasize code comprehension, with the philosophy that it is important that students understand code before, or as, they learn to write code on their own. This stands in contrast to the typical emphasis on code writing. There is evidence that students who are unable to understand code samples are unable to write similar ones accurately (Lister et al., 2004; Lopez, Whalley, Robbins, & Lister, 2008). Some researchers investigate the cognitive processes involved with code comprehension (Schulte, Clear, Taherkhani, Busjahn, & Paterson, 2010), while others experiment with curricula designed to promote code comprehension. A variety of curricular approaches engage students with analyzing and comparing well-written code samples (Astrachan & Reed, 1995; Deimel & Moffat, 1982; Kimura, 1979; Linn & Clancy, 1992). Some emphasize the importance of tracing code (Cutts, n.d.; Lopez et al., 2008) and understanding the invisible/notional machine of a programming language with the help of program visualization aids (du Boulay, O'Shea, & Monk, 1981; Sorva, 2013). One popular technique is to ask students to predict what code does. Another is to use completion problems; here students are given well-structured code and asked to complete missing sections of it (Deimel & Moffat, 1982; Van Merriënboer & Paas, 1990). Completion problems help bridge the gap between comprehending code and writing code. Thus, predicting and completing are useful activities for de-construction in addition to explaining, comparing, and labeling.

Learning From Intentional Errors and Bugs

Given that budding bicycle mechanics and fashion designers typically gain mastery by fixing and mending broken things, it follows that an educator could systematically present a collection of broken items to students to provide knowledge of common problems, and practice with fixing them. Neither mechanical dissection nor *Tod Phod Jod* take this approach, perhaps due to the logistical challenges of doing this with physical objects. This approach is used to some extent in math and CS education, through intentional errors and bugs. Here, the instructional designer intentionally places a carefully designed error or bug in an otherwise well-built example to highlight a key concept, common error, or common misconception.

Although people can learn from their own mistakes (Duncker, 1945), this is often serendipitous. Could learning be more comprehensive and efficient if instruction included intentional errors? This question sparks controversy. Behavioral psychologists view this approach as undesirable because it may introduce or reinforce misconceptions. Several math education researchers who use intentional errors view the behaviorist philosophy as normative, and characterize their own use of them as controversial (Isotani et al., 2011; Tsamir & Tirosh, 2003; Tsovaltzi et al., 2010). In contrast, several theories support the idea that intentional errors can promote learning. These include the theories of cognitive dissonance (Festinger, 1957, 1962), negative knowledge (Kaess & Zeaman, 1960; Oser, Nöpflin, Hofer, & Aerni, 2012), impasse-driven learning (VanLehn, 1988), learning from errors (Ohlsson, 1996), and overlapping waves (Siegler, 2002). Learning from intentional bugs may ease the pain of debugging one's own programs. Researchers of intentional errors hypothesize that they may ultimately reduce frustration and increase feelings of competency.

Learning Through Effective Practice

Most of us are familiar with the proverb *practice makes perfect*, but what does education research have to say about practice? The testing effect is a well-known phenomenon. Researchers have found that frequent quizzes provide retrieval practice that helps students learn, retain, and transfer knowledge. This is most successful if tests are distributed/spaced over time, if they engage learners in effortful recall rather than rote memorization, and if feedback is given (P. C. Brown, Roediger III, & McDaniel, 2014; Dunlosky, Rawson, Marsh, Nathan, & Willingham, 2013; Roediger & Butler, 2011).

Some education researchers view practice from a metacognitive perspective. Palincsar and Brown introduced the reciprocal teaching pedagogy, which guides teachers to explicitly teach metacognitive strategies and encourage students to practice these strategies (Palincsar & Brown, 1984). Singley and Anderson researched the role of practice in developing the ability to transfer knowledge from one context to another (Singley & Anderson, 1989). K. Anders Ericsson and colleagues introduced the idea of *deliberate practice* in response to longstanding beliefs that expertise is innate and that experts don't have to practice much. Considering domains such as chess, music, sports, and writing, they found that a lot of time spent on practice does not necessarily lead to progress. In order for practice to be effective it must have certain qualities: deliberate practice is focused, effortful, and individualized by a teacher or coach to address weaknesses and build strengths. Although deliberate practice is usually not enjoyable, people do it to master a domain (Ericsson & Charness, 1994).

Many computing teachers engage their students with *blocked practice*. For example, topic A is introduced and students practice topic A. Then topic B is introduced and students practice topic B, and so on. Education psychology researchers have found that distributing practice over time and interleaving topics is superior to blocked practice. It is also helpful to scaffold instruction from concrete to abstract, balance examples with problem solving, provide feedback, and consider students' prior knowledge (Dunlosky et al., 2013; Koedinger, Booth, & Klahr, 2013). Practice correlates positively with developing programming skills (Douce, Livingstone, & Orwell, 2005; Macnamara, Hambrick, & Oswald, 2014; Ventura Jr., 2005). The Leeds group recommends that teachers frequently give students practice problems to develop automaticity with basic programming skills (Lister et al., 2004). Kranch studied novice programmers and argues that they need plenty of practice with fundamentals (Kranch, 2011). Unfortunately, many beginning students say they lack the time and motivation to practice (Kinnunen & Malmi, 2006).

Research Questions

- How can a de-constructionist approach be realized in a programming class?
- What are students' attitudes about this approach?

Method

I conducted a design based research study. Design based research involves iterative experimentation in authentic settings such as classrooms; it typically involves a curricular intervention and cooperation with teachers (A. L. Brown, 1992; Collins, 1990). "Design experiments have both a pragmatic bent – 'engineering' particular forms of learning – and a theoretical orientation – developing domain-specific theories by systematically studying those forms of learning and the means of supporting them" (Cobb, Confrey, DiSessa, Lehrer, & Schauble, 2003, p. 9). I conducted a usability study, two pilot studies, and a semester-long experiment. Here I report on parts of the semester-long experiment, for which I designed web-based de-constructionist practice problems for learning to program with Python.

Context and Participants

The study was conducted in cooperation with the CS department of an urban public research university in the northeast United States. In 2015 the university had ~28.5k undergraduate students: 51% female; 11% Asian; 13% African American/Black; and 55% White. The study involved Professors Town and Park (pseudonyms). Each taught two sections of an introductory CS1-Python course with (N=87) undergraduate students overall. Both CS majors and non-majors attended the course. 15% of the students were female; 24% identified as a (non-Caucasian, non-Asian) racial or ethnic minority. I was the lab instructor and developed the activities for the weekly labs.

I developed a problems set for each lab starting with week 4, and iteratively designed problems for the following week for a total of ten labs. I used the Runestone Interactive e-book system to create web-based practice problems (Ericson et al., 2015; Miller & Ranum, 2014). Runestone has a unique collection of components that provide interactivity and real-time feedback. In addition to components for multiple choice and free response questions, several components help students explore code through interactive code reading. The *clickable* component (See Figure 2) allows the student to click on one or more sections of code to identify key features. The *code lens* component (See Figure 3) provides program visualization as the user steps through code line by line (Cross, Hendrix, & Barowski, 2011; Guo, 2013). Using the *drag 'n drop* component (See Figure 4), the user clicks and drags items from one column to match items in another. With the *Parsons problem* component, the user re-orders scrambled code (Parsons & Haden, 2006). Several components require students to write code. The author may supply starter code (to be completed or changed), or ask the student to write code from scratch. The author may also supply unit tests, to test the user's code and supply feedback. Practice problems related to debugging can be created with any of these components.



Figure 2. Clickable

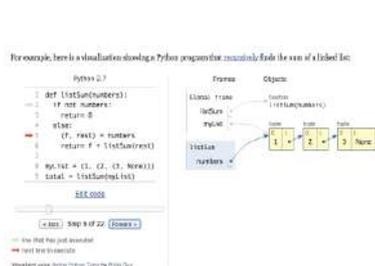


Figure 3. Code lens (Guo, 2013)



Figure 4. Drag 'n Drop

As a guide I used an instructional design model for de-construction (Griffin, 2016), which I refined during the experiment (See Figure 1). Following the model, the practice problems progressed from easier reading and tracing problems (that involved explaining, identifying, labelling, predicting, and comparing)

to harder ones that required writing code. Most problems involved short code segments. The problem sets incorporated distributed practice and interleaving. Students who finished before the lab period ended were free to leave. Students worked individually but could consult with classmates, with me, or my teaching assistant. A student who completed all ten labs interacted with 109 practice problems, ~11 per lab.

The treatment group (Bugs) and control group (NoBugs) were each comprised of two lab sections, one for each teacher. The Bugs group got 22 practice problems with bugs; the NoBugs group got 22 similar reading/tracing problems without bugs. Thus, ~80% of the problems were common; ~20% were specific to condition. Debugging problems were generally introduced as follows. For a given topic, problems with correct example code were introduced first. Next, easy problems with bugs were introduced, such as ones where the bug location was shown, which the student was asked to explain, categorize, or fix. Then more difficult problems were introduced, such as ones that required finding a bug. (These are not hard and fast rules.)

Timeline

The course spanned a 14-week semester, with two 80-min lectures and one 2-hour lab per week. Class time was devoted to lectures, lab time to practice problems. I met with the professors weekly. They taught the same topics with the same textbook and gave similar exams.

Instruments and Procedures

At the end of the semester an online survey was administered to gauge students' attitudes about the practice problems. It had the following Likert-scale questions:

- 1) Indicate your level of agreement with the following statement: The "Interactive Python" e-book exercises during lab helped me to learn Python (strongly disagree, disagree, neutral, agree, strongly agree).
- 2) In general, I found the e-book exercises to be: (too easy, about the right level of difficulty, too hard)
- 3) I recommend that in the future, the labs have: (fewer e-book exercises, about the same number, more)
- 4-5) Did you like the {reading/tracing, writing} exercises? (strong dislike, dislike, neutral, like, strong like)
- 6-7) Do you think the {reading/tracing, writing} exercises helped you learn Python? (not helpful, neutral, helpful, very helpful)

The Bugs group got two additional questions, about liking and learning from the debugging problems. Additional instruments to measure learning gains were administered; that data will be reported in a future publication. Throughout the experiment I recorded researcher memos and took field notes about students' interactions with the practice problems in the labs.

Results

This section summarizes the responses to the post attitudes survey.

Attitudes About Overall Learning, Difficulty, and Quantity. A majority of students responded positively about overall learning, difficulty, and quantity of practice problems. Of the (N=76) survey respondents, 92% agreed or strongly agreed that the practice problems helped them to learn Python. 88% thought the level of difficulty was about right; 8% found them too easy; 4% too hard. Regarding dosage, 62% recommended the same amount for future courses; 11% recommended fewer, 28% more.

Attitudes about Liking the Practice Problems. A majority of students liked or strongly liked each type of practice problem. (N=75) students responded to the question about how much they liked the reading/tracing problems. 83% liked or strongly liked them; 8% disliked them; 9% were neutral. Of the (N=76) students who responded to the question about liking the code writing problems. 87% liked or strongly liked them; 3% disliked them; 8% were neutral. (N=39) students in the Bugs group responded about liking the debugging problems. 69% liked or strongly liked them; 5% disliked them; 26% were neutral.

Attitudes About Learning From the Practice Problems. A majority of students thought both the reading/tracing and the writing problems helped them to learn, while 38% thought so about the debugging problems. (N=76) students answered the question about learning from the reading/tracing problems; 82% thought they were helpful or very helpful; 3% thought they were not helpful; 16% were neutral. About the code writing problems, 96% of (N=76) thought they were helpful or very helpful; 1% thought they were not helpful; 3% were neutral. For learning from debugging, an extra response category was added (Detrimental) because the review of the literature suggests that some think errors may deter learning. Of the (N=39) students in the Bugs group who responded, 38% thought they were helpful or very helpful; 15% thought they were not helpful; one student (3%) thought they were detrimental; 44% were neutral.

Discussion

This design-based research study implemented and further developed the emerging pedagogy of de-constructionism. In a 10-week intervention with a quasi-experimental design, undergraduates in a Python programming class solved sets of practice problems designed with a de-constructionist approach. This approach emphasizes ample practice with taking apart well-built examples, some of which have intentional bugs. Before writing code for a given topic, students engaged with interactive reading and tracing code by explaining, comparing, identifying, labelling, matching, tracing, and unscrambling. The treatment group got some problems with intentional bugs.

The study sought to understand students' attitudes about learning with a de-constructionist approach. According to the post attitudes survey, a majority of students liked each type of practice problem (for reading/tracing, writing, and debugging). While a majority of students thought the reading/tracing and writing problems were helpful for learning, only 38% thought so about the debugging problems. Further research is warranted to determine if it is useful to help students see the potential benefits of learning from intentional bugs, e.g. to help one become aware of common errors and learn how to repair them.

As the lab instructor, I observed that students' interactions with the practice problems were generally smooth; they appeared motivated to complete the problem sets. I was assured that all students who attended the lab interacted with the key course concepts. Students appreciated the immediate feedback and the chance to re-try until they completed a problem successfully. Approximately a third to half of the students finished during the first hour of the 2-hour lab period. Usually several took the entire period.

As the instructional designer, my experience with Runestone Interactive was generally positive. Despite being a fledgling system, it is reliable and responsive, and has a rich variety of components. Creating the practice problems was somewhat laborious. It required encoding them with the reStructuredText language, but Runestone is evolving to streamline this process. I found that implementing distributed practice and interleaved practice was more difficult and time consuming than implementing blocked practice.

As the researcher, in some respects it was beneficial for me to also have the roles of lab instructor and instructional designer. This was an early-stage, exploratory study that implemented the Model for De-Construction for the first time (See Figure 1). The lab exercises were iteratively designed weekly based on observations from the previous week. In future studies, having two or three people perform these roles would be advantageous.

Future plans include incorporating constructionist activities into the lab period. Rather than having students leave when they finish the problem sets, students could work on creative projects. Ideally the rubrics would be such that students who need more time to master the practice problems are not penalized for having less time to work on creative projects. This approach is common with the *mastery learning* approach (Bloom, 1968; Griffin, Pirmann, & Gray, 2016). Unfortunately, many CS professors are unaccustomed to designing rubrics for creative projects, such as the professors in this study who have doctorates in mathematics and computer science. Introducing constructionism into such learning environments would be valuable for both teachers and students. Students could have motivating design experiences, and teachers could learn how to design, manage, and evaluate creative projects.

Additional future plans include adding appealing graphics, and gamification elements. A priority would be to add these elements to debugging and reading/tracing activities, since students already view writing code as a worthwhile activity. Having students collaborate on de-constructionist activities, e.g. with Pair Programming, is worth exploring. Also desirable would be to differentiate instruction based on prior experience and/or demonstrated mastery.

The constructionist/de-constructionist dialectic is useful for teachers and instructional designers but it is not all-inclusive. Other pedagogies are also important, such as ones that teach problem solving or systematic approaches to debugging. De-constructionism draws inspiration from both reverse engineering and mathematics education. While this study explored de-constructionism as it applies to learning programming, it can be implemented for a variety of subjects with both physical and conceptual elements.

References

- Agogino, A. M., Sheppard, S., & Oladipupo, A. (1992). Making Connections to Engineering During the First Two Years. In *Frontiers in Education, Proceedings of the Twenty-Second Annual Conference (FIE '92)* (pp. 563–569). IEEE.
- AP Computer Science Principles. (n.d.). Retrieved January 3, 2018, from <https://apcentral.collegeboard.org/courses/ap-computer-science-principles>
- Astrachan, O., & Reed, D. (1995). AAA and CS 1: The Applied Apprenticeship Approach to CS 1. In *Papers of the 26th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '95)* (pp. 1–5). ACM.
- Bloom, B. S. (1968). Learning for Mastery. *UCLA Evaluation Comment*, 1(2).
- Booth, J. L., McGinn, K. M., Young, L. K., & Barbieri, C. (2015). Simple Practice Doesn't Always Make Perfect: Evidence From the Worked Example Effect. *Policy Insights from the Behavioral and Brain Sciences*, 2(1), 24–32.
- Boychev, P. (2015). Constructionism and Deconstructionism. In *Constructivist Foundations* (Vol. 10, pp. 355–369).
- Bransford, J. D., Brown, A. L., Cocking, R. R., Donovan, M. S., & Pellegrino, J. W. (Eds.). (2000). *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. Washington DC: National Academy Press.
- Brown, A. L. (1992). Design Experiments: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings. *Journal of the Learning Sciences*, 2(2), 141–178.
- Brown, P. C., Roediger III, H. L., & McDaniel, M. A. (2014). *Make it Stick: The Science of Successful Learning*. Cambridge MA; London England: The Belknap Press of Harvard University Press.
- Catrambone, R. (1998). The Subgoal Learning Model: Creating Better Examples So That Students Can Solve Novel Problems. *Journal of Experimental Psychology: General*, 127(4), 355–376.
- Chi, M. T. H., De Leeuw, N., Chiu, M.-H., & Lavancher, C. (1994). Eliciting Self-Explanations Improves Understanding. *Cognitive Science*, 18(3), 439–477.
- Chikofsky, E. J., & Cross, J. H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. In *IEEE Software* (Vol. 7, pp. 13–17).
- Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Schauble, L. (2003). Design Experiments in Educational Research. *Educational Researcher*, 32(1), 9–13.
- Collins, A. (1990). Towards a Design Science of Education, Technical Report No. 1. Center for Technology in Education. (pp. 1–9). New York NY: Center for Technology in Education.
- Conati, C., & VanLehn, K. (2000). Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education*, 11(1), 389–415.

- Cross, J. H., Hendrix, T. D., & Barowski, L. A. (2011). Combining Dynamic Program Viewing and Testing in Early Computing Courses. In *Computer Software and Applications Conference IEEE 35th Annual* (pp. 184–192). IEEE.
- Cutts, Q. (n.d.). Quintin Cutts - Teaching Programming: Too much doing, not enough understanding. Retrieved March 13, 2018, from <https://www.youtube.com/watch?v=Pim4aYfiZiY>
- Deimel, L. E., & Moffat, D. V. (1982). A More Analytical Approach to Teaching the Introductory Programming Course. In J. Smith & M. Schuster (Eds.), *Proceedings of the NECC* (pp. 114–118). Columbia: The University of Missouri.
- Di Eugenio, B., Green, N., Alzoubi, O., Alizadeh, M., Harsley, R., & Fossati, D. (2015). Worked-out Examples in a Computer Science Intelligent Tutoring System. In *Proceedings of the 16th Annual Conference on Information Technology Education* (pp. 121–121). ACM.
- Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: a review. *ACM Journal of Educational Resources in Computing*, 5(3), 1–13.
- du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14, 237–249.
- Duncker, K. (1945). On problem-solving. *Psychological Monographs*, 58(5).
- Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving Students' Learning With Effective Learning Techniques: Promising Directions From Cognitive and Educational Psychology. *Psychological Science in the Public Interest*, 14(1), 4–58.
- Ericson, B. J., Guzdial, M. J., & Morrison, B. B. (2015). Analysis of Interactive Features Designed to Enhance Learning in an Ebook. In *Proceedings of the 11th International Conference on Computing Education Research (ICER '15)* (pp. 169–178). ACM.
- Ericsson, K. A., & Charness, N. (1994). Expert performance: Its structure and acquisition. *American Psychologist*, 49(8), 725–747.
- Festinger, L. (1957). *A theory of cognitive dissonance*. Stanford, CA: Stanford University Press.
- Festinger, L. (1962). *A theory of cognitive dissonance (Vol. 2)*. Stanford CA: Stanford University Press.
- Griffin, J. M. (2016). Learning by Taking Apart: Deconstructing Code by Reading, Tracing, and Debugging. In *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16)* (pp. 148–153). ACM.
- Griffin, J. M., Pirmann, T., & Gray, B. (2016). Two Teachers, Two Perspectives on CS Principles. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE '16)* (pp. 461–466). ACM.
- Guo, P. J. (2013). Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 579–584). ACM.
- Guzdial, M. (2003). A Media Computation Course for Non-Majors. In *Proceedings of the 8th annual conference on Innovation and Technology in Computer Science Education - ITiCSE '03*.
- Harsley, R., & Morgan, S. (2015). Learning Together: Expanding the One-To-One ITS Model for Computer Science Education, 9–10.
- Isotani, S., Adams, D., Mayer, R. E., Durkin, K., Rittle-Johnson, B., & McLaren, B. M. (2011). Can erroneous examples help middle-school students learn decimals? *Proceedings of the Sixth European Conference on Technology Enhanced Learning: Towards Ubiquitous Learning (EC-TEL-2011)*, 1–14.
- Kaess, W., & Zeaman, D. (1960). POSITIVE AND NEGATIVE KNOWLEDGE OF RESULTS ON A PRESSEY-TYPE PUNCHBOARD. *Journal of Educational Psychology*, 60(1), 12–17.
- Kalyuga, S. (2007). Expertise Reversal Effect and Its Implications for Learner-Tailored Instruction. *Educational Psychology Review*, 19(4), 509–539.

- Kick, R., & Trees, F. P. (2015). AP CS Principles: Engaging, Challenging, and Rewarding. *ACM Inroads*, 6(1), 42–45.
- Kimura, T. (1979). Reading before Composition. In *Proceedings of the 10th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '79* (pp. 162–166).
- Kinnunen, P., & Malmi, L. (2006). Why Students Drop Out CS1 Course? In *Proceedings of the second International workshop on Computing Education Research - ICER '06* (pp. 97–108). ACM.
- Koedinger, K. R., Booth, J. L., & Klahr, D. (2013). Instructional Complexity and the Science to Constrain It. *Science*, 342(6161), 935–937.
- Kranch, D. A. (2011). Teaching the Novice Programmer: A Study of Instructional Sequences and Perception. *Education and Information Technologies*, 17(3), 291–313.
- Linn, M. C., & Clancy, M. J. (1992). The Case for Case Studies of Programming Problems. *Communications of the ACM*, 35(3), 121–132.
- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... Thomas, L. (2004). A Multi-National Study of Reading and Tracing Skills in Novice Programmers. *ACM SIGCSE Bulletin*, 36(4), 119.
- Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships Between Reading, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the 4th International Workshop on Computing Education Research (ICER '08)* (pp. 101–112). ACM.
- Macnamara, B. N., Hambrick, D. Z., & Oswald, F. L. (2014). Deliberate practice and performance in music, games, sports, education, and professions: A meta-analysis. *Psychological Science*, 25(8), 1608–1618.
- Malan, D. J., & Leitner, H. H. (2007). Scratch for Budding Computer Scientists. *ACM SIGCSE Bulletin*, 39(1), 223–227.
- Margulieux, L. E., Catrambone, R., & Guzdial, M. (2016). Employing subgoals in computer programming education. *Computer Science Education*, 26(1), 44–67.
- Margulieux, L. E., Morrison, B. B., Guzdial, M., & Catrambone, R. (2016). Training Learners to Self-Explain: Designing Instructions and Examples to Improve Problem Solving. In *Proceedings of International Conference of the Learning Sciences, ICLS* (Vol. 1, pp. 98–105).
- Miller, B., & Ranum, D. (2014). Runestone Interactive: Tools for Creating Interactive Course Materials. In *Proceedings of the First ACM Conference on Learning @ Scale (L@S '14)* (pp. 213–214). ACM.
- Mobile CSP. (n.d.). Retrieved May 1, 2015, from <http://mobile-csp.org>
- Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. In *Proceedings of the 11th International Conference on Computing Education Research (ICER '15)* (pp. 21–29). ACM.
- Murer, M., Fuchsberger, V., & Tscheligi, M. (2017). Un-Crafting: De-Constructive Engagements with Interactive Artifacts. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '17)* (pp. 67–77). ACM.
- Ohlsson, S. (1996). Learning from error and the design of task environments. *International Journal of Educational Research*, 25(5), 419–448.
- Oser, F., Näpflin, C., Hofer, C., & Aerni, P. (2012). Towards a theory of Negative Knowledge (NK). Almost-mistakes as drivers of episodic memory amplification. *Professional and Practice-Based Learning*, 6, 53–70.
- Palincsar, A. S., & Brown, A. L. (1984). Reciprocal Teaching of Comprehension Monitoring Activities. *Cognition and Instruction*, 1, 117–175.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc.

- Papert, S. (1987). *Constructionism: A New Opportunity for Elementary Science Education (NSF Award Abstract #8751190)*. Retrieved from https://nsf.gov/awardsearch/showAward?AWD_ID=8751190
- Parsons, D., & Haden, P. (2006). Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In *Proceedings of the 8th Australasian Conference on Computing Education* (Vol. 52). Australian Computer Society, Inc.
- Patitsas, E., Craig, M., & Easterbrook, S. (2013). Comparing and Contrasting Different Algorithms Leads to Increased Student Learning. *Proceedings of the 9th International Conference on Computing Education Research (ICER '13)*, 145–152.
- Rashtriya Avishkar Abhiyan. (n.d.). Retrieved September 21, 2017, from http://mhrd.gov.in/sites/upload_files/mhrd/files/raa/Order_of_RAA_Guidelines.pdf
- Regan, M., & Sheppard, S. (1996). Interactive Multimedia Courseware and the Hands-On Learning Experience: An Assessment Study. *Journal of Engineering Education*, 85(2), 123–132.
- Rittle-Johnson, B., & Star, J. R. (2007). Does comparing solution methods facilitate conceptual and procedural knowledge? An experimental study on learning to solve equations. *Journal of Educational Psychology*, 99(3), 561–574.
- Roediger, H. L., & Butler, A. C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Sciences*, 15(1), 20–27.
- Roschelle, J., & Linde, C. (1996). *Toy Dissection Formative In-Depth Assessment Report*. Institute for Research on Learning (IRL), Palo Alto, CA.
- Schulte, C., Clear, T., Taherkhani, A., Busjahn, T., & Paterson, J. H. (2010). An introduction to program comprehension for computer science educators. In *Proceedings of the 2010 ITiCSE working group reports on Working group reports - ITiCSE-WGR '10* (p. 65).
- Seabrook, J. (2010, September). How to Make It. *The New Yorker*. Retrieved from <http://www.newyorker.com/magazine/2010/09/20/how-to-make-it>
- Self, J. (1997). From constructionism to deconstructionism: anticipating trends in educational styles. *European Journal of Engineering Education*, 22(3), 295–307.
- Sheppard, S. D. (1992). Mechanical Dissection: An Experience in How Things Work. In *Proceedings of the Engineering Education Conference: Curriculum Innovation & Integration* (pp. 6–10).
- Siegler, R. S. (2002). Microgenetic Studies On Self-Explanation. In N. Granott & J. Parziale (Eds.), *Microdevelopment: Transition Processes in Development and Learning* (pp. 31–58). Cambridge University Press.
- Singley, M. K., & Anderson, J. R. (1989). *The Transfer of Cognitive Skill*. Cambridge, MA: Harvard University Press.
- Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13(2), 1–31.
- Sudol-DeLyser, L. A., Stehlik, M., & Carver, S. (2012). Code comprehension problems as learning events. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '12*, 81.
- Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, 12(2), 257–285.
- Sweller, J. (2006). The worked example effect and human cognition. *Learning and Instruction*, 16(2), 165–169.
- Sweller, J., & Cooper, G. A. (1985). The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. *Cognition and Instruction*, 2(1), 59–89.
- The Beauty and Joy of Computing. (n.d.). Retrieved August 15, 2015, from <http://bjc.berkeley.edu/>

- Tod Fod Jod: To Encourage Students to Discover, Experiment, Innovate. (2012, May 30). Office of Adviser to the Prime Minister on Public Information Infrastructure and Innovations. Retrieved from <https://www.slideshare.net/pmpiii/tod-fod-jod>
- Tsamir, P., & Tirosh, D. (2003). In-service mathematics teachers' views of errors in the classroom. In *International Symposium: Elementary Mathematics Teaching*. Prague.
- Tsovaltzi, D., Melis, E., McLaren, B. M., Meyer, A., Dietrich, M., & Gogvadze, G. (2010). Learning from Erroneous Examples: When and How do Students Benefit from them? In *Proceedings of the European Conference on Technology Enhanced Learning, LNCS (vol. 6383)*. Heidelberg: Springer.
- Van Merriënboer, J., & Paas, F. G. W. C. (1990). Automation and Schema Acquisition in Learning Elementary Computer Programming: Implications for the Design of Practice. *Computers in Human Behavior*, 6(3), 273–289.
- VanLehn, K. (1988). Towards a Theory of impasse-driven Learning. *Learning Issues for Intelligent Tutoring Systems*, 19–41.
- Ventura Jr., P. R. (2005). Identifying Predictors of Success for an Objects-First CS1. *Computer Science Education*, 15(3), 223–243.
- Vishnoi, A. (2012). *NIC lesson on learning: Tod-Fod-Jod*. *The Indian Express*. New Delhi India. Retrieved from <http://archive.indianexpress.com/news/nic-lesson-on-learning-todfodjod/1015042/0>
- Wood, W. H., & Agogino, A. M. (1996). Engineering Courseware Content and Delivery: The NEEDS Infrastructure for Distance-Independent Education. *Journal of the American Society for Information Science*, 47(11), 863–869.
- Wu, C. (2008). Some Disassembly Required. In *ASEE Prism* (Vol. 18, pp. 56–59). American Society for Engineering Education.

Mind the Gap: Teaching High School Students about Wealth Inequality through Agent-based Participatory Simulations

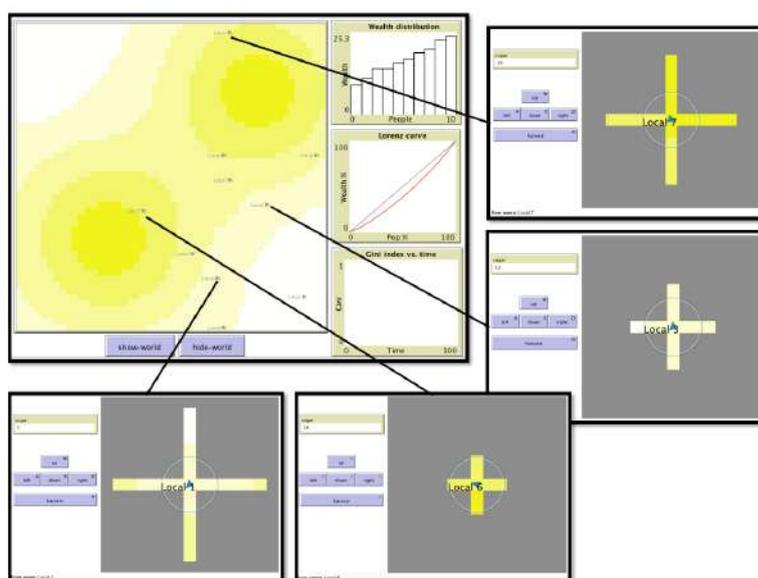
Yu Guo, yuguo2012@u.northwestern.edu
Northwestern University, USA

Uri Wilensky, uri@northwestern.edu
Northwestern University, USA

Abstract

This research paper presents a design-based research study centring on a constructionist curricular unit, called Mind the Gap (MTG), which was designed to help high school students learn about a complex and controversial social issue in the United States—wealth inequality. The four-day-long unit was implemented in eight economics classes with a total of more than 200 students across two high schools with vastly different demographics. In both schools, students' engagement with the unit was high. Preliminary data analysis has shown that students 1) made connections between their simulation experience and the real world to reason about wealth inequality, and 2) showed attitudinal changes favouring more equality after the unit.

MTG revolves around a series of three participatory simulations, which are microworlds that allow students to project themselves into through their own avatars like in a multiplayer online game and interact with the virtual environments in order to “figure out” the rules embedded in these simulations. This work contributes to both the literature of designing constructionist learning environment and people's perception and understanding of wealth inequality. In democratic countries, people's understanding of inequality is the key to achieve more equal societies.



Mind the Gap ABPS. Upper left: teacher's view; the other four: students' view

Keywords

agent-based participatory simulation; wealth inequality; economics curriculum; complex social phenomena; artificial society

Introduction

Today, American people face many controversial social issues that deeply divide the country. Examples include wealth inequality, racial segregation, and climate change. Although natural and social scientists have been studying these problems and have provided accumulating evidence for causes and consequences, when people think about these problems, they tend to form very polarized views based on the vastly different local environments they are in (e.g., Xu & Garand 2010), political beliefs (e.g., Kteily, Sheehy-Skeffington, & Ho, 2017), and media coverage (e.g., Diermeier et al. 2017). The scientific methods and evidence, which can help people understand the mechanisms and consequences of many social issues, remain underutilized in people's reasoning about complex social problems.

Through the lens of complexity science, wealth inequality, racial segregation, and climate change can be seen as patterns that emerge from interactions among constituting elements. The system level patterns at very large scale can be strikingly different from the behavior of the constituting elements at the local level, making these complex systems unintuitive and hard for people to understand (Wilensky & Resnick, 1999; Chi, 2005; Penner, 2000; 2001). Agent-based modeling (ABM) is a computational modeling method that scientists use to study these complex systems. With the ABM approach, elements of complex systems are modeled as autonomous agents, which have their own properties and behaviors. In an ABM, numerous such agents interact with each other and give rise to complex patterns at the system's level.

ABM is also an effective way to help students understand complex systems. Extensive research has been done on using NetLogo (Wilensky, 1999)—an agent-based modeling tool that dynamically simulate and visualize complex systems—to create instructional interventions in STEM education. Evidence show that these curricula supported students' understanding of complex phenomena in a variety of scientific subjects, including biology, chemistry, and physics (e.g., Wilensky & Reisman, 2006; Levy & Wilensky, 2008; Sengupta & Wilensky, 2009).

Teaching about complex social phenomena is an understudied area (Hjorth & Krist, 2016). While social phenomena such as wealth inequality can be treated as complex systems, learning about these phenomena presents new challenges. Hjorth & Wilensky (2014) designed and taught urban planning and policy reasoning with an agent-based models and found that compared with physical or biological systems with constituting elements like molecules or animals, social systems—composed of people with wants and needs—are much more heterogeneous and therefore more difficult for students to reason about (Hjorth & Krist, 2016).

A special form of ABM—participatory simulation—can be especially promising for helping students understand complex social issues. Agents' behavior in regular ABMs are pre-programmed, but agents in agent-based participatory simulations (ABPS) are controlled directly by students, who project themselves in to the models by taking on the roles of the agents and acting out their behaviors. For example, the Disease participatory simulation (Wilensky & Stroup, 1999) models the spread of a contagious disease across a population. In this simulation, each student uses a computer to control an avatar's movements in a shared virtual world. When an infected avatar touches a healthy avatar, there is a certain chance that the healthy one also becomes infected. The simulation starts with nobody being infected, but when the teacher randomly chooses a student to infect, the simulation usually turns into a chasing game, in which the infected student tries to infect as many other students as possible. Once others are infected, they in turn chase those who are not infected. This participatory simulation teaches an important complex systems idea about logistic growth: no matter what the students do at the individual level, the larger pattern of infection across the population remains the same—the epidemic starts with a very slow spreading speed, but as it develops, it reaches a breakout point, after which the number of infected people skyrockets, and finally the progress levels off, when the whole population is infected. Such an approach not only teaches students' the dynamics of complex systems, but also brings out students' emotions, desires, and decisions, which can be productive elements that contribute to understanding complex social issues.

ABPS as Constructionist learning environments

Constructionist learning environments are designed to support students' constructing knowledge and artifacts (Papert, 1980). Core features of these learning environments include leveraging students' prior knowledge—tapping into students existing experience, closeness to objects (Papert & Harel, 1991)—bringing student close to objects that they can manipulate, and epistemological pluralism (Turkle & Papert, 1992)—valuing different ways of knowing.

Microworlds are a genre of constructionist learning environments in which students explore a very specific aspect of the world. In this sense, a microworld is a model of the real or a hypothetical world, in which the rules and regularities that govern the world are beneath the surface as far as the students' direct experience is concerned, but the rules and regularities manifest themselves by transforming the states of concrete objects as students interact with the microworld (Groen & Kieran 1983). Edwards (1995) citing Piorlli & Greeno (1988) and Pratt (1991) describes a fundamental aspect of microworlds as “the scientific or mathematical phenomenon which the designer intends to introduce to the learner is instantiated or embodied in computer code. It is by translating mathematical or scientific regularities into procedures and computational objects that the designer constructs a microworld, and this process involves a complex series of choices and design decisions.” (Edwards 1995)

ABPS can be seen as a special type of microworlds, in which models can be executed by participants' following rules in the microworlds and acting out the behavior of elements that they are representing (Colella 2000). Students explore these powerful models by becoming part of them. Students construct specific runs, or instances, of the model by collectively acting out the rules. These runs become artifacts that can be replayed and investigated. In addition, in ABPS, students co-construct theories to explain the phenomena that they are part of. The explanation is grounded in students' individual experience of being part of the phenomena and interacting with others in the ABPS. Students construct publicly sharable entities and pay collective attention to these entities, which can either be projected on the big screen for the whole class to see, or exist ephemerally in the shared virtual space, such as the configuration of everybody's avatar in that space.

Mind the Gap Curricular Unit

We designed an ABPS curricular unit, called “Mind the Gap” (Guo & Wilensky, 2018a), for students to learn about wealth inequality—a complex and controversial socioeconomic phenomenon.

Previous studies have shown that most people have misperceptions and misconceptions about wealth inequality, such as severely underestimating the extent of wealth inequality in the U.S. and attributing the problem solely to individual characteristics, such as work ethic (e.g., Bullock 2008; Hauser & Norton 2017). Most existing instructional interventions on wealth inequality use multicultural or social justice approaches to engage students in knowing factual knowledge, analyzing their social environments, critically thinking about diversity, and advocating for possibilities for marginalized groups (Nagda, Gurin, & Lopez, 2003; Seider 2011; Mistry, Brown, Chow, & Collins, 2012). These interventions are important because they raise students' awareness of social issues, empower marginalized students, and help students avoid deficit thinking. However, students still have difficulties identifying or explaining the structural problems of wealth inequality after going through these curricula (e.g., Seider, 2011; Mistry et al., 2012).

Building on a body of work on participatory simulations of social complex phenomena (e.g., Sterman 1992; Wilensky, 2002; Maroulis & Wilensky, 2004), we use ABPSs to teach the complex and controversial socioeconomic topic of wealth inequality. Participatory simulations in STEM education are shown to promote engagement (e.g., Colella, 2000; Klopfer, Yoon, & Perry, 2005), improve understanding of complex systems mechanisms (e.g., Wilensky & Stroup, 1999; 2002; Stroup & Wilensky, 2014; Berland & Wilensky, 2015), and high classroom participation (e.g., Fies & Langman, 2011). However, less is known about the affordances of participatory simulations in social science topics.

Mind the Gap (MTG) centers on a series of three ABPSs—microworlds in which students' avatars interact with the computer-based virtual environments that represent simplified economies. Instead of

encompassing all the factors and intricate relations that may contribute to wealth inequality in the real world, these microworlds highlight three mechanisms of the phenomenon: 1) emergence, 2) randomness, and 3) feedback loops. These ABPSs serve as computational laboratories of social systems for students to play the roles of people with different socioeconomic status, explore the rules and the structures of the system, experience success and frustration, and make sense of the emergence of complex patterns from individuals' behavior.

The design of the three ABPSs are based on SugarScape agent-based models (Epstein & Axtell, 1996; Li & Wilensky, 2009a, 2009b, 2009c) that allow computational scientist to investigate complex social phenomena such as wealth inequality, migration, trade, and epidemics through a bottom up approach—agents' following very simple rules. It is surprising for people to see a small set of simple rules, when followed by each agent, can generate highly complex patterns at the system level, which are similar to real-world social phenomena.

The original SugarScape models represent a society with two major parts: 1) the “land”, where resources (sugar) can be harvested to become people's wealth; 2) the people, which are computational agents that are pre-programmed to find the highest concentration of sugar around them and to “harvest” that sugar. The land is represented by a 51 by 51 checkerboard. Each tile on the checkerboard contains certain amount of sugar. The people have randomly assigned traits and are randomly placed on the checkerboard.

Our design of MTG ABPSs preserved the basic structures of the SugarScape models, such as the 51 by 51 checkerboard, the random placement of agents, the random assignments of agent traits, and most of the rules. However, in our ABPS models, agents are avatars that are directly controlled by students, who can make decisions of what to do next within the constraints of certain rules. Students are connected to the virtual space through the HubNet architecture—a client-server technology designed for participatory simulations (Wilensky & Stroup, 1999b). Therefore, in MTG ABPSs, students not only explore the simulations, they *become* parts of the simulation.

Figure 1 shows the teacher's view of the MTG Equality model. At the center of the view is the checkerboard. The grid is added to this figure to visualize the checkerboard. In the real model, the grid is not visible to either the teacher or the students. The unicolor of yellow shows that each tile contains the same amount of sugar. However, in the other two models, the shades of yellow can differ from tile to tile, representing different concentration of sugar—the darker the color, the higher the sugar. Students are not supposed to know about the resource distribution. The teacher can hide the checkerboard from the students by clicking the “hide-world” button, which makes the whole checkerboard grey and students' avatars invisible. After playing the simulation, in the discussion phase, the teacher can use the “show-world” button to show students what kind of world they were in.

The three plots on the right—a bar graph, a Lorenz curve, and a Gini index plot—show three frequently used forms of representation of wealth inequality in economics. The plots automatically update based on real time aggregation of the amount of sugar that students own. The “setup” and “go” buttons at the upper left corner prepares the model and runs the model. The “sugar-mean” monitor shows the average of sugar that students own, and the sugar distribution plot shows a histogram of tile sugar on the checkerboard.

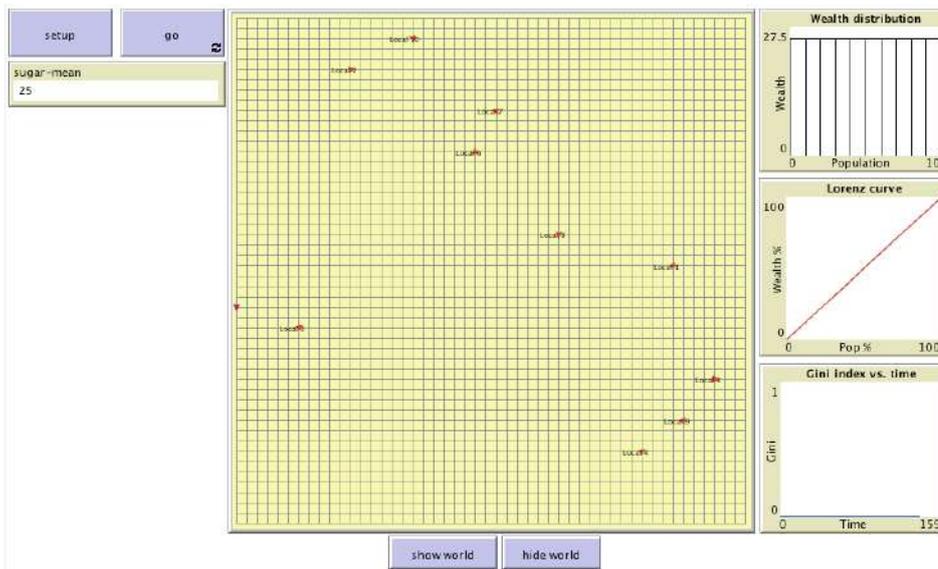


Figure 1. MTG Equal Opportunity model teacher's view (grid added to visualize the checkerboard)

Students see a different view from the teacher (Figure 2). The red arrowhead at the centre of the view is a student's avatar. Because the avatar has only imperfect local knowledge about its surroundings, it can only see a few tiles away in each direction. The yellow cross represents the field of view of the avatar.

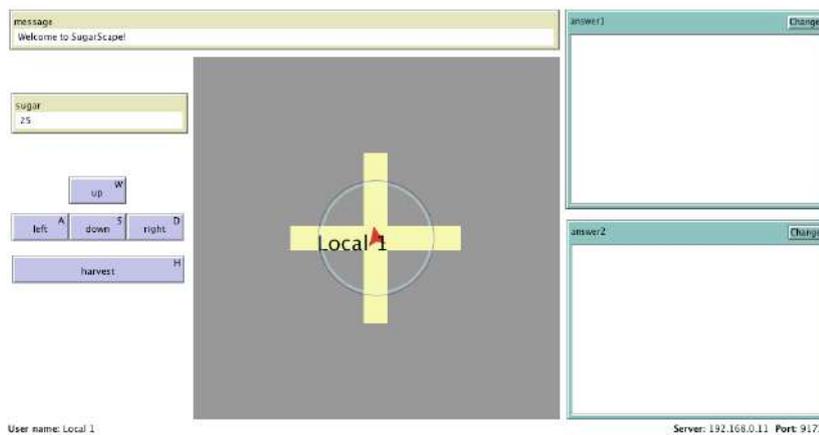


Figure 2. Students' View

A Student can use the “up”, “down”, “left”, and “right” buttons on the left to move their avatar around and explore the surroundings. He or she can also use the “harvest” button to collect sugar from the tiles they are on. The “sugar” monitor above the buttons show the current amount of sugar that the avatar owns. The “message” bar at the top shows real time tips and warnings about the student's behaviour. The two big input boxes on the right are data collection tools, accepting students' typed answers to two open-ended questions posted by the teacher during class.

Students have a few attributes:

- Vision: how many steps (tiles) away a student can see.
- Endowment: how many units of sugar a student starts with
- Metabolism: how many units of sugar is needed for moving one step or doing one harvest

Students have some actions they can take:

- Move: by clicking the direction buttons or the keyboard shortcuts, students can move around. Each click moves the student by one step and burns metabolism amount of sugar.
- Harvest: by clicking the harvest button, students harvest all the sugar on the tile that he or she is standing on. One harvest burns metabolism amount of sugar.

MTG curricular unit consists a series of three models: 1) Equal Opportunity, 2) Random Assignment, and 3) Feedback Loop (with education as an example). Each subsequent model builds on the previous one with added complexity that more closely resembles the real world (Figure 3). At the beginning of the unit, students are asked an overarching question that drives their inquiry: Why are rich people rich while poor people poor?

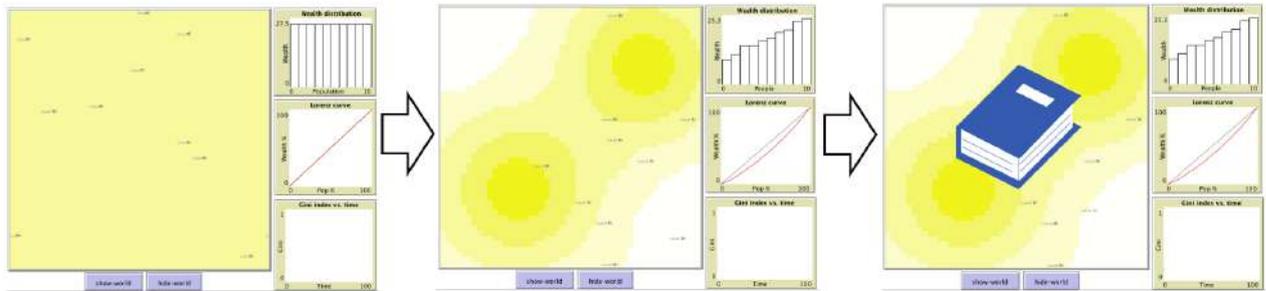


Figure 3. A series of three MTG models.

Left: Equal Opportunity model; middle: Random Assignment model; right: Feedback Loop model

The Equal Opportunity model (Guo & Wilensky, 2018b) allows students to experience that even everybody starts with equal opportunities, inequality can still emerge due to individual differences. In Equal Opportunity, sugar is evenly distributed, as shown by the unicolour yellow. Students are randomly placed on the checkerboard with the same personal traits, including equal vision, endowment, and metabolism. Students strive to become the richest in the class by harvesting as much sugar as possible. Inequality inevitably emerge due to differences in students' strategies, understanding of rules, and motivation. To answer the question about why rich people are rich while poor people are poor in this simulation, given that everyone starts with equal opportunity, it is fair to say that personal differences, such as intelligence, efforts, and work ethics determine wealth status.

The Random Assignment model (Guo & Wilensky, 2018c) allows students to experience the power of an uncontrollable force that contributes to wealth inequality. In this model, sugar is unevenly distributed across the checkerboard. Students are randomly placed on the board and are also randomly assigned different visions, metabolisms, and endowments. Therefore, the initial conditions that a student starts with to a large extent determine the student's course of life in this simulation. Students should realize that when faced with the force of randomness, the draw of luck, instead of personal abilities, usually shapes the course of life.

The Feedback Loop model (Guo & Wilensky, 2018d) allows students to experience another type of strong force that shapes people's course of life. Feedback loops, usually called virtuous circles or vicious circles, are systematized or institutionalized forces. Unlike randomness, which is not biased against anyone, feedback loops are usually socially constructed, privileging certain groups of people at the cost of oppressing other groups. This model uses education as an example to let students experience that depending on the cost of education, it can become a force that either closing or widening the gap between the rich and the poor. This model gives students the opportunity to "go to school" by pressing a button. While going to school has benefits, such as boosting earning per harvest by 130% and expanding vision by one step, it also has monetary and opportunity costs. When education is less expensive, all students can make use of it to improve their vision and earning. However, when education is expensive, it becomes a virtuous circle for the rich and a vicious circle for the poor. As the result, the rich become richer and the poor become poor, closely reflecting a crucial inequality issue in the real world.

For each model, the teacher facilitates the students to go through a five-step inquiry cycle: introduction, 2) question, 3) simulation, 4) reflection, and 5) connection (See Figure 4).

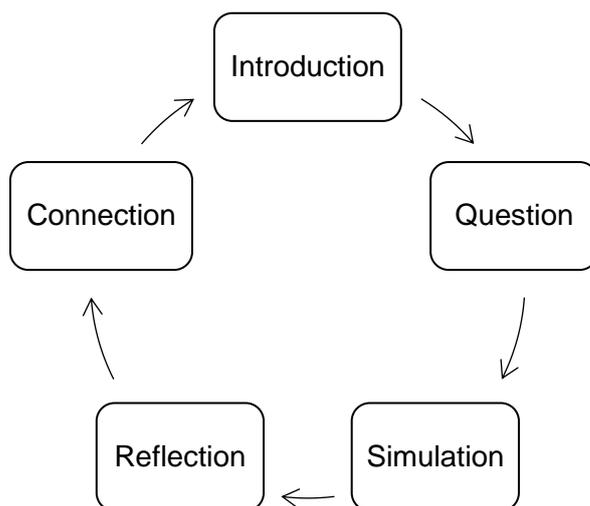


Figure 4. A Five-Step Inquiry Cycle.

In Step 1, students are introduced to the interface and rules of the model. In Step 2, students are asked to think about what makes rich people rich and poor people poor in that specific model. In Step 3, students play the simulation and compete to become the wealthiest person. In Step 4, whole class discussions are held for students to reflect on their experience in the model: who was the richest in this round? How did s/he achieve it? What was his or her starting condition? Who was the poorest? Why did s/he end up being poor? The richest and the poorest students are asked to share their stories in the simulation with the whole class. In Step 5, group discussions and whole class discussions are held for students to make connections between their simulation experience and the real-world: What are the real-world analogies of vision, endowment, metabolism, and the color of the starting tile? What is a real-world version story of the richest student? What is a real-world version story of the poorest student?

Students go through these five steps of inquiry cycle in each activity and keep revisiting the question about why the rich are rich while the poor are poor in different models. The increased complexity in each activity is expected to contribute to students' more sophisticated thinking about the question.

Method

The central goal of this design-based research study (Cobb, Confrey, diSessa, Lehrer, & Schauble, 2003) to investigate high school students' learning processes and outcomes in the MTG unit.

Research questions

With this curriculum design, we ask two research questions:

- How do students connect their simulation experience with their real-world experience?
- Does the unit improve students' understanding of wealth inequality and change their attitude toward it?

Participants

We implemented this curricular unit in eight high school economics classes with a total of more than 200 students across two schools with vastly different demographics. So far, we only finished preliminary analysis on the data from one school with two participating classes, so in this paper, we only report the implementation, analysis, and findings based on this corpus of data.

The two classes are in a public high school located in a highly ethnically diverse suburb of a large Midwestern city in the U.S. Students in these two classes are mostly sophomores and seniors, who are

taught by the same teacher for their regular economics curriculum. All 51 students were invited to participate in this study. Fifty (50) students consented to participate. Forty-four (44) completed the demographic survey: There are 19 females and 25 males. Twenty-one (21) self-identified as Asian, 14 White, 4 biracial (2 Asian and White, 1 Black and White, and 1 unspecified), 1 African American, and 1 Hawaiian or other pacific islanders. 13 students speak a language that is different from English at home (including Assyrian, Bangla, Cantonese, Gujarati, Romanian, Serbia, Farsi, Tagalog, Telugu, and Urdu). A majority of 32 students identify with the Democratic Party, 3 with the Republican Party, 4 with Other parties, and 6 with None.

Implementation

The implementation of the MTG unit spanned four days with 42 minutes of class time on each day. The first 20 minutes of the first day and the last 15 minutes of the last day were used for students to take the pre- and post- questionnaires. The first author taught the unit following the 5-step inquiry cycle described in the section above.

Data collection

Multiple types of data were collected. The pre- and post- questionnaires collect data on students' knowledge and attitudes toward wealth inequality before and after the unit. During the unit, students' typed responses to questions about each simulation are collected through input boxes built in the MTG models. Computer log data of all students' interactions with the models (including key strokes and performance) were collected. Whole class video recordings captured the class dynamics throughout the four days. A total of 20 students (10 from each class) were selected in consultation with their economics teacher as focal students to represent the racial and gender diversity. Two additional types of data were collected from the focal students: 1) Pre- and post- clinical interviews, in which researchers probed students' thoughts and attitudes toward wealth inequality through structured and follow-up questions. 2) Screen recordings: Focal students' computer screens were recorded using Camtasia screen grabbing software, which records on-screen behaviour and conversations between students.

Analysis

We analysed the pre- and post- questionnaires for evidence of attitudinal change towards wealth inequality before and after the unit. In both pre- and post- questionnaires, students were asked to rate a series of statements about wealth inequality on Likert scales. Questions include "On a scale of 1-7, with 1 being extremely unfair and 7 being extremely fair, please rate": 1) How fair is the wealth distribution in the U.S.?, 2) "The rich deserve their economic status", 3) "The poor deserve their economic status", 4) "In America, anyone can advance, regardless of their family of origin, economic status, or ethnicity", 5) "It is possible to move from poverty to affluence thorough hard work", and 6) "Poverty is a sign of personal failure". Students' ratings were summarized with two histograms, one for the pre-, one for the post. Because 44 students completed the pre-test and 43 the post-test, the histograms were normalized (converted to percentages) for easier comparison. Summary statistics were generated to describe the mean and the standard deviation of the pre- and post- results. The pre- and post- means were then compared.

In addition, we conducted interaction analysis (Jordan & Henderson, 1995) based on the whole class video data. The first author watched the videos to identify interesting moments during students' class discussions, in which students utilized resources from both their experience gained from the simulations and their real-world experience. These interesting moments were then transcribed. Below, we provide an excerpt of class video transcripts that illustrate these moments.

Findings

Changing attitudes

Overall, students' ratings shifted toward a more pro-equality attitude in the post questionnaire. More students rated wealth distribution in the U.S. as less fair and showed stronger disagreement toward the subsequent statements. We calculated and compared the means of pre- and post- ratings based on the

assumption that the 1-7 scale is an even continuum, and the same rating shows the same degree of agreement for all students.

Figure 5 shows the comparison of pre- and post- rating distribution of the fairness questions.

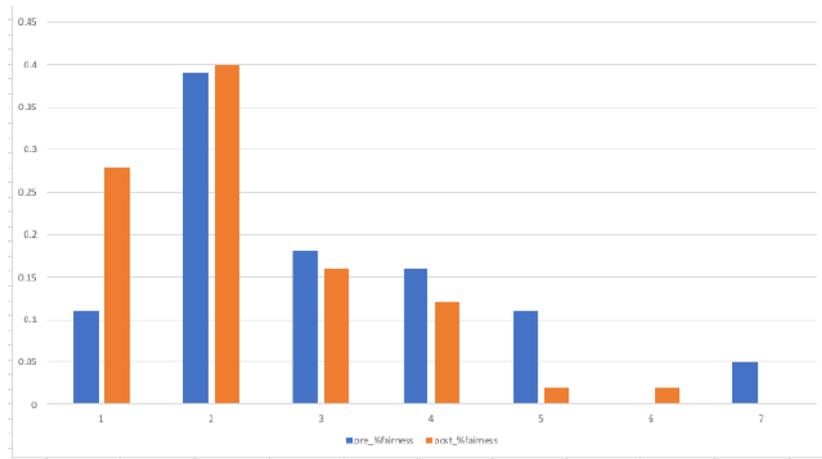


Figure 5. Pre- and post- comparison of students' rating distribution of the fairness question.

No.	Statement	pre	post	shift
1	fairness	2.95	2.28	0.67
2	The rich deserve	4.25	3.74	0.51
3	The poor deserve	2.66	2.56	0.1
4	Anyone can advance	3.8	3.4	0.4
5	Get rich by hard work	4.93	4.47	0.46
6	Poverty is failure	2.2	2.02	0.18

Table 1. Attitude shifts on all six Likert scale questions.

In Figure 5, the left bars (blue) show students pre- ratings and the right bars (orange) show post ratings. Although most students came to the unit already believing that wealth distribution in the U.S. was somewhat unfair (an average rating of 2.95), after the unit, students believed it even more strongly (post mean = 2.28). The average rating changed by 0.67 towards left, which means students think the wealth distribution is more unfair. The biggest change occurred in the lowest rating category (extremely unfair). Before the unit, only 11% (5 students) thought the distribution was extremely unfair. After the unit, the percentage increased to 28% (12 students).

Shifts in attitudes toward all statements followed a similar pattern, as shown in Table 1. These shifts are evidence that after the unit, students had more realistic perceptions of the fairness (Q1) and the opportunity structures in the U.S. (Q4 and Q5) based on understanding of mechanisms of wealth distribution and exposure to real-world data during the unit. Minimum changes were observed in negative statements about poor people (0.1 in Q3 and 0.18 in Q6). Students' pre-unit ratings of these questions were already very low (2.66 and 2.2, respectively). The minimum changes may have less to do with understanding the mechanisms that students learned from the unit, but have more to do with empathy that the students already have toward the poor.

Making connections between simulation experience and the real world

Below we present an excerpt of whole class discussion about the real-world meaning of students' experience in the MTG Randomness model. Kate was one of the richest persons for this round. The

teacher asked her to share her experience in the simulation and asked the class to find a real-world story that matches Kate's simulation experience (all names are pseudonyms).

Teacher: Can You share your story?

Kate: So I was kind of very near the dark yellow, so I just moved over and a started harvesting.

Teacher: Ok, so basically you spawned and you saw dark yellow right next to you. You moved over and started harvesting, right? And what was your vision?

Kate: One.

Teacher: One? OK, Interesting! Very interesting.

...

Teacher: Now let's get into groups and talk about: Is there a real version of their stories in the real world. let's think about her story. She started right next to the darkest yellow. She just went over and started harvesting.

[Students discussed in groups of three for 2 minutes]

Teacher: Which group wants to share their story of the rich in the real world? Go ahead.

Jack: We kind of thought of like really specific, like Sam Walton's kids have a bunch of money coming from Walmart and so as long as they don't mess up and move away from the wealth they have, they can keep that wealth and build on it and like how Kate just moved over one spot to build on the wealth that she already had.

Teacher: You guys agree? That's pretty much true right? It is like a kind of the inheritance thing, right? So, if your parents are super wealthy, they will give you a lot of resources. Just make sure you don't do too stupid things to move away from them. Make use of it, then you're probably pretty well off, right? Ok, yeah, I think that's a great story. Any other things to add? Any other things about the rich story?

Amy: Not about the rich, but sort of in general, I think that it's interesting that even if she was in a very good neighbourhood, but she had very limited vision. I think that was like kind interesting, because I think that's almost less realistic rather than more realistic, because I think if the simulations were more like real life, if you have a high vision, you are more likely to be in a very yellow area and have a low metabolism. it wouldn't be randomly assigned.

Teacher: I think this is a great point. In [MTG Randomness model], everything is random. It's possible that you are born in a very rich place but with a very limited vision. In the real world, do you think that happens that often? Not really, right? Because if you were born in a very resourceful place, you probably get better education. Your parents are probably well-educated, and you probably know a lot of people. Your vision is actually a lot better than a lot of other people. I think that's a great point. Thank you for bringing it up.

This excerpt shows that students' engagement with the simulation is quite sophisticated. Jack and his group understood that spawning near a dark yellow area in the simulation is analogous to being born to a rich family in the real world. It is easier for those who were born rich to build on the wealth they inherited to obtain more wealth. The MTG Randomness simulation helped students relate randomness in the simulation and in the world that can make life easier just by chance. However, Amy had deeper thoughts about the analogy between the simulation and the real world: she pointed out that random assignment in the model was not realistic, because things are correlated in the real world. If the simulation were more realistic, then "if you have a high vision, you are more likely to be in a very yellow area and have a low metabolism", which can form systematic, rather than random biases that work for or against people.

In this case, the less realistic feature of the simulation is not a drawback of the design. Instead, it is precisely the nature of a microworld and a computational model, which foreground certain aspects of the system being modelled to highlight its basic mechanisms. The MTG Randomness model highlights randomness as a strong mechanism that contributes to inequality. The students were able to identify its

limits and provide real world mechanisms in contrast. The discussion triggered by comparing the simulation with the real world is a valuable learning opportunity about the mechanisms of wealth inequality and the scientific practice of modelling for the whole class.

Conclusion and Discussion

Mind the Gap agent-based participatory simulation curricular unit is our attempt to address people's understanding and attitude toward complex and controversial social issues through designing constructionist learning environments. The implementation of this unit in eight high school economics classes showed students' high engagement with this type of design. Preliminary evidence has shown that students' attitude shifted toward a more pro-equality stance after learning the unit. Students also showed sophisticated thinking about rules and conditions in the simulation and in the real world in their discussions when making connections between the virtual and the real world.

Wealth inequality is an example of a myriad of social issues that people face in today's society. This pioneer work shows what learning pathways and learning outcomes are possible to achieve with this type of ABPS constructionist learning environment. We will continue data analysis to discover more evidence of students' knowledge and attitude change. With a better understanding of the learning affordances of ABPS learning environments, we can use it to teach other complex and controversial social topics, such as racial segregation, climate change, and gun control. Equipped with better understanding of mechanisms of these social issues, people in democracies can exercise their rights more responsively to achieve more equal societies for everyone.

References

- Berland, M., & Wilensky, U. (2015). Comparing Virtual and Physical Robotics Environments for Supporting Complex Systems and Computational Thinking. *Journal of Science Education and Technology*, 24(5), 628–647. <https://doi.org/10.1007/s10956-015-9552-x>
- Bullock, H. E. (2008). Justifying inequality: A social psychological analysis of beliefs about poverty and the poor. In L. Ann, & D. Harris (Eds). *The colors of poverty: Why racial and ethnic disparities persist*. Russell Sage Foundation: New York, NY.
- Chi, M. T. (2005). Commonsense conceptions of emergent processes: Why some misconceptions are robust. *The Journal of the Learning Sciences*, 14(2), 161–199.
- Cobb, P., Confrey, J., diSessa, A., Lehrer, R., Schauble, L., & others. (2003). Design experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- Colella, V. (2000). Participatory simulations: Building collaborative understanding through immersive dynamic modeling. *The Journal of the Learning Sciences*, 9(4), 471–500.
- Diermeier, M., Goecke, H., Niehues, J., & Thomas, T. (2017). Impact of inequality-related media coverage on the concerns of the citizens. DICE Discussion Paper.
- Edwards, L. D. (1995). Microworlds as representations. In *Computers and exploratory learning* (pp. 127–154). Springer.
- Epstein, J. M. & Axtell, R. L. (1996). *Growing artificial societies: social science from the bottom up*. Brookings Institution Press.
- Fies, C., & Langman, J. (2011). Bridging worlds: Measuring learners' discursive practice in a partsim supported biology lesson. *International Journal of Science and Mathematics Education*, 9(6), 1415–1438.
- Groen, G., & Kieran, C. (1983). In search of Piagetian mathematics. *The Development of Mathematical Thinking*, 351–375.
- Guo, Y. & Wilensky, U. (2018a). Mind the Gap curriculum. <http://ccl.northwestern.edu/MindtheGap/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

- Guo, Y. & Wilensky, U. (2018b). NetLogo MTG 1 Equal Opportunities HubNet model. <http://ccl.northwestern.edu/netlogo/models/MTG1EqualOpportunitiesHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Guo, Y. & Wilensky, U. (2018c). NetLogo MTG 2 Random Assignment HubNet model. <http://ccl.northwestern.edu/netlogo/models/MTG2RandomAssignmentHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Guo, Y. & Wilensky, U. (2018d). NetLogo MTG 3 Feedback Loop HubNet model. <http://ccl.northwestern.edu/netlogo/models/MTG3FeedbackLoopHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Hauser, O. P., & Norton, M. I. (2017). (Mis)perceptions of inequality. *Current Opinion in Psychology*, 18, 21–25.
- Hjorth, A., & Krist, C. (2016). Unpacking Social Factors in Mechanistic Reasoning (Or, Why a Wealthy Person is Not Exactly Like a Grey Squirrel). Singapore: International Society of the Learning Sciences.
- Hjorth, A., & Wilensky, U. (2014). Redesigning Your City—A Constructionist Environment for Urban Planning Education. *Informatics in Education—An International Journal*, (Vol13_2), 197-208. Chicago
- Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences*, 4(1), 39–103.
- Klopfer, E., Yoon, S., & Perry, J. (2005). Using palm technology in participatory simulations of complex systems: A new take on ubiquitous and accessible mobile computing. *Journal of Science Education and Technology*, 14(3), 285–297.
- Li, J. and Wilensky, U. (2009a). NetLogo Sugarscape 1 Immediate Growback model. <http://ccl.northwestern.edu/netlogo/models/Sugarscape1ImmediateGrowback>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Li, J. and Wilensky, U. (2009b). NetLogo Sugarscape 2 Constant Growback model. <http://ccl.northwestern.edu/netlogo/models/Sugarscape2ConstantGrowback>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Li, J. and Wilensky, U. (2009c). NetLogo Sugarscape 3 Wealth Distribution model. <http://ccl.northwestern.edu/netlogo/models/Sugarscape3WealthDistribution>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Kteily, N. S., Sheehy-Skeffington, J., & Ho, A. K. (2017). Hierarchy in the eye of the beholder:(Anti-)egalitarianism shapes perceived levels of social inequality. *Journal of Personality and Social Psychology*, 112(1), 136.
- Levy, S. T., & Wilensky, U. (2008). Inventing a “mid level” to make ends meet: Reasoning between the levels of complexity. *Cognition and Instruction*, 26(1), 1–47.
- Maroulis, S. and Wilensky, U. (2004). NetLogo Oil Cartel HubNet model. <http://ccl.northwestern.edu/netlogo/models/OilCartelHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Mistry, R. S., Brown, C. S., Chow, K. A., & Collins, G. S. (2012). Increasing the Complexity of Young Adolescents’ Beliefs About Poverty and Inequality: Results of an 8th Grade Social Studies Curriculum Intervention. *Journal of Youth and Adolescence*, 41(6), 704–716.
- Nagda, B., Gurin, P., & Lopez, G. E. (2003). Transformative pedagogy for democracy and social justice. *Race, Ethnicity and Education*, 6, 165-191.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 1–11.
- Penner, D. E. (2000). Explaining systems: Investigating middle school students’ understanding of emergent phenomena. *Journal of Research in Science Teaching*, 37(8), 784–806.

- Penner, D. E. (2001). Complexity, emergence, and synthetic models in science education. *Designing for Science*, 177–208.
- Pirolli, Peter, & Greeno, James. (1988). The problems space of instructional design. In J. Psofka, D. Massey, & S. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (pp. 181-201). Hillsdale, NJ: Erlbaum.
- Pratt, David. (1991). The design of Logo microworlds. In L. Nevile (Ed.), *Proceedings of the Fifth International Logo and Mathematics Education Conference* (pp. 2541). Cairns, Australia.
- Seider, S. (2011). The Role of Privilege as Identity in Adolescents' Beliefs About Homelessness, Opportunity, and Inequality. *Youth & Society*, 43(1), 333–364.
- Sengupta, P., & Wilensky, U. (2009). Learning Electricity with NIELS: Thinking with Electrons and Thinking in Levels. *International Journal of Computers for Mathematical Learning*, 14(1), 21–50.
- Sterman, J. (1992). Teaching Takes Off: Flight Simulators for Management Education-"The Beer Game". Retrieved from <http://web.mit.edu/jsterman/www/SDG/beergame.html>
- Stroup, W. M., & Wilensky, U. (2014). On the Embedded Complementarity of Agent-Based and Aggregate Reasoning in Students' Developing Understanding of Dynamic Systems. *Technology, Knowledge and Learning*, 19(1–2), 19–52.
- Turkle, S., & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior*, 11(1), 3–33.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- Wilensky, U. (2002). NetLogo HubNet Tragedy of the Commons HubNet model. <http://ccl.northwestern.edu/netlogo/models/HubNetTragedyoftheCommonsHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep or a firefly: Learning biology through constructing and testing computational theories - An embodied modeling approach. *Cognition & Instruction*, 24(2), 171-209.
- Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*, 8(1), 3–19.
- Wilensky, U. and Stroup, W. (1999). NetLogo HubNet Disease HubNet model. <http://ccl.northwestern.edu/netlogo/models/HubNetDiseaseHubNet>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U. & Stroup, W., (1999b). HubNet. <http://ccl.northwestern.edu/netlogo/hubnet.html>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- Wilensky, U., & Stroup, W. (2002). *Participatory Simulations: Envisioning the networked classroom as a way to support systems learning for all*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA, April 13.
- Xu, P., & Garand, J. C. (2010). Economic context and Americans' perceptions of income inequality. *Social Science Quarterly*, 91(5), 1220–1241.

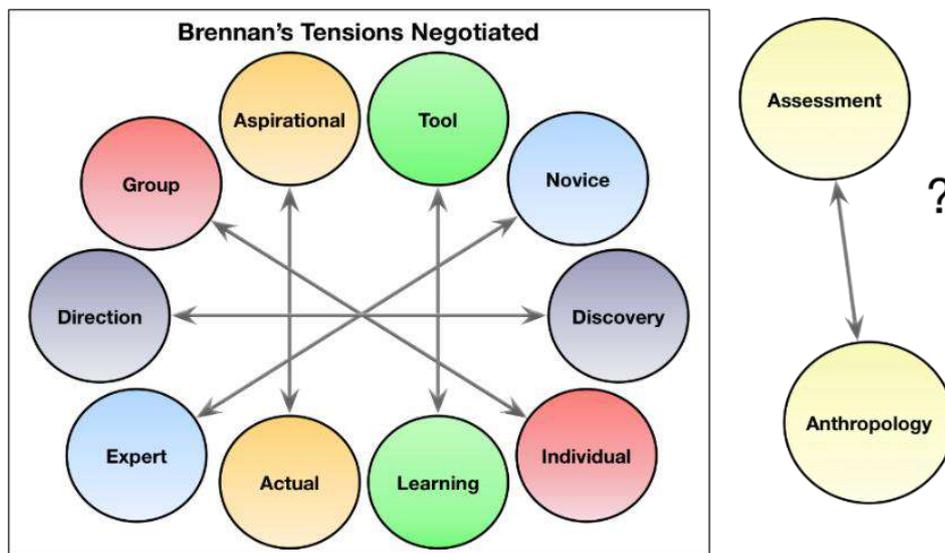
To Assess or Not to Assess: Tensions Negotiated in Six Years of Teaching Teachers about Computational Thinking

Daniel Hickmott, Daniel.Hickmott@uon.edu
School of Education, The University of Newcastle, Australia

Elena Prieto-Rodriguez, Elena.Prieto@newcastle.edu.au
School of Education, The University of Newcastle, Australia

Abstract

Coding and computational thinking have recently become compulsory skills in many school curricula around the world. This presents a major challenge for teachers, as many of them would not have had formal education on the underlying concepts needed to apply these skills, or the pedagogies to teach them. A range of professional development opportunities are currently being offered to teachers to address this challenge. One notable example of a model of PD designed to include constructionist learning experiences is *ScratchEd*, which was designed by Karen Brennan. Upon reflecting on her experiences designing and running *ScratchEd* for seven years, Brennan identified five tensions that professional development providers may need to negotiate, and proposed that these tensions could be used for scrutinising and critiquing professional development.



The five 'tensions negotiated' by Karen Brennan and our additional tension

In this paper, we analyse the process we have followed in the past six years to design, evaluate and improve our professional development through the lens of Brennan's 'tensions negotiated'. While we have experienced the same tensions, we contend that the extent to which we assess teachers' learning is a new tension that extends those identified by Brennan. There are strong reasons to assess teachers' knowledge, however, quantitative measures of learning could be at odds with Constructionism: as Papert argued in *Mindstorms*, constructionist educators should study their learning environments as anthropologists. Consequently, we have called this new tension the *tension between anthropology and assessment*.

Keywords

teacher professional development; constructionism; computational thinking; programming; pedagogical content knowledge.

Introduction

Recently, while working with a colleague to develop a joint research project on teaching quality, he asserted, 'just add coding or computational thinking somewhere, it's the only way to get funds these days'. This colleague is a successful academic in the field of physical education and has what one could consider an impressive track record with funding applications. While this comment is not entirely reflective of the funding situation in the field of educational research, and admittedly was pronounced in jest, it provides an interesting insight into how prominent coding has become for education authorities worldwide. Many countries have recently introduced, or are planning to introduce, curricula that include the teaching of Coding and Computational Thinking throughout K-12 (Webb et al., 2016). Australia has recently introduced a Digital Technologies learning area within its new national curriculum (Falkner, Vivian, Falkner, & Williams, 2017), England introduced a Computing Curriculum in 2014 that is mandatory for all K-6 students (Sentance & Csizmadia, 2016), and in the United States there have been national efforts to introduce K-12 Computer Science education into all of the states (Brown & Briggs, 2015; Fisher, 2016).

However, Coding and Computational Thinking (C&CT) are not new to schools and there have been many attempts to bring these skills into mainstream K-12 education since the 1970s. Many of these efforts were led by *constructionists*, who encouraged students coding in Logo, and similar programming languages, to explore "powerful ideas", as Papert (1980, p. 138) had envisioned. The teaching of coding in Logo did not become widespread in K-12 education in the 1980s and 1990s, due to a complex mix of social, political and technical issues, and a "lack of subject-matter integration" (Agalianos, Whitty, & Noss, 2006; Kafai & Burke, 2013, p. 61). Papert (1987, p. 24) lamented that many researchers and educators had taken a "technocentric" view of Logo and that they had ignored many of the powerful ideas from *Mindstorms* (Papert, 2000).

The renewed interest in C&CT has motivated educators and researchers to work towards fulfilling Papert's dream (Resnick, 2017), and although Seymour has recently passed away, his legacy lives on in the 21st Century. His ideas have had a large influence on the design of Scratch, which is widely available, free and commonly used in schools (Kafai & Burke, 2013). The Maker Movement, which encourages learners to construct digital and physical artefacts that are personally meaningful to them, has also been influenced by Papert's *Constructionism* (Blikstein, 2013). Despite the wide availability of free tools for learning C&CT, there are still some major challenges that educators and researchers are facing when introducing C&CT in K-12. One of the main challenges is the preparation of teachers, as C&CT is unlikely to have been part of their K-12 or tertiary education (Falkner et al., 2017; Yadav, Sands, Good, & Lishinki, 2018).

To overcome the challenges that the introduction of C&CT in school curricula presents for teachers, many professional development (PD) initiatives are being developed and implemented worldwide (Garneli, Giannakos, & Chorianopoulos, 2015). These initiatives focus on different aspects of C&CT and typically provide a range of experiences and knowledge for teachers to take to their classrooms. While there is general consensus that content knowledge (CK) is essential in order to effectively teach, there is also general acceptance of the importance of pedagogical content knowledge (PCK), as introduced by Lee Shulman (1986). PCK is widely used in the education literature as a means of understanding the particular types of knowledge unique to teachers. This type of knowledge includes, for example, knowledge of student misconceptions about specific topics and how teachers might respond to these and knowledge of effective analogies for illustrating concepts or ideas. PCK has been studied in the context of C&CT both in primary and secondary school settings (Angeli et al., 2016). However, while PCK is recognised as being of crucial importance for effective teaching in C&CT (Saeli, Perrenet, Jochems, & Zwaneveld, 2011), research about ways to understand and promote PCK in C&CT is still in its infancy (Cooper, Grover, Guzdial, & Simon, 2014). There are still many lessons to learn about providing appropriate and effective PD to pre-service and in-service teachers (Guzdial, 2015; Yadav et al., 2018).

Interestingly, in the special issue on 'Constructionism and Creativity' of the journal *Constructivist Foundations*, Karen Brennan reflected, "*I am often asked 'What lessons have you learned from your [PD] work?' I have to appreciate that my experiences and understandings are more aptly described as*

'tensions negotiated' than 'lessons learned'" (Brennan, 2015, p. 293). In her article, she describes the tensions she has encountered when running the *ScratchEd* model of PD. Brennan (2015) provides examples of when these tensions have occurred in each of the PD formats and explains the approaches she has used to negotiate these tensions.

Our experiences running PD over the last six years have taught us what we thought were invaluable lessons about what effective PD is and how certain aspects of the PD can be evaluated. However, it has become that apparent our lessons learned are much more like the tensions negotiated by Brennan (2015), and that our latest lesson learned is that we have not really learned any lessons - we have just discovered ways to negotiate tensions.

In this paper, we analyse the process we followed in the past six years to design, evaluate and improve our PD through the lens of the tensions negotiated by Brennan (2015). Our analysis highlights the importance of evaluating PD and developing metrics that can help us evaluate teachers' learning during and after the PD. As we have designed the most recent PD workshops with *Constructionism* as a "framework for action" (DiSessa & Cobb, 2004, p. 83), we often find that it is difficult to decide between measuring the changes of teachers' PCK and CK, and not measuring these changes. We contend this difficulty is a new tension that differs from those identified by Brennan (2015), which we call the *tension between anthropology and assessment*.

Professional Development Design

As stated in the previous section, many PD initiatives have been established around the world to address the challenge of preparing educators for teaching C&CT in K-12 (Menekse, 2015). These include Massive Open Online Courses (Falkner et al., 2017), face-to-face workshops (Menekse, 2015), and the development of local hubs that provide teachers with support from peers, known as 'Master teachers' (Sentance, Humphreys, & Dorling, 2014).

One notable example of a model of PD that has been designed to include constructionist learning experiences is *ScratchEd* (Brennan, 2015). Initially, Brennan (2015) developed an online community for educators to share their experiences teaching with Scratch. After the establishment of the online community, Brennan developed Scratch educator meetups that are run for three hours on a Saturday morning each month in Boston. Brennan (2015, p. 293) stated that the online community "cannot provide constructionist experiences", and, consequently, the meetups were intended to provide these experiences. Brennan also developed an online workshop titled the Creative Computing Online Workshop (CCOW), which was available to teachers globally. The CCOW lasted six weeks and involved a variety of activities, including the development and sharing of design journals. The *ScratchEd* model combined these three formats of PD: the online community, monthly face-to-face meetups and the CCOW.

We have been developing and running face-to-face PD workshops at our university for six years. The main aim of the workshops has been to help prepare teachers for teaching the Australian Digital Technologies curriculum. In 2013 and 2014, the workshops were only available for High School teachers, but in 2015 we began to include Primary School teachers. These workshops have involved a variety of sessions: activities with step-by-step instructions, collaborative problem-solving exercises and lesson-planning activities, and presentations by academics and industry representatives.



Figure 1. Participants in 2017 workshop

The design and implementation of the workshops has evolved each year as a result of participants' feedback, which has been collected through validated surveys. As discussed later in the *Tensions Negotiated* section, the feedback that we received in the surveys, particularly the feedback in responses to the surveys' open-ended questions, has influenced the changes that we have made to the PD. For example, the feedback from our first workshop in 2013 indicated that there was too much theory and lectures in the sessions, which led us to begin the inclusion of more hands-on and constructionist sessions in the PD in the years that followed. We have also been informed by the general PD literature, in which there has been extensive research about what factors are present in effective PD (for example, Desimone, 2009). Unlike Brennan (2015), we have not had any online components in our PD yet, and we would argue that face-to-face PD will always be valuable and essential. The availability of face-to-face PD is particularly important for teachers who are only beginning to learn C&CT and have low-confidence about teaching them, as argued by Sentance and Csizmadia (2017).

Tensions Negotiated

Six years after beginning the *ScratchEd* project, Brennan (2015) reflected on her experiences and observations when running the different forms of PD and also analysed interviews with 30 of the teachers that participated in her PD. In her analysis, Brennan (2015, p. 294) concluded that her experiences and understandings would be best described as "tensions negotiated", rather than "lessons learned" and described the five tensions that she considered to be the most pressing. She also explained how these tensions were experienced and negotiated in the different formats of the *ScratchEd* model.

Like Brennan (2015), we have sought to provide constructionist learning experiences in our PD, as we believe that learning environments that are influenced by constructionist ideas should be encouraged in K-12. However, the format of the PD that we have run over the last six years has been very different to the forms of PD in the *ScratchEd* model. Our PD has mainly been run as two-day workshops that have been planned weeks or months beforehand and have not had an online component like the *ScratchEd* online community or CCOW. Like Brennan, when we have reflected on our experiences about running the workshops, we have often thought about the lessons that we have learned that could be useful for other PD providers. But, also like Brennan, we have found *tensions negotiated* to be a more appropriate description than *lessons learned*. In the conclusion of her article, Brennan argues that the tensions she identified are not specific to Scratch or her *ScratchEd* model, and that they could be scrutinized or critiqued by other PD providers.

In the following sections, we explain how we have experienced each of the five tensions identified by Brennan when developing and running our PD. To determine how the PD was impacted by the different tensions, we initially analysed the open-ended responses to the feedback surveys (n = 137) through the lens of the five tensions identified by Brennan (2015), by thematic coding the responses in NVivo 11. We also reflected on our experiences during the PD and discussed how these related to each of the different tensions. We found that teachers have been satisfied with our workshops, and some aspects have improved as a consequence of responding to feedback. However, we found that this feedback provided only insight on teachers' general satisfaction with our workshops and, to a certain extent, the classroom applicability of the PD content, but was not enough to give us a full picture of their learning.

Also, our funding bodies have started to require measures of impact that go beyond self-reported self-efficacy outcomes. Consequently, we determined that we are experiencing a tension that was not identified by Brennan, which we have called the *tension between anthropology and assessment*, after the argument made by Papert (1980) that educators should study their learning environments as anthropologists.

Tension between tool and learning

The *tension between tool and learning* refers to balancing the PD's focus between teaching about concepts/tools (CK) and helping teachers create learning environments for students to use these concepts/tools (PCK). On one side of the tension is a focus on teaching a tool, such as Scratch, and the essential concepts needed to use and understand that tool. On the other side of the tension are the pedagogical practices and different approaches to classroom activities, such as the "creative design activities" described by Brennan (2015, p. 294).

In our workshops, we have conducted sessions with a variety of software and hardware tools and have usually focused on imparting CK. The rationale for including a variety of tools has been to make teachers aware of the many options that they have for teaching C&CT at different levels of K-12. Each year, the teachers that have attended the workshops have taught at a variety of levels and subjects and, consequently, we always try to provide a variety of options of activities that are relevant to them.

We have often focussed on imparting CK in the majority of the workshop sessions because many teachers have not learned about C&CT during their K-12 or tertiary education and much of this CK is likely to be new to them (Falkner et al., 2017; Yadav et al., 2018). Consequently, we expected that the workshops participants would not have much CK and that they primarily needed to acquire CK before they acquire PCK. Although the workshop sessions always involve the teaching of certain tools, we tend to have central concepts that we aim to impart in the sessions and theme the sessions accordingly. For example, we titled the session that we introduce Scratch in, "Visual Programming with Scratch". During that session we explained what visual programming is, gave examples of how visual programming can be included in K-12, and described visual programming tools that could be used instead of Scratch.

Despite the workshop sessions having had a concept as a central theme, we have found that teachers usually responded to questions about applying the concepts learned during the workshops with answers about introducing the tools to their students, rather than introducing the concepts or particular pedagogical approaches in their classes. For example, in response to the question "*Do you think you will you apply what you learned in the workshop? If so, what?*", many teachers responded with a list of tools, such as, "*Yes, makey makeys, scratch at a deeper level than I've been teaching it and sphero*". Comments such as this led us to include a combination of sessions that focus on a particular concept and/or tool, and sessions that focus on particular pedagogical approaches. For example, in the 2017 Primary School workshop we introduced C&CT concepts in Scratch in a session on the first day and then ran a session on the second day that focussed on teaching CT through design activities with the Creative Computing Curriculum guide (Brennan, Balch, & Chung, 2014), which has activities that use Scratch.

Another difficulty encountered during our PD, which Brennan (2015) does not explicitly mention, and that we consider to be encompassed in the *tension between tool and learning*, is the technical issues we experienced when using certain tools. For example, we encountered technical issues when using the MIT AppInventor software with Android tablets during the "Building Mobile Apps" activity, due to the way the network was configured at our university. Although we resolved these issues quickly during the session, this troubleshooting detracted from the activity. Teachers may encounter these issues themselves with their classes, so it could be argued that it is beneficial to include instruction on how to troubleshoot these issues in PD. Despite the difficulties faced, we believe that the tools that can be difficult to use in some environments, are conducive to constructionist learning experiences. Consequently, we intend to include more instructional time that addresses the potential technical issues, so that teachers can troubleshoot these issues themselves and assist their students to troubleshoot these issues.

Tension between direction and discovery

The *tension between direction and discovery* refers to the balancing by instructors to provide guidance and resources to learners, while also allowing learners to discover resources and concepts on their own. This is similar to the “play paradox” defined by Noss and Hoyles (1996), which they used to refer to the balancing of exploration and guidance when students are learning in a microworld (for example, Logo).

When we design our PD, there are certain learning outcomes that we aim to address in each of the sessions, which align with concepts from the Australian Digital Technologies curriculum. These concepts include what Brennan and Resnick (2012) call *computational concepts*, which include concepts like sequencing and loops. Additionally, Primary School teachers have also been encouraged to integrate C&CT across different subject areas (NSW Education Standards Authority, 2018), and to teach C&CT to assist development of students’ “general capabilities”, for example, Literacy, Numeracy and Creative Thinking (ACARA, 2018). Our past PD workshops have been focussed on upskilling teachers’ CK and consequently many of the sessions’ planned learning outcomes have been related to the essential *computational concepts* from the Australian Digital Technologies curriculum.

Brennan (2015, p. 293) designed the *ScratchEd* PD model with the main assumption that “teachers should have learning experiences that are comparable to their students’ learning experiences”. Like Brennan, we believe that if we are encouraging teachers to include constructionist learning experiences in their classes, we should be including these experiences in our PD. In recent PD workshops, we have made changes to provide more constructionist learning experiences, but we have often had to negotiate the *tension between direction and discovery* when designing and running the PD activities. In past evaluations of our workshops we found that teachers have enjoyed the sessions that had step-by-step exercises more than the sessions that involved self-guided activities (Prieto-Rodriguez & Hickmott, 2016). For example, one teacher thought “it was very useful to have printed sheets to follow...” during the sessions. Conversely, some teachers responded to the surveys with suggestions to include more self-directed activities and “problems to solve with minimal guidance on how to solve them”. Consequently, we have had to learn to negotiate the balance of providing different types of instruction or pedagogical approaches, as the teachers themselves are likely to do in their own classes.

In the 2016 and 2017 workshops, we began to include sessions that allowed the teachers to spend more time on self-guided exploration and play. For example, in the 2017 Primary School workshop we included a session themed around the Creative Computing curriculum guide and included the “10 Blocks” activity from that guide in the session. In the “10 Blocks” activity, learners are asked to create a Scratch program but are limited to 10 different types of blocks, which the instructor can learn about before they run the activity. These types of activities can help teachers believe that it is “...ok to not know everything as the teacher”, as stated by one of the survey respondents,

Another difficulty that we have experienced designing our PD, that we consider to be encompassed within the *tension between direction and discovery*, is related to the limited amount of time for the PD. In most of our workshops, 20-30 local teachers have attended six-hour long workshops for two consecutive days. The workshops take place during the school term and at a local university, so the teachers have to be away from their classes during this time and their schools need to cover the costs of substitute teachers. Consequently, we plan many of the sessions prior to the workshop, in order to make the best use of the time we have with the teachers, and this planning has often resulted in activities that do give learners specific direction. We are aware of approaches to PD that solely involve self-directed, constructionist activities, such as the four-day *Constructing Modern Knowledge* workshops run by Martinez and Stager (2013). However, we have typically had 1-2 instructors during the workshops, which would have made it difficult to provide guidance to 20-30 teachers working on open-ended projects, and the two days has not been enough time to cover essential CK and to also run design activities. There are also difficulties that are discussed in the *tension between expert and novice* and *tension between actual and aspirational* sections, which prevent us from only including self-directed, constructionist sessions in our PD. To address this tension, we intend to run after-school sessions over a term in our PD. This structure would allow teachers to work on design projects during their free time and for us to provide guidance during the sessions or through email.

Tension between individual and group

The *tension between individual and group* refers to the challenge of facilitating productive collaboration between the teachers that attend the PD. As Brennan (2015) argues, learners should be encouraged to connect with each other and learners working in groups can learn new perspectives from one another. We have not had to negotiate this tension to the same extent as Brennan, which could be due to the collaborative and face-to-face nature of our workshops. There are a few aspects of our workshops that allow for collaboration to occur with minimal intervention from us, such as introduction sections, meal breaks, and collaborative hands-on or lesson planning activities.

One challenge, that we consider to be part of the *tension between individual and group*, is the sustaining of knowledge sharing and connections after the PD has been completed. Unlike the *ScratchEd* model, we do not have an online community for our PD for teachers to share resources or discuss ideas in. Additionally, teachers that have attended the workshops have usually been from different schools and it has been rare for more than two teachers from the same school to attend the same workshop. To help address this challenge, we decided to design a workshop in 2018 that is focused on assisting experienced C&CT teachers acquire the CK and PCK they need to establish local professional learning communities (PLCs).

Tension between expert and novice

The *tension between expert and novice* refers to the tension between the teachers who are considered to be knowledgeable about the content in the PD activities (the experts) and the teachers who have only begun to learn the content (the novices). Like Brennan (2015), the teachers that have attended our PD have had a wide range of CK and experience teaching C&CT. Some of the teachers that have attended our PD were professional software developers before their teaching careers, whereas some other teachers had not learned anything about C&CT prior to our PD. However, the ways in which we have experienced this tension seem to differ from (Brennan, 2015), which could also be due to the differences in the PD format. Brennan (2015) gave examples that were mainly experienced in the interactions between the teachers when they were learning together. On the other hand, we ourselves have experienced this tension when trying to choose planned learning outcomes, session themes and activities that are appropriate for teachers that have varying levels of CK and PCK.

We consider the difficulty of providing resources and instruction that are suitable for a wide range of teachers' CK to be encompassed by the *tension between expert and novice*. We have encountered this difficulty since our first PD in 2013 and, consequently, negotiating this tension has been the precursor to several of the major changes we have made to our PD. For example, prior to 2015 we had only run one workshop per year, with activities that involved high school Digital Technologies concepts. However, in 2015, we decided to run two different workshops, the Introductory and Advanced workshops, which had activities that were designed to cater for teachers with different levels of CT CK. This decision was a result of reflecting on our observations of teachers' varying levels of CK and responses to the feedback surveys in the previous years' workshops. We have also given the teachers the choice to participate in different activities that are suitable to their level of expertise, which are similar to the breakout sessions in the Meetups described by Brennan. For example, in the 2016 High School workshop, teachers chose between a session that introduced the teaching of Data Science in R and an Introduction to General-Purpose Programming with Sonic Pi.

As we have often presented short talks before the activities in each PD session, we have often been positioned as the experts during the PD. However, some of the teachers have had expertise in areas of CK and PCK that we did not have. Consequently, we have invited some teachers to assist us when running activities or present their own sessions, and also encouraged the expert teachers to share their knowledge with the rest of the group during the PD. For example, in the 2017 Primary School workshop we invited a teacher that had attended a workshop in an earlier year, to help us run a Collaborative Lesson Planning Activity. That teacher had extensive knowledge about relevant K-6 curriculum outcomes, that we did not have ourselves, and consequently was able to help the teachers map their lesson plans to these outcomes.

We have also encountered *the tension between expert and novice* when we have considered the changes to make to the PD from teachers' suggestions in the feedback surveys. Although we have run workshops that have had activities that aim to cater for different levels of teachers' CK, we still have received feedback that indicated we may have needed to differentiate the content more. An example of this was found in the feedback responses of the 2017 Maths workshop, which was intended to be suitable for teachers with some CK and some experience Coding in a Blocks-based language, like Scratch. One teacher stated that the workshop content was "All well delivered and at right level", whereas another teacher responded that "As a beginner of coding, I found the pace of the coding activities too fast." Despite the changes we have made to the PD to address the challenge of differentiation, negotiating this tension seems to be inevitable, as the teachers that have attended the PD have always had a wide range of CK. As discussed in the *tension between anthropology and assessment* section, one way to address this would be to measure the teachers' CK before the PD, which could help us plan sessions that are appropriate for teachers with different levels of CK.

Tension between actual and aspirational

The *tension between actual and aspirational* refers to the difficulty of providing constructionist learning experiences to teachers that they can replicate in their classes. Brennan (2015, p. 295) argues that "constructionist learning experiences are fundamentally at odds with the lived reality of K-12 education" in many ways. However, we contend that teachers that have attended our PD would be able to incorporate some constructionist learning experiences into their classes, particularly in view of the emphasis that creative thinking has in the general capabilities of the Australian Curriculum (ACARA, 2018).

Unlike Brennan (2015), the changes to our PD have been largely been motivated by the challenges reported by the teachers, which would be considered to be on the *actual* side of the *tension between actual and aspirational*. The feedback from the teachers in our workshop surveys, particularly those in 2013 and 2014, indicated that the teachers wanted resources that could be directly used in their classes, such as lesson plans. Furthermore, the current consensus is that PD is effective when it is aligned with the needs of the participating schools and involves collaboration with school administration (Desimone, 2009; Menekse, 2015). Therefore, we have often designed PD which has addressed the *actual* nature of K-12 education, rather than the *aspirational* nature of a more constructionist approach.

In recent years' workshops, we have also included sessions that involved open-ended, constructionist learning experiences such as the Creative Computing session in the 2017 Primary School workshop. Although there is a climate of high-stakes testing in Australia, there are ways to introduce design activities that align with outcomes in the relevant curricula. For example, our local state education authority states that, "Designing, making, data collection and analysis" are part of the Science and Technology subject area (NSW Education Standards Authority, 2018). Additionally, in the national curriculum there are general abilities, such as Critical and Creative Thinking (ACARA, 2018), which could be addressed through the inclusion of design activities in teachers' classes.

One of the other challenges that we consider to be part of the *tension between actual and aspirational* is related to the inclusion of C&CT into a curriculum that is already considered to be overcrowded (Polesel, Rice, & Dulfer, 2014). Ideally, teachers would have sufficient time to learn and teach C&CT in addition to other subjects, but it is challenging for teachers to find this time. To address this challenge, local educational authorities and researchers have recommended that teachers find ways to integrate C&CT across existing subject areas (Barr & Stephenson, 2011; NSW Education Standards Authority, 2018). One way we have found to negotiate this tension is the creation, in 2017, of two workshops that were focussed on integrating C&CT with mathematics: ScratchMaths for Primary School teachers and Networks for High School mathematics teachers. The contents of these workshops were aligned with specific curriculum outcomes, from mathematics and C&CT.

Tension between anthropology and assessment

One of the tensions that we have experienced, which was not identified by Brennan (2015), is the tension between assessing the teachers' CK and PCK and restricting our evaluation to quality assurance and self-reported measures of self-efficacy or impact. Brennan (2015, p. 295) does not

mention the assessment of the teachers during the *ScratchEd* PD directly but does state that there is a “lack of meaningful metrics for assessment and evaluation” when defining the *tension between actual and aspirational*. However, we contend that there are metrics for assessment of CK and PCK that can be meaningful and appropriate for suitable contexts and, consequently, we consider the *tension between anthropology and assessment* as separate to the *tension between actual and aspirational*. We also recognise that the formats of different PD in the *ScratchEd* PD may be more difficult to assess than workshops. For example, it could be impractical to assess teachers’ CK and PCK before and after a breakout session in one of the *ScratchEd* Meetups.

Although the teachers that have participated in our PD have not suggested that we include assessment, we contend that there are two main reasons why a more rigorous evaluation of CK and PCK would be beneficial. Firstly, in order to improve the effectiveness of our PD, we need to know whether we have had a positive effect on teachers’ CK and PCK. Presently, we base improvements to our activities based on self-reported measures given by participating teachers. Secondly, organisations and governments spend significant funds on PD and, consequently, there is a need to identify whether the PD has had a positive impact on teachers’ classroom practices beyond teachers self-reporting. For example, researchers from Google, who have funded the CS4HS (Computer Science 4 High School) program, have recently begun investigating the long-term impacts of their programs on teachers’ self-reported knowledge of C&CT and their beliefs about teaching C&CT (Ravitz, Stephenson, Parker, & Blazevski, 2017).

One of the challenges that we have encountered, that we consider part of the *tension between anthropology and assessment*, is the difficulty of identifying appropriate instruments to measure teachers’ CK and PCK. There have been instruments developed for assessing understanding of CK, but much of this work has been done in tertiary education (Guzdial, 2015), and these instruments may not be appropriate for measuring the CK that K-12 teachers need. In his review of PD studies conducted in the United States of America between 2004 and 2014, Menekse (2015) reports that seven of the studies involved some assessment of teachers CK. However, a deeper reading of these studies uncovered that only three of them involved measurement of CK that was not self-reported. Furthermore, these three studies used instruments that were developed by the authors or by an unidentified source, and did not report reliability of scales. There is also currently limited research into the assessment of PCK specific to C&CT (Saeli et al., 2011; Yadav et al., 2018). A form of PCK, which is referred to as Computational Pedagogical Content Knowledge (CPACK), was present in one of the studies reviewed by Menekse (2015). However, in the study in which Yasar and Veronesi (2015) introduce CPACK, they do not report measures of teachers’ CPACK or include details on how data can be collected or analysed to measure teachers’ CPACK. Similarly, while the research conducted by Yadav et al. (2018) examines teachers’ PCK through teaching vignettes, their methods do not include quantitative measures.

Discussion and Further Research

In this paper, we have analysed our PD through the framework of the ‘tensions negotiated’ defined by Brennan (2015). In this section, we summarise our findings and discuss the implications for PD design. We also outline our plans for future research.

To negotiate the *tension between tool and learning* we have found it is helpful to include a balance of activities that address both tools and learning, as we recognise that both CK and PCK are essential. We have also identified potential technical issues that can be encountered when using the different tools, and believe it is important that teachers are made aware of these issues.

To negotiate the *tension between direction and discovery* we have observed it is valuable to include activities where teachers can explore concepts without direct instruction from a resource or instructor, such as the *10 Blocks* activity. We have also found that it is beneficial to provide teachers with choices of activities in the form of parallel sessions, or to let them choose their own direction for learning and then offer instructional support. In response to survey feedback, we have included sessions where teachers worked together to create an artefact unaided, such as a unit plan, or sessions where teachers were given a problem to solve with minimal guidance. These sessions were very well received in subsequent years.

To negotiate the *tension between individual and group* we have included an increasing number of collaborative activities, such as collaborative lesson planning and problem solving. For our PD design for 2018, we have planned to train and support teachers who are interested in establishing PLCs. These PLCs could encourage the sharing of knowledge and collaboration between teachers after they have participated in the PD. The development of the workshop was influenced by the work to develop Master Teachers in England (Sentance et al., 2014).

To negotiate the *tension between expert and novice* we observed that it is useful to design extension activities for the teachers that complete the PD activities quickly. We have differentiated the activities to cater for different levels of CK within workshops and created different workshops to accommodate different PCK needs according to the level of schooling that participants teach at. We also found that inviting expert teachers to share their experiences implementing C&CT with their classes was highly regarded by all participating teachers.

To negotiate the *tension between the actual and aspirational* we have started running workshops that address existing subject areas and specific curriculum outcomes. It is assumed that the teachers participating in these workshops already have subject knowledge in that domain, and that they would be able to assess those subjects already. These workshops are narrower in scope but provide greater depth of learning. We have made changes to the PD that are intended to address the *aspirational* aspects of *constructionism* as well. We have found these changes have provided teachers with skills and resources that can help them introduce design activities to their classes, such as the Creative Computing curriculum guide (Brennan et al., 2014). Another PD project planned for 2018, which may address this tension, involves running the PD over a longer period of time. This could allow for longer-term open-ended projects that teachers can, in turn, develop with their students.

The new identified tension, *the tension between anthropology and assessment*, extends the tensions negotiated by Brennan (2015). Although there are arguments for measuring outcomes in our PD, as explained in the previous section, it could also be argued that PD with constructionism as a framework for action should not involve any assessment of learning outcomes. Papert and the MIT Logo group were known for being opposed to standardised testing (Agalianos et al., 2006), and Papert (1987) argued that focusing on assessing outcomes could lead to technocentrism. However, there are examples of constructionist research that involve the collection of quantitative measurable learning outcomes, including the study of children designing instructional mathematics games conducted by Harel and Papert (1990). Harel and Papert (1990, p. 10) referred to these outcomes as “thinner results” because, while they could be used to assess the students’ performance, they are not as in-depth and rich as the qualitative data collected in the study. While we agree with this sentiment, the analysis of quantitative data helps us understand what we can do to improve our PD. Furthermore, the reporting of quantifiable measures helps address the concerns of funding agencies and government bodies regarding the impact of our PD on teachers.

We believe that learners, whether they are teachers or students, should be encouraged to explore ideas through the self-directed creation of artefacts that are personally meaningful to them. However, as Noss and Hoyles (1996) argue, this does not mean that we should not plan for learning outcomes or that there should be no assessment of these outcomes. Thus, we need to negotiate the tension between acting as anthropologists and assessing teachers’ learning in CK and PCK with quantitative measures. Our next PD workshops will integrate our desire for authentic constructionist experiences and the desire to improve these experiences by ensuring that the PD has had a measurable positive effect on teachers’ CK and PCK.

Acknowledgements

The authors would like to thank Google Inc. for funding the CS4HS workshops held at our institution. We would also like to thank the university academics who presented talks and gave laboratory tours during the workshops, as well as the administrative staff of our institution.

References

- ACARA. (2018). F-10 Curriculum - General Capabilities. from <https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/>
- Agalianos, A., Whitty, G., & Noss, R. (2006). The Social Shaping of Logo. *Social Studies of Science*, 36(2), 241-267.
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: implications for teacher knowledge. *Journal of Educational Technology & Society*, 19(3), 47.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48-54.
- Blikstein, P. (2013). Digital fabrication and 'making' in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, 1-21.
- Brennan, K. (2015). Beyond Technocentrism: Supporting Constructionism in the Classroom. *Constructivist Foundations*, 10(3), 289-296.
- Brennan, K., Balch, C., & Chung, M. (2014). Creative Computing - Scratch Ed. Retrieved March 2018, from <http://scratched.gse.harvard.edu/guide/>
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.
- Brown, Q., & Briggs, A. (2015). The CS10K initiative: progress in K-12 through "exploring computer science" part 1. *ACM Inroads*, 6(3), 52-53.
- Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). A future for computing education research. *Communications of the ACM*, 57(11), 34-36.
- Desimone, L. M. (2009). Improving Impact Studies of Teachers' Professional Development: Toward Better Conceptualizations and Measures. *Educational Researcher*, 38(3), 181-199.
- DiSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *The journal of the learning sciences*, 13(1), 77-103.
- Falkner, K., Vivian, R., Falkner, N., & Williams, S.-A. (2017). *Reflecting on Three Offerings of a Community-Centric MOOC for K-6 Computer Science Teachers*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, Washington, USA.
- Fisher, L. M. (2016). A decade of ACM efforts contribute to computer science for all. *Communications of the ACM*, 59(4), 25-27.
- Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015, 18-20 March 2015). *Computing education in K-12 schools: A review of the literature*. Paper presented at the 2015 IEEE Global Engineering Education Conference (EDUCON).
- Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8(6), 1-165.
- Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive learning environments*, 1(1), 1-32.
- Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Education Week*, 61-65.
- Martinez, S. L., & Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom: Constructing modern knowledge* press Torrance, CA.
- Menekse, M. (2015). Computer science teacher professional development in the United States: a review of studies published between 2004 and 2014. *Computer Science Education*, 25(4), 325-350.

- Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers* (Vol. 17): Springer Science & Business Media.
- NSW Education Standards Authority. (2018). Digital Technologies and ICT Resources. Retrieved 11 March 2018, from <http://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/learning-areas/technologies/coding-across-the-curriculum>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*: Basic Books, Inc.
- Papert, S. (1987). Information Technology and Education: Computer Criticism vs. Technocentric Thinking. *Educational Researcher*, 16(1), 22-30.
- Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, 39(3/4), 720.
- Polesel, J., Rice, S., & Dulfer, N. (2014). The impact of high-stakes testing on curriculum and pedagogy: A teacher perspective from Australia. *Journal of Education Policy*, 29(5), 640-657.
- Prieto-Rodriguez, E., & Hickmott, D. (2016). *Preparing teachers for the Digital Technologies curriculum: preliminary results of a pilot study*. Paper presented at the Constructionism 2016 Conference, Bangkok, Thailand.
- Ravitz, J., Stephenson, C., Parker, K., & Blazeovski, J. (2017). Early Lessons from Evaluation of Computer Science Teacher Professional Development in Google's CS4HS Program. *ACM Trans. Comput. Educ.*, 17(4), 1-16.
- Resnick, M. (2017). *Fulfilling Papert's Dream: Computational Fluency for All*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, Washington, USA.
- Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching programming in secondary school: a pedagogical content knowledge perspective. *Informatics in Education*, 10(1).
- Sentance, S., & Csizmadia, A. (2016). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 1-27.
- Sentance, S., & Csizmadia, A. (2017). *Professional Recognition Matters: Certification for In-service Computer Science Teachers*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Sentance, S., Humphreys, S., & Dorling, M. (2014). *The network of teaching excellence in computer science and master teachers*. Paper presented at the Proceedings of the 9th Workshop in Primary and Secondary Computing Education, Berlin, Germany.
- Shulman, L. S. (1986). Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher*, 15(2), 4-14.
- Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2016). Computer science in K-12 school curricula of the 21st century: Why, what and when? *Education and Information Technologies*, 1-24.
- Yadav, A., Sands, P., Good, J., & Lishinki, A. (2018). Computer Science and Computational Thinking in the Curriculum: Research and Practice. In J. Voogt, G. Knezek, R. Christensen, & K.-W. Lai (Eds.), *Handbook of Information Technology in Primary and Secondary Education* (pp. 1-18). Cham: Springer International Publishing.
- Yasar, O., & Veronesi, P. (2015). *Computational Pedagogical Content Knowledge (CPACK): Integrating Modeling and Simulation Technology into STEM Teacher Education*. Paper presented at the Society for Information Technology & Teacher Education International Conference 2015, Las Vegas, NV, United States.

Sharing is Caring in the Commons – Students’ Conceptions about Sharing and Sustainability in Social-Ecological Systems

Arthur Hjorth, arthur.hjorth@u.northwestern.edu

Center for Connected Learning and Computer-Based Modeling, Northwestern University, USA

Corey Brady, corey.brady@vanderbilt.edu

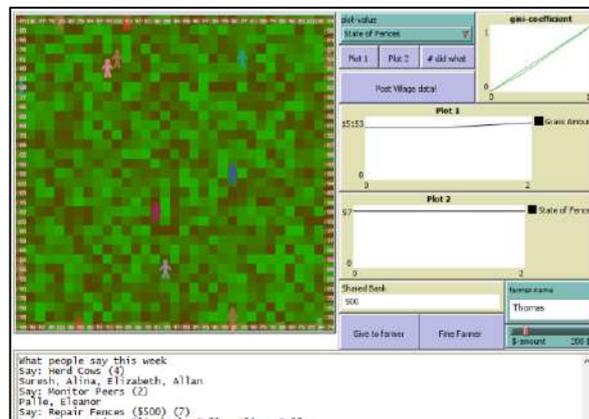
Department of Teaching and Learning, Vanderbilt University, USA

Uri Wilensky, uri@northwestern.edu

Center for Connected Learning and Computer-Based Modeling, Northwestern University, USA

Abstract

In this paper, we present the design and analysis of a high school environmental science participatory simulation-based activity about sharing natural resources. In particular, we focus on this activity’s utility in surfacing students’ diverse ways of thinking about Social-Ecological Systems, offering entry points for Constructionist design. The activity was implemented as part of a three-week unit in an environmental science AP class. At the core of the activity was a participatory NetLogo simulation in which students played the role of dairy farmers. Through their interactions with the simulation and collective and individual decision-making, students struggled to reason and argue productively about the difficulties involved with sharing natural resource systems.



The virtual grazing ground and accompanying data interface gives students insights into how their individual and collective decisions affect the ecosystem.

We gave students two written assignments, in which they were asked questions about true, historical descriptions of communities sharing natural resource systems. Here we analyse the responses, identifying four distinct ‘thinking patterns’ across student responses. We discuss how these four patterns were productive in so far as they helped students to reason about the historical cases, but also how these patterns in their thinking restricted them from thinking productively about the full nature of the case studies. Finally, we discuss how future designs might address the less productive aspects of these patterns.

Keywords

NetLogo; agent-based modelling; HubNet; participatory simulation; social studies; complex systems

Abstract

This paper presents data from a high school implementation of a unit on common-pool resource sharing dilemmas in Social-Ecological systems. We designed and implemented a computer-simulation-based

classroom activity in a high school Environmental Science course. Students took on the role of cattle farmers who had to maintain and share a virtual grazing ground and prevent a “Tragedy of the Commons.” We present an analysis of students’ responses to questions about how best to coordinate collective action and ensure sustainable utilization of the commons. We identify four patterns in students’ thinking and discuss how these patterns were simultaneously productive and a hindrance to reasoning about different aspects of this complex problem. Finally, we discuss the impact of our findings on our own iterative design-based research, as well as wider implications for future learning design and research on social-ecological systems and sustainability.

Introduction

The United States’ National Council for the Social Studies’ recent C3 framework (NCSS, 2013) proposes a new set of standards for Social Studies, including the use of computer simulations to test and understand the effects of policies and collective action. Despite decades of using computer modelling in science education (Wilensky & Jacobson, 2014), there have been relatively fewer applications in social studies classrooms. In this paper, we present data from a 3-week high school classroom implementation of activities focusing on the challenge of sustainable food production and featuring computer-based network-supported participatory simulations, or PartSims (Klopfer, Yoon, & Perry, 2005; Wilensky & Stroup, 1999b). Our core PartSim activities took Hardin’s (1968) seminal Tragedy of the Commons paper as a point of departure. Hardin argues that collectively shared resource systems are doomed to end in “tragedy” because individual actors within the system will act in their short-term interest to the detriment of the collective. However, Elinor Ostrom’s Nobel Prize-winning work (Brondizio, Ostrom, & Young, 2009; Ostrom, 1990, 2007) has shown that while true under certain conditions, other conditions make it possible for communities to sustainably share and maintain resource systems. In such contexts, collectives can work to discover and sustain solutions to common-pool resource sharing problems. Ostrom and colleagues thus refer to a *drama* of the commons (*The Drama of the Commons*, 2002), which may end either in “comedy” or in “tragedy” depending on policies and practices adopted by participants.

Given the societal importance of the sustainable use of natural resource systems like oceans, aquifers, and populations of fish or game, it is critical to prepare students to envision and engage the potential impact of policies oriented towards addressing the drama of the commons. In this paper, we present an analysis of a study in which we designed and implemented a 3-week PartSim-based unit on sharing natural resource systems in a high school Environmental Science class. Our focal PartSim activities, and our unit as a whole, gave the classroom group the opportunity to role-play a village of cattle farmers. Together, they collaboratively explored possibilities for sustainable growth and for collectively maintaining a “virtual commons” in a simulated world. Our analysis focuses on students’ written responses to two sets of questions on this topic. We identify four patterns in student thinking, explain how they both helped and hindered students in making sense of the topic, and discuss why future learning design and research would benefit from addressing these thinking patterns.

What is hard about sharing

Following Hardin (1968) and Ostrom (1990), we can think of the difficulties of sharing a resource system from the perspective of an individual, and from the perspective of an individual’s larger community. From the perspective of an individual, the dilemma is a classical free-rider problem: the long-term benefits of collaboration are *shared* between all community members, whereas “cheating” benefits only the *individual* cheater. This leads to an individual-centred logic which dictates that cheating creates a net gain for the cheater. From the community’s point of view, the difficulties of sharing stem from the very nature of conditionally self-replenishing resource systems:

Resource systems can best be thought of as stock variables that are capable under favorable conditions of producing a maximum quantity of a flow variable without harming the stock or the resource system itself. (Ostrom, 1990, p. 30)

To exemplify, let us imagine a shared fishing pond: the *stock variable* is the amount of fish in the pond, and *flow variable* is the amount of fish being “produced” in the pond (through reproduction). *Favourable*

conditions include natural or anthropogenic factors like weather or nutrition levels in the water that affect how much flow the system produces. Finally, the last part of the quote alludes to the fact that these systems are often fragile and can be harmed or even driven to collapse if not cared for properly. The challenges facing communities who depend on sharing sustained access to these systems are then two-fold: First, how does a community collectively manage a system in order to create and sustain the set of ‘favourable conditions’ that optimize the amount of *flow* resource? This difficulty is exacerbated by the fact that the cost to maintain favourable conditions sometimes cannot be spread easily among community members. (For instance, while the community could decide to share the costs associated with feeding the fish, the time-cost involved in physically feeding the fish or measuring current nutrition levels will fall on one or a few individuals.) Second, and related, how does a community set up a system for sharing the produced resource that makes individual community members feel that their efforts are rewarded fairly? This difficulty involves a number of social challenges: for instance, establishing a means to ensure compliance with group norms that enables members of the community to “rest easy” that their fellow citizens are not cheating.

Research questions

In this paper we offer a preliminary analysis, through which we aim to better understand how students reason about communities that share resource systems. We ask,

1. How do students reason about the problems facing communities who rely on shared natural resource systems?
2. What kinds of community rules do students imagine would help address these problems, and why?

Design, Implementation, and Data

The study presented here emerged from the first year of a multi-year design-based research project (Cobb, Confrey, Lehrer, Schauble, 2003) in partnership with a classroom teacher at a suburban high school in the US Midwest. Together, we collaboratively designed a unit to run over 15 periods in a high school Environmental Science classroom. The 22 students in our partner teacher’s class were consented, and all chose to participate in the study. In addition to fitting into the larger themes of the course, we designed the unit to target the NCSS’s standards for social sciences (NCSS, 2013), and the NGSS science standards (NGSS Lead States, 2013), focusing on the use of computer simulations of ecological systems, to test the viability and effectiveness of policy interventions and collective action. Seven periods out of 15 during the first two weeks of implementation revolved around PartSim activities and are the focus of this study.

The Virtual Commons Sharing Activity

PartSims are socially shared computer-mediated simulations in which a group of students takes on the role of agents in a system whose aggregate behaviour emerges in real time as the students interact. As such, PartSims provide a means for a group of learners to experience a phenomenon from both the micro-level (as individual participants) and the macro-level (as a collective group experiencing the emergence of these outcomes). PartSims have a long history in the design of Constructionist learning activities, and have been used to teach topics as diverse as chemistry (Brady et al, 2015), geography and policy decision-making (Gilligan et. al, 2015), and as a tool for teaching complex systems principles (Brady et al, 2017; Guo & Wilensky, 2016).

From a Constructionist perspective, the purpose of a HubNet activity is to give a group of students a shared, manipulable object with an underlying set of complex systems interactions, and a socially meaningful purpose for these manipulations. Our PartSim was programmed in NetLogo (Wilensky, 1999) using the HubNet (Wilensky & Stroup, 1999a) architecture. The purpose of the PartSim was to give students the experience of sharing a “virtual commons” with collective responsibilities. The Virtual Commons HubNet activity began on day 3 of the unit. During the first two days, we ran two activities focusing on the Tragedy of the Commons. In the first, a non-virtual activity, students had to share a

fishing ecosystem consisting of candy fish. In the second, students had to share a grazing ground in a HubNet NetLogo model. Both of these activities were designed with Hardin’s original constraints in mind: students were not allowed to speak or coordinate with each other, nor were they permitted to share responsibilities. In both of these activities, students inevitably “crashed” the ecosystem and enacted the Tragedy of the Commons.

In contrast, and reflecting an important difference between Hardin’s *thought experiment* and Ostrom’s *empirical research*, our subsequent Virtual Commons PartSim was designed to give students incentives to work together to maintain their resource systems: Students played the role of dairy farmers who relied on a shared grazing ground. For each “week” (turn) of the activity, students held a town hall meeting in which they assessed the state of the commons by looking at data together, and agreed on what tasks would need to be done in the following week as well as, *by whom*.

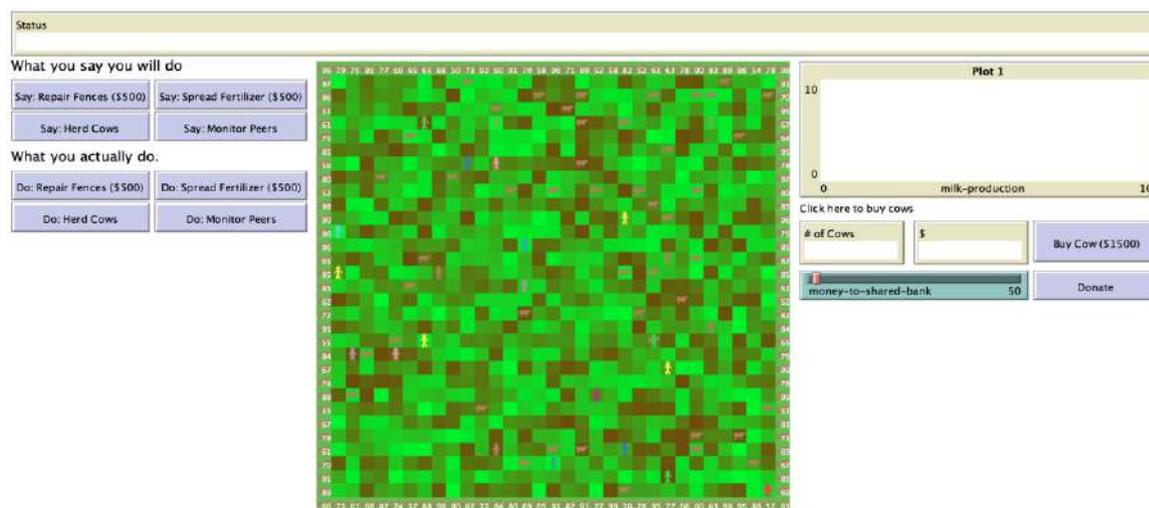


Figure 1. Each student’s private view of the Virtual Commons PartSim model.

Each student could take on one of four tasks (see Figure 1): (a) spreading fertilizer (accelerating the regrowth of grass on the commons); (b) repairing fences (ensuring that the village’s cattle do not escape); (c) herding their own cattle (increasing their own milk production for the week); or (c) monitoring their peers (confirming that they were doing what they promised to do). Importantly, the simulation allowed students to “cheat” – by promising to engage in one task, while secretly doing another. This made monitoring an important task, as monitors enabled the village to catch cheaters.

During the town hall meetings, students gathered around the shared view (Figure 2), which displayed relevant information, including the number of cows on the field, the amount of grass on the grazing ground, the total milk production, the Gini coefficient for the farm community, and the general state of repair of their fences. Based on these data and their shared experiences up until this point, students discussed the dilemmas they faced, both as individuals and as a group, and devised means to address them through collective action. In addition to pursuing collective farming goals, they also conducted “experiments” to learn information about their environment. These included varying the number of people who spread fertilizer (to assess how quickly grass grew back on its own or to estimate “carrying capacity”) or repaired fences (to assess the rate at which fences deteriorated).

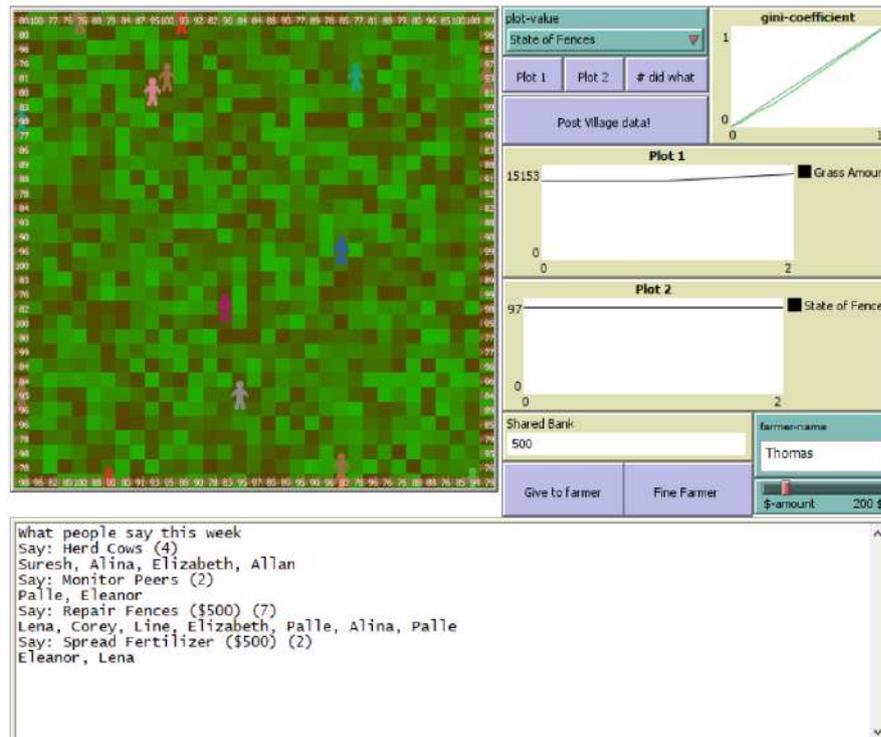


Figure 2. Students' collective view of the grazing ground.

Our aim with the Virtual Commons PartSim activity was to give students a contrasting experience to the Tragedy of the Commons, so that they might come to understand problems facing sustainable sharing as well as to explore the viability of potential solutions. By exploring the aggregate outcome of their collective choices, students experienced the potential problems that commons-sharing communities face, while also confronting the strengths and weaknesses of different solutions. We hypothesized that this activity would support students in reasoning about other, similar, situations in which social collectives share a commons.

The Elicitation: Two Real Cases of Resource Sharing

At the end of each of the three weeks of the unit, students were asked to respond individually to questions about written case studies in which communities share resource systems. These assignments were given as homework over the weekends, and questions were distributed via Google Forms. The data presented in this paper come from the first and the second of these assignments. As mentioned above, our purpose analysing these data is more to identify *patterns in students' ways of thinking* about resource systems, than to argue for the effectiveness of our unit in creating conceptual change or learning in this area.

These two homework assignments presented students with historical cases from Ostrom's (1990) work. We decided to use these two cases for a variety of reasons: first, because Ostrom's own treatment of them provided us with an "expert analysis" with which we could compare student responses. Second, both these cases took place years ago before technological innovations would offer easier solutions to some of these problems. By taking out technical solutions, we forced students to reason about the social and ecological aspects of the dilemma, instead of coming up with intricate technical solutions like satellite surveillance or GPS tracking. Finally, these cases involved self-replenishing but fragile resource systems with similar features to the one that students experienced in the Virtual Commons, but were still different enough that we could see whether students would reason about these systems by drawing on their experiences from the Virtual Commons.

The first case described a community of farmers outside of Valencia, Spain who shared an irrigation canal system in the Middle Ages. Maintenance of the canals relied on the coordinated effort of many

people and more resources than any one farmer could afford. Additionally, the amount of water fluctuated each year due to differences in precipitation.

The second case described a community of fishermen in Sri Lanka living by a bay, whose livelihood depended on the use of large *seine* nets. These nets were so costly that seven families typically had to co-own each one, and so large that only one boat could fish in the bay at a time. The nets, then, posed both financial and logistical constraints, forcing people to cooperate. In addition, maintaining the health of the bay (a fragile ecological resource system) was yet another factor requiring community collaboration.

Analysing the Responses

The questions (9 questions for the first case and 8 for the second), probed student thinking about the difficulties that these communities might face, and asked students to think about what rules the community could instate to address the difficulties, and probed their thinking about why those rules would help. We changed the number of questions between the two cases as part of our iterative design process, because a preliminary look at student responses after the first week of implementation suggested to us that two of the questions overlapped in a confusing manner. Our second iteration of questions collapsed these two questions into one and rephrased the prompt.

Because we were still in the exploratory stages of our research, we took an open-ended approach to coding students' responses. We used "ways in which students identify challenges at the community or individual level" as "sensitizing concepts" (Miles & Huberman, 1994) in our initial coding. The first two authors coded all student responses individually, and we then converged on four interesting patterns that both researchers noticed across many student responses.

Findings

In this section, we will describe four patterns in students' thinking that emerged from our open-ended analysis, and that we believe future learning design and research should focus on. We describe each pattern and how it relates to the specific details of each of the cases; provide examples of student responses that exemplify each pattern; and, finally, discuss how each pattern seemed to participate in the broader reasoning of the students who exhibited it. These patterns proved to be two-sided in their effects on student reasoning about common-pool resource sharing. On one hand, they acted as productive tools to support students in thinking and in articulating ideas; on the other, they seemed to foreground particular aspects of sharing dilemmas at the cost of backgrounding other features, thus tending to limit student thinking in some ways.

The "Fixed Flow" Pattern

We found that when asked to identify a potential tragedy of the commons in each of the two cases, it many students reasoned primarily about short-term aspects of sharing the *flow* resource. In contrast, very few students mentioned long-term, *stock*-related aspects relating to optimizing the yield of the system.

For example, consider this typical *flow*-focused response to Case 2 (fishing):

By cheating [...] that group would get the most fish and [...] affect the other groups by forcing them to split a smaller number of fish and depriving them of equal opportunity. (S12)

Compare this thinking with the following typical *stock*-focused response, also to Case 2. (Note: Here and elsewhere students' spelling and grammar are maintained.)

If one of the fisherman caught more than assigned then the it effects the rest of the community because thre wouldn't be enough fish to reproduce and have enough for next 'harvest.' (S21)

Interestingly, both students are talking here about overfishing. But the *function* of overfishing is different and reflects the distinction between fish as *stock* and fish as *flow*. In the former response, the student identifies the problem with overfishing as there being an immediately lower number of fish to split

between the rest of the fishermen at that moment. In the latter, the student identifies the problem with overfishing as diminishing or threatening *future* yield.

This illustrates the importance of thinking about both stock and flow. However, we often saw that students focused only on flow. In Case 1 (irrigation), 21 students brought up potential *flow*-related problems with people taking too much water and not leaving enough for the rest of the community, and 14 students brought up potential rules to address these problems. In contrast, only 10 students mentioned *stock*-related problems, and only four students mentioned solutions to them. Likewise, in Case 2, 20 out of 22 students mentioned potential *flow*-problems - relating to taking more than one's share by fishing too much - and 12 brought up rules to prevent them. In contrast, only 3 student responses brought up problems relating to stock – the maintenance of the system and its ability to produce future fish – and only 2 identified potential rules to address these problems. Moreover, those who did reason about long-term stock issues were very likely *also* to reason about short-term flow issues, but not vice versa: of the 10 students who mentioned stock problems, 8 *also* mentioned flow problems.

We call the thinking pattern that attends to flow-aspects *at the expense of stock-aspects* the “fixed flow” pattern, because responses exhibiting this pattern treat the shared resource as an invariant quantity of flow – water in the canal, or fish in the bay. That is, this thinking pattern attends to “fair sharing” of the flow, possibly making a hidden assumption that the group's access of the resource will not damage the system's capacity to produce future flows. This pattern of thinking can be problematic if it prevents students from considering how to preserve the “favourable conditions” that optimize the long-term availability of flow resources to be shared, either through maintenance of the stock or the system's infrastructure.

The “Social, Not Ecological” Pattern

We also observed a pattern of thinking in student responses that emphasized solving the social problems of freeloading, at the expense of considering ecological problems. Students' responses exhibiting this pattern often assumed that if the community managed to collaborate, then everything would be fine. For example, consider the contrast between this response to Case 1:

If the farmers collaborate, they ensure some level of food security and economic security.
(S7)

...and these responses (to Case 1 and Case 2, respectively):

If they collaborate, each farmer will have enough for his on family. (S16)

If the fishing groups cooperate then they will each get some fish and there will be a sustainable population (S20)

In the first response, the student explicitly reasons that while collaborating will ensure some level of food security, it will not absolutely *guarantee* it. In the last two, the students seem to assume that if the farmers or fishermen prevent each other from cheating, everything will be fine. But there are no guaranteed ‘happily ever after’ scenarios in the commons, even for communities that share resources equitably. Even if everyone gets their fair share, a community can run out of fish or grass if they do not solve the long-term, yield-related *stock* problems. Because of its emphasis on the social dimension of the sharing dilemma, we described this pattern as a “social, not ecological” way of thinking. We saw this pattern in 9 responses in Case 1, and 8 in Case 2, with four students exhibiting the pattern in both their responses to both cases. These responses suggest that students may forefront the social coordination problems without taking into account the ecological dimensions of the system. While the social side of problems *is* important, it cannot stand on its own, and future design iterations of our unit will try to forefront the ecological side more in an attempt to address this pattern.

The “Social, Not Ecological” pattern and the “Fixed Flow” pattern share similarities in the features of common-pool resource sharing that they background. In particular, both are focused on concerns about equal access to flows at the expense of questions about the sustainability of stocks. However, for us they represented different ways of thinking, because of the way they engaged with social dynamics and norms. As we seek to engage groups of learners in broadening their perspectives, these ways of

thinking represent different entry points and leverage points for the design of activities and learning environments.

The Profit and Competition Pattern

As we have noted, our student responses focused on short-term, *flow*-based problems, with almost all students in the population mentioning these challenges. One pattern in students' *characterization* of these problems had to do with an interpersonal source or motivation for cheating in the commons. This pattern grounded reasoning about why people cheat in an implicit or explicit belief that people in groups are *competitive*, such that community members compete over the commons and attempt to generate the most individual profit. Responses exhibiting this pattern focused on motivations for cheating that pertain to people's *desires* rather than their *needs*. Consider these three responses:

Farmers would [cheat] to get ahead and water more and more crops to make money. (S12)

and,

They want to produce the most and be the best farmer. They want to be known as having the best most consistent product. (S6)

and,

By cheating in a way like going out to fish earlier than everyone, that group would get the most fish and be the most wealthy and profitable. (S13)

Twelve responses to Case 1 and twelve to Case 2 brought up a competitive profit motive for cheating, with 7 students responding in this way to both cases. However, Ostrom's research shows that community members may cheat for a wide variety of reasons beyond the desire to profit over others. For instance, community members experiencing temporary financial or health-related difficulties may cheat out of necessity. Alternatively, community members may feel an incentive to cheat if they perceive that others are cheating and getting away with it. Thus, preventing cheating can be a very complex matter. But students exhibiting the "profit and competition" pattern tended to reduce this problem to an interpersonal competitive dynamic.

Ostrom's work suggests that designing collaboration rules for preventing cheating requires a deep understanding of the underlying reasons for why people cheat, as these motivations are what the rules must target. Thus, while the "profit and competition" pattern may be productive for reasoning about preventing one kind of cheating, it may draw students' attention away from solutions to other manifestations of *flow*-sharing problems.

The "Fixed Human Nature" Pattern

Another prominent pattern in student responses that addressed *flow*-sharing problems appeared to be based in an image of human nature as having essential qualities independent of context or situation. This pattern was distinguishable from the "profit and competition" pattern in which the competitive nature was seen as coming out of the *interactions* between people. In contrast, in the "fixed human nature" pattern, students downplayed the impact of social arrangements and conditions in preventing cheating, and instead saw rule-breaking as inevitable. Humans (or some types of humans) were seen as essentially predisposed to pursue antisocial behaviour that would benefit them individually. We provide examples of this pattern of thinking from responses to Case 1.

In some instances, students posited essentialized features of human nature in particular individuals or types of individuals within the broader population. For example, the following response indicates a belief in the existence of an antisocial element:

A few rotten apples out of the farming bunch may misuse the water and everyone's fields would collapse. (S10)

In other responses, students expressed essentialized understandings of human nature in general, or they bridged from behaviours exhibited by individuals to traits of all people. Consider the following response:

Certain farmers will always ask for more than they need due to the selfish nature of humans.
(S22)

Here, while the phrase “certain farmers” suggests a conception about a *subset* of the population, “the selfish nature of humans” points toward a more general human trait. Similarly, in reflecting on the impact of variability in the flow resource of water in Case 1, another student remarked:

I think this because people worry about themselves first and foremost, then secondly comes the idea of sharing. (S18)

Under this conception of human nature, it is possible to make context-independent statements about how humans will behave and interact. This pattern in student thinking stands in contrast to an alternative conception of human nature, which holds that it is malleable, and that human behaviour is highly context-dependent. A “fixed human nature” pattern tends to underestimate the potential for policies, rules, and information to alter collective patterns of action. Thus, this pattern may hinder students’ ability to conceive of creative responses to the dilemmas and challenges that constitute the “drama of the commons,” limiting their capacity to imagine solutions for sustainable common-pool resource sharing and management.

Discussion, Limitations, and Conclusion

This paper has presented an analysis of high school students’ reasoning about communities sharing natural resource systems as part of a 3-week unit on sustainable food production featuring participatory simulation activities. Based on our analysis, we identified four patterns across students’ responses. These patterns proved to be productive for students to reason about some aspects of the cases, but we speculate that they also hindered students in reasoning about other important aspects.

We have taken a design-based research approach to the iterative construction and implementation of this unit, and the design changes to our subsequent iterations have aimed to better identify and address the problematic aspects of these thinking patterns. We have implemented the activity sequence analysed here two additional times, including the following refinements, responsive to our analysis of patterns of thinking.

In order to address the “fixed flow” pattern, our current design gives students increased access to the *data* produced by the model in order to provide the classroom group with opportunities to explore how their behaviours, both as individuals and as a community, effect changes in the “size of the pie” from which they are taking equal or unequal shares.

To address the “social, not ecological” pattern, we have made two changes to the simulation itself. First, we reduced the carrying capacity of the system, and second, we have sped up how quickly the simulation runs. Both of these changes increase the likelihood that student villages will experience a crash, even with a modest number of cows. Our aim with this is to let students discover that even a well-coordinated community can experience crashes because of scarcity.

Finally, in order to address the “profit and competition” pattern and the “fixed human nature” pattern, we have developed a set of activities within and around the PartSim experiences, in which students run collective self-defined “experiments” in and on their commons. While the groups spontaneously thought of some of their actions as providing information about their environment even our first-year implementation, our subsequent design refinements have amplified this tendency. Data from these experiments have allowed student groups to make better decisions about how many cows should be allowed to graze, and how many people should repair fences or fertilize the grazing ground each week. In addition to serving the more general learning goal of using computer simulations and data to make decisions, our hope with these activities is that students can experience both the challenges and benefits of working together to manage shared resources and to gauge their success in doing so.

An important limitation of our study is that our sample of students comes from a predominantly white, private, suburban, parochial high school from which 99% of graduates go to college. Importantly, this study focuses on reasoning about sharing, and we believe that these students who often come from very high-resource homes will have a particular set of experiences of sharing as a result of their

upbringing, not representative of the high-school aged population as a whole. To address this limitation in the makeup of our population, we hope soon to implement our unit in other socio-economic settings.

In analysing student response data, we identified substantial variation not only across students but also within students across the two cases that students reasoned about. However, it is important to keep in mind that the two cases are quite different, exhibiting distinctive features of the natural resource systems that communities share. In order to tease apart (a) how differences in students' responses to the cases stem from genuine differences or changes in *thinking* about the cases on the one hand, from (b) substantive differences between the social-ecological systems in the cases on the other hand, we are varying the order in which students experience each of the two cases.

Finally, we believe that participatory simulations offer a particularly powerful approach to learning about common-pool resource sharing dilemmas in a classroom setting. PartSims give students the dual perspective of individual and group in experiencing social-ecological dilemmas and allow them to engage in deliberation about resource systems sharing. Given the importance of this topic to the survival of our species and planet, we feel that collective resource systems sharing and maintenance is an important area of focus for education research and design. In this paper, we have presented what we see as early educational research on this topic.

ACKNOWLEDGMENTS

We are grateful for the support from Loyola Academy in Wilmette, IL, USA and all students who participated. The authors gratefully acknowledge the support by NSF Grant #1441552. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

- Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilensky, U. (2017). All roads lead to computing: Making, participatory simulations, and social computing as pathways to computer science. *IEEE Transactions on Education*, 60(1), 59-66.
- Brady, C., Holbert, N., Soylu, F., Novak, M., & Wilensky, U. (2015). Sandboxes for model-based inquiry. *Journal of Science Education and Technology*, 24(2-3), 265-286.
- Brondizio, E. S., Ostrom, E., & Young, O. R. (2009). Connectivity and the Governance of Multilevel Social-Ecological Systems: The Role of Social Capital. *Annual Review of Environment and Resources*, 34(1), 253–278. <https://doi.org/10.1146/annurev.environ.020708.100707>
- Cobb, P., Confrey, J., Lehrer, R., Schauble, L., & others. (2003). Design experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- Gilligan, J. M., Brady, C., Camp, J. V., Nay, J. J., & Sengupta, P. (2015, December). Participatory simulations of urban flooding for learning and decision support. In *Proceedings of the 2015 Winter Simulation Conference* (pp. 3174-3175). IEEE Press.
- Guo, Y., & Wilensky, U. (2016). Learning About Complex Systems with the BeeSmart Participatory Simulation. In *Proceedings of Constructionism 2016*. Bangkok, Thailand. February 1-5.
- Hardin, G. (1968). The tragedy of the commons. *Science*, 162(3859), 1243–1248.
- Klopfer, E., Yoon, S., & Perry, J. (2005). Using palm technology in participatory simulations of complex systems: A new take on ubiquitous and accessible mobile computing. *Journal of Science Education and Technology*, 14(3), 285–297.
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis : an expanded sourcebook*. Thousand Oaks: Sage Publications.
- NCSS. (2013). *The college, career, and civic life (C3) framework for social studies state standards: Guidance for enhancing the rigor of K–12 civics, economics, geography, and history*. Author Silver Spring, MD.

- NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. Washington, DC: The National Academies Press.
- Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. Cambridge university press.
- Ostrom, E. (2007). A diagnostic approach for going beyond panaceas. *Proceedings of the National Academy of Sciences*, 104(39), 15181–15187.
- The Drama of the Commons*. (2002). Washington, D.C.: National Academies Press. Retrieved from <http://www.nap.edu/catalog/10287>
- Wilensky, U. (1999). NetLogo: Center for connected learning and computer-based modeling. *Northwestern University, Evanston, IL*.
- Wilensky, U., & Jacobson, R. (2014). Complex Systems in the Learning Sciences. In *The Cambridge handbook of the learning sciences* (Vols. 1–2). Cambridge, UK: Cambridge University Press.
- Wilensky, U., & Stroup, W. (1999a). HubNet. *Center for Connected Learning and Computer-Based Modeling, Northwestern University: Evanston, IL USA*, < [Http://Ccl.northwestern.edu/Ps](http://Ccl.northwestern.edu/Ps).
- Wilensky, U., & Stroup, W. (1999b). Learning through participatory simulations: Network-based design for systems learning in classrooms. In *Proceedings of the 1999 conference on Computer support for collaborative learning* (p. 80). International Society of the Learning Sciences.

Urban Planning-in-Pieces: A Computational Approach to Understanding Conceptual Change and Causal Reasoning about Urban Planning

Arthur Hjorth, arthur.hjorth@u.northwestern.edu

Center for Connected Learning and Computational Modeling, Northwestern University, USA

Uri Wilensky, uri@northwestern.edu

Center for Connected Learning and Computational Modeling, Northwestern University, USA

Abstract

In this paper, we present a computer-simulation-based classroom unit designed to improve students' *causal* reasoning about the effects of urban planning. Based on data collected in three separate implementations of this unit, we use Association Rule Mining (ARM) to assist in Knowledge Analysis (KA) of students' responses to pre- and post-questions. We first define *causal-nodes* as a construct, and then qualitatively identify distinct causal-nodes in students' responses. We then validate these nodes quantitatively within and between students' responses. Finally, we compare changes in the association rules between these causal-nodes between pre- and post-responses in order to find changes in students' explanations. This paper makes two distinct contributions: first, it shows a productive use of an existing computational method to aid in the analysis of conceptual change. Second, it contributes towards better understanding how to design for and analyse causal reasoning in social science education.

Keywords

knowledge analysis; netlogo; social studies; complex systems; causal reasoning; computational analysis; computational methods

Causal Reasoning in the Social Studies

The College, Career, and Civic Life (C3) Framework for Social Studies State Standards (NCSS, 2013) presents a new framework for K-12 social studies, emphasizing 'student explanations' and 'complex causal reasoning' in Social Studies. However, in contrast to a similar move in science education (NGSS Lead States, 2013), design and implementation of *social studies* curricular activities that focus on causal reasoning are presently understudied. This leaves two gaps that we think are important in the literature: first, how do we design to encourage and strengthen causal reasoning in social studies; and second, how do we study this thinking in a social studies context?

In this paper we hope to contribute to these currently understudied goal by presenting a computationally-assisted Knowledge Analysis (diSessa & Sherin, 2015) of students' causal-explanations in response to pre- and post-questions as part of a unit of our own design, in which students used a computer simulation to learn about urban planning. The simulation and accompanying activities were designed to help students think *causally* about the social impact of urban planning decisions. Analysing students pre- and post-question responses, we identify a set of *causal-nodes* and use Association Rule Mining (ARM) (Agrawal, Imieliński, & Swami, 1993) to look at how, at the classroom-level, these nodes are reorganized between pre- and post-responses. Based on our findings and discussion, we claim that ARM is a particularly well-suited approach to helping us in the analysis of student reasoning with a piecemeal view of knowledge and learning, and that it potentially can help us scale up Knowledge Analysis to larger dataset than it has currently been applied to.

Research Questions

In this paper, we address the following questions,

1. How can we design activities that improve students complex causal reasoning about social issues?
2. What are the *constituent parts* of students' causal-explanations when responding to questions about city planning?
3. How are these parts assembled into larger explanation-structures, and how can ARM help us make sense of change in these explanatory structures?

Why Association Rule Mining for Knowledge Analysis?

Knowledge Analysis takes a piecemeal approach to understanding the mental representations of knowledge, and views reasoning as the *assembly* of these pieces in-the-moment (diSessa, 2002; diSessa & Sherin, 2015; diSessa & Sherin, 1998; Sherin, 2006; Sherin, Krakowski, & Lee, 2012). The purpose of KA is to provide an analytical space for better understanding knowledge and learning: In this view, briefly, learning is the acquisition of new knowledge pieces into a learners' repertoire, and/or changes in the assembled constellations of these pieces. An important feature of a single knowledge piece, is that it can be combined with other knowledge pieces into essentially different, larger structures, and that the meaning of a network of knowledge pieces is an emergent property of these manifold structures.

Likely due to the fine-grained nature of KA, most KA-oriented studies are based on somewhat small sample sizes. Recent work has moved to include more computational methods in the Learning Sciences (Martin & Sherin, 2013; Sherin, 2013). In contrast to Educational Data Mining's focus on quantitatively *assessing* student thinking, this work has primarily focused on using computational methods to qualitatively better understand the process of learning or the constituent parts of conceptual change (Berland, Baker, & Blikstein, 2014; Blikstein, 2011), and on utilizing the power of computation to scale up the size of data. In this paper, we propose to use Association Rule Mining for a similar purpose: as a method for assisting us in a Knowledge Analysis, both as an approach to validating the causal-nodes that we identify in students, and as a way of scaling up KA to help us look at learning at the classroom-level.

Association Rule Mining (Agrawal et al., 1993) is a method for understanding co-occurrences between elements in a set of data. More specifically, an ARM-analysis takes a set of 'transactions' that each include some elements, and then calculates, across all these transactions, how well the presence of one particular element in a transaction predicts the presence of another element in a transaction. We believe that ARM's focus on looking at the co-occurrence of individual items makes it particularly well-suited for helping us in analyses of students thinking with a piecemeal view of knowledge: If we view student whole explanations as transactions, and knowledge-pieces as the individual elements in these transactions, we can use ARM to better understand the changes in students' reasoning both at the individual- and at the classroom-level by calculating if and how these pieces are reassembled over time.

Previous use of simulation in urban planning education

Simulations have been used for SimCity in particular has received attention since the early-1990s as a teaching tool in formal education that involves urban planning or thinking about cities (Adams, 1998; Dorn, 1989; Gaber, 2007; Kolson, 1996; Pahl, 1991). Early uses of SimCity focused on two different aspects. The first sought to make a general case for using commercial computer simulations games in education (Shaffer, Squire, Halverson, & Gee, 2004; Shaffer, 2006; Shaffer, 2006; Squire, 2003) and sought to show how the well-designed and engaging commercial games would make school fun for children. The other focused on simulation literacy (Gee, 2003, 2007; Turkle, 1997), and its specific interest was in how (or if) children engaged with the *ideological* assumptions programmed into the city. More recent work has focused using SimCity in formal *urban planning* education at the college or graduate level: how to align SimCity with formal curriculum and use it as an introductory tools at the college and graduate level (Bereitschaft, 2016; Devisch, 2008); how SimCity can help urban planning students engage with their own creativity (Kim & Shin, 2016); or as a reflective tool to improve on urban planning pedagogy itself (Kim & Shin, 2016).

However, while simulation of *causal* mechanisms are at the very core of the representational power of simulation games (Frasca, 2001, 2003; Squire, 2003), none of the reviewed work has focused specifically on causal reasoning. Thus, while simulations in general and SimCity in particular have seen use in education, this work does not address the specific need created by NCSS (2013) for a conceptual framework for analysing and measuring learners' complex *causal* reasoning. One aim and contribution of this paper is to fill out this gap in the literature and explore how to design simulations to help students improve on their complex causal reasoning about complex social phenomena. We previously reported on qualitative findings from pilot data from this study (Hjorth & Wilensky, 2014a, 2014b; Hjorth & Krist, 2016), but have now collected enough data to take quantitative approaches like this as well.

The Design & Study

The data in this paper come from a unit that we designed on Urban Development and Regional Planning and which we implemented during two quarters at the undergraduate level at a mid-sized, private research university in a metropolitan area in the American Midwest. The course was called, 'Introduction to Social Policy' and is required for Social Policy majors. Students were given course credit equivalent to one course essay for participating. In this paper, we focus on students' pre- and post-responses to a question about how urban planning affects the distribution of commute times for different income groups. Over the span of three implementations, a total of 60 students consented. We sent out the pre-questionnaire 10 days before class in which we used the model, and we sent out the post-questionnaire 10 days after class had finished, and students typically responded within two days. Not all students responded to our questions, and we had some attrition between pre- and post-, and *during* students' responding to questionnaires due to technical problems. In the end, we had 41 students who responded to both the pre- and the post-question.

Our Design and Activity

We designed a unit that ran over the span of two class periods in which students use a NetLogo (Wilensky, 1999) simulation to build cities. Before the simulation activity, the professor (who was not part of the research team) first led a 45-minute discussion about *why people live where they live*. The purpose of this discussion was to cue students' causal reasoning about the role of human decisions, and how these decisions play a causal role in the emergence of economically segregated neighborhoods. Students then worked together in groups of three over a period of about an hour and a half, using the simulation.

The simulation and activity were designed to help students iteratively improve their causal-understanding. We did this by letting students iteratively *articulate* a causal-explanation, *test* it, and potentially *revise* it. This was achieved by designing an iterative, four-phase activity:

1. The Design Phase

The primary purpose of the design phase was to prompt students to reason causally about how to design a city that meets a set of measurable policy goals. Students were first asked to write a set goals for their city. This included specifying which one of three policy outcomes – commute times, local neighbourhood school funding, and access to parks & leisure areas – to focus on, and setting measurable goals for that outcome. Examples include, "We want everybody to have less than 30-minute commute time", or "We want the poorest 20% to have as much funding for schools as the wealthiest 20%". They would then be asked to describe how they would achieve these goals. Students responded along the lines of, "We will put highways everywhere so there are enough roads for everybody", or "We will put parks all over the city to make it attractive for wealthy people to live anywhere so their property taxes are spread out across the city." (Schools are primarily funded through property taxes in most of the US.)

2. The Implementation Phase

During the Implantation Phase, students built their cities inside the simulation. The simulation is designed to let students do this in various ways: Students designate zoning in the city, specifying the

density of dwellings, and making certain areas more or less desirable and relatedly more or less expensive to live in. They can also put parks in the city, or build railroads or highways for people to commute on. Finally, they designate certain parts of the city as 'Industrial areas' where jobs will be located.

3. The Growing Phase

The purpose of the Growing Phase was to let students see the *dynamic* effects of their design decisions. During the Growing Phase, the computer model simulates and visualizes how (AI-based) computer-agents move into the city that students built. These agents have different income levels and different job locations and make decisions about where to live based on affordability, desirability, and distance to their job. When agents move into an area, their income is reflected in the house prices of nearby areas, so if a wealthy person moves into a neighbourhood, house prices go up, and vice versa. Agents also use the roads near them to go to work, and the more agents use a road, the more congested it gets, making nearby areas less attractive.

4. The Data Analysis Phase

The purpose of the Data Analysis phase was two-fold: first, it was for students to use data as a means of assessing the success of their city; and second, for students to potentially revise their causal-understanding of the model in the cases when their cities did not meet their policy goals. There are two different ways in which data can be visualized in the model: one is spatial, and the other is with bar charts. They provide different perspectives on the same questions, but it is often necessary to look at both in order to really make sense of how the city evolved. Bar charts showed how each income decile was affected by each of the three policy outcomes measures, and the map helped students visualize the geographic distribution of how people were impacted by the policy outcomes.

Data & Analysis

The student responses that we focus on in this paper were all in response to the question,

“Can you explain why a wealthy person’s income might make their commute time longer than a poor person’s?”

A conventional explanation, consistent with real-world data, could include a variety of factors and sound something along the lines of, “In American cities, higher-income people often live in suburbs, because they can afford to buy houses there, and because they can afford to own cars that allow them to commute between their workplace in the city and the suburbs that often have no public transit options. They choose to do so because they want to live in places that they perceive as safe, and have well-funded public schools. High paying jobs are often located in the downtown areas of cities, and consequently high-income people must make the commute in and out, often on congested roads due to the number of people who also commute in and out at the same time”. While it is true that some high-income people live closer to their jobs than some low-income people – in part because they can afford more freedom to choose where to live, and where to work – the result is nonetheless that higher income people typically have longer commutes, but with a large spread in a bimodal distribution. We were curious about how students would reason about it exactly because it contrasts many people’s initial assumption that a higher income leads to more desirable outcomes by all measures.

While some KA-approaches have strict, conceptual selection criteria for the pieces they identify (diSessa, 1993), in this study, we take a pragmatic and somewhat promiscuous approach to identifying the individual knowledge-pieces. Rather than looking for a particular grain size or ontogenetic origin, we used ‘causality’ as a sensitizing concept (Miles & Huberman, 1994) when identifying knowledge-pieces in students’ explanations, and looked for parts of their explanation that we could put “because... “ in front of. Because our knowledge-pieces relate to students’ reasoning about causality, and inspired by Sherin, Krakowski and Lee’s (Sherin et al., 2012) *node-mode* approach towards a more permissive inclusion of students’ knowledge-pieces, in the following, we will refer to them as ‘causal-nodes’. Further, we will refer to the process of *using* a causal-node when constructing an explanation as

'activating' that causal-node, and we call the process of combining causal-nodes into larger explanations as 'co-activating' those causal-nodes.

Causal-Nodes and Student Response Examples

To give the reader a sense of what causal-nodes in our data look like, we provide some examples of student responses and connect them to our causal-nodes. The student responses are always reproduced in full, and as the student wrote them. Across all 82 responses (41 pre- and post-responses) our first round of coding identified 21 different causal-nodes. We iteratively condensed this set twice, eliminating similar codes, eventually arriving at a total of 9 different causal-nodes. Table 1 provides a description of all nine causal-nodes, grouped by what we think of as three interesting types: the first relates to the *geographic location* of people and their jobs. The next group relates to how a person's income affects their *actual commute* – either by influencing their available modes of transportation, or things that affect their commute speed. The final type relates to *the wants or desires of people*, and how having money better allows high income people to fulfill them.

As Table 1 (next page) shows, the most frequent causal-nodes were the ones that deal with the geographic location of people and jobs. Consider the following response,

A wealthy person has greater freedom to choose where to live and often wealthier neighborhoods are in suburbs away from urban areas. (S2-post)

In this response, we see two different causal-nodes: first, that money gives people more choice, and second, that suburban areas in which wealthy people choose to live are geographically far from urban areas. However, the response does not explicate exactly how these two causal-nodes end up resulting in longer commute times for higher income people. In contrast, consider this elaborate student response,

Often wealthy people live in a suburb outside of the city because they can afford a house out there and the schools are often better. They also often work in the city at a job that is paying them enough to be able to afford to live in the suburbs. Another big thing is that more wealthy people can afford a car or are able to pay for the train everyday so that they can live far away from their work. (S7-post)

This response contains the same two causal-nodes as the previous response: people with more money have more choice, and they choose to live in suburbs. However, we see additional causal-nodes in this response: first, the response explicitly states that their jobs can be in the city; second, that this causes them to have to travel a larger geographical distance; and finally, that owning a car is expensive and that wealthy people can afford to own one (or afford to take commuter trains.) These were somewhat typical responses, but they show how the different constellations of casual-nodes can result in different responses, or in similar kinds of responses with variations in what students focus on. We also saw responses that focused more on practical issues relating to the process of commuting,

A wealthy person probably commutes by driving his or her own car. A person in a car is subject to traffic from stoplights and other vehicles. Trains, on the other hand, can travel at a quicker rate and also don't have to stop at lights or wait for other vehicles. (S37-post)

This response does not activate any of the geographic location-related nodes, but focuses purely on the commute-related ones. However, we also saw some responses that seemed to activate different causal-nodes that included perceived differences between *why* wealthy people choose to do what they do, even in the face of a longer commute time,

If they choose to live in a suburb or nice area that is farther for work because of the neighborhood or home or school system. They also more likely have ways to commute comfortably and efficiently. (S23-post)

Students that reasoned about *why* wealthy people make the choices that they do often focused on their perceived better school systems, more green areas or nicer houses in suburbs. But we also saw a few responses that essentialized different characteristics or desires in low- and high-income people. For instance, in this response,

They might have more time to spend on leisure, and might not be in a rush. Since they make enough money, they don't have to work as much. (S10-pre)

Node_ID	Causal-Nodes ('Because...')	Pre-frequency	Post-frequency	Within-person stability	Combinability
Group 1: Geographic/Spatial Location					
0	High income people live in suburbs / low income people live in the cities	0.73	0.83	1.13	8 / 8
1	Jobs are located in cities, not in suburbs	0.46	0.49	1.19	8 / 8
2	High income people may live further from workplaces / Low income people might live closer to workplaces	0.63	0.59	1.12	8 / 8
Group 2: Commute-Related					
3	Buying/owning a car is expensive	0.1	0.1	5.12	7 / 8
4	High income people more likely to own and commute by car	0.1	0.05	5.12	5 / 8
5	Expressways/highways can be congested, or driving can be slow	0.34	0.2	2.56	7 / 8
Group 3: Desires or Possibilities relating to Income					
6	High income people have more choice / low-income people have less choice	0.59	0.59	1.14	8 / 8
7	Wealthy people care less about commute time / are willing to commute longer	0.12	0.07	2.73	6 / 8
8	People want to live in safe areas or places with more space or green areas or better schools or nicer houses	0.39	0.34	1.1	7 / 8

Table 1. Causal-nodes and descriptive statistics

the student seems to construct an explanation around an assumption that higher income people somehow don't feel the same sense of urgency, or that they have a different set of preferences around their choices when it comes to commute times.

As the examples above hopefully illustrate, we saw a large variety in the types of student responses, and in the ways in which we see them activate and co-activate various causal-nodes. Some student responses only included causal-nodes from one or two of the three causal-node groups, while others mixed them across groups.

Validation: Are these Knowledge-Pieces?

KA views knowledge as constructed in the moment through the assembly of knowledge pieces. However, it does of course not view this process as random. Rather, taking a Knowledge Analysis approach to making sense of student reasoning, we would expect to see three different properties relating to the consistency of the nodes that we identify:

The pieces should be found somewhat frequently in a specific population that reasons about a given domain or question (i.e. 'frequency *between people*'). This would indicate that these are more general thinking-bits, and not only parts of one person's idiosyncratic way of thinking.

The pieces should be generatively combinable, meaning they should be combinable into different kinds of explanations (i.e. *frequency between different explanations*). This would indicate that they are truly 'pieces', and not in themselves larger explanations. Finally,

Even though individuals might acquire new nodes and/or reorganize their existing ones, we would expect there to be *some degree of stability* in the knowledge-nodes that students activate when responding to a similar question in a short timeframe (i.e. 'frequency *within people*'). This would indicate that these are parts of somewhat stable knowledge structures, and not completely randomly selected when the student is prompted to answer a question.

Validating Frequency Between People

This criterion is straight forward to validate. We calculated the frequency of each node in pre- and post-responses. The results can be seen in **Error! Reference source not found.** in respectively pre- and post-frequency. We see that even the least frequent of our codes appear in at least 10% of responses in either pre- or post. While the exact cut off for this rule is contestable, we believe that seeing the activation of a causal-node at some point in time across 10% of responses seems like a reasonable number.

Validating Frequency between Different Explanations

There are two different ways in which this can be validated. First, a very simple quantitative statistic showing with how many of the other 8 causal-nodes that we see each node co-occur. In the 'combinability'-column in **Error! Reference source not found.**, we show how many *other* causal-nodes we see each causal-node co-occur with across all responses. Even the least frequent causal-nodes (3 and 4) are used respectively with 7 and 5 of the other 8 casual-nodes, suggesting that these nodes can be mixed and matched in many different ways. Second, as we showed in the previous section, causal-nodes were combined into different constellations of explanations that changed the function of the of the individual causal-node in the larger reasoning structure. In other words, we see this combinability both quantitatively and qualitatively.

Validating Frequency Within People

Finally, we expect there to be some stability in the causal-nodes individuals activate in their pre- and post-responses. What we should address then is, does activating a causal-node in a pre-response *better* predict that a person also activates it in their post-response than we would expect to see if people randomly activated causal-nodes in their responses. For each of the causal-nodes, we calculated the conditional probability that people who activated it in their pre-response also activated it in their post-response and divided by the frequency of that causal-node in post-responses. If this ratio is greater than 1, we see a higher than expected frequency amongst people who also activated the node in their pre-response. As can be seen in the "stability" column, all causal-nodes had a higher than expected degree of stability between pre- and post-responses.

Findings: Association Rule Changes

Now that we have identified students' causal-nodes and hopefully made a convincing argument that these are, indeed, knowledge-pieces of some sort, we can run an Association Rule Mining on our data. The primary output of an ARM are so-called *association-rules*. They take the form of, "if a student activated causal-nodes X and Y, we observed that they *also* activated causal-nodes A and B with a *confidence* of P, a *support* of S, and a *lift* of L". ARM calculates the association-rules for all possible combinations across all responses. Since we have 10 different codes, and each of them can either be present or not, we end up with a total of $2^{10} = 1,024$ combinations – too many rules to read through in any meaningful way. ARM assists us in navigating this large analytical space by providing three metrics, *confidence*, *support*, and *lift*, that each help identify interesting and important rules: *Confidence*

calculates how well the presence of one set of causal-nodes predicts another set of rules, or more formally: the conditional probability that a set of causal-nodes are activated, given the activation of another set of causal-nodes. However, confidence does not consider the aggregate frequencies of causal-nodes, and thus it often overestimates the confidence with which a less frequent causal-node predicts a more frequent one and vice versa.

Rule		Lift			Confidence			Support		
		Pre	Post	Delta	Pre	Post	Delta	Pre	Post	Delta
R1	1->2&6	0.84	1.17	0.3	0.37	0.40	0.03	0.17	0.20	0.02
R2	1->0&8	1.21	1.37	0.15	0.47	0.40	-0.07	0.22	0.20	-0.02

Table 2. Two Examples of Association Rules (bolded text indicates why they were chosen)

Support and *lift* help us filter out rules in two related ways: *support* simply calculates how often, across the dataset, we see a particular combination of causal-nodes. *Lift* enhances the *confidence* measurement by calculating the ratio between *observed* confidence and the *expected* confidence, given the independent probabilities of each of the two sets of codes. (This is the same approach we used for calculating the within-person stability in the previous section.) Consequently, as we are interested in looking at changes in how students co-activate causal-nodes, the immediately most important measurement is change in *lift* between pre- and post-responses, but looking at lift isolated in pre- or in post- can also help us understand what kinds of constellations of causal-nodes students bring to the unit, and which ones they leave with. We ran an ARM for pre- and post-responses separately, and then calculated the *changes* to *confidence*, *support* and *lift* between pre- and post-rules. To find interesting and frequent rules, we filtered out rules that had less than .15 support, and included only combinations that we observed in at least 7 out of 41 responses in *both* pre- and post-responses. A full account of all interesting rules is outside the scope of this paper, but in the following, we will give two examples of interesting rules, explain why we think they are interesting and why we chose them, and discuss what they tell us about changes in students' thinking at the classroom level.

R1: Connecting Job Location with Choice and Relative Distances

R1 in Table 2 shows that students who reasoned that jobs are located in cities and not in suburbs were much more likely in the post-response than those who said so in the pre-response, to also say that high income people may live further from workplaces, *and* that high-income people have more choice. We chose this rule because it had the highest change in delta in our ARM. Of course, even when *lift* takes into account the expected frequencies, this change *could* have happened simply because fewer people activated any of the three causal-nodes in their post-responses, and those that did could be the ones spuriously "pulling up" the association rule. However, Table 1 tells us that the frequencies of all three codes are fairly stable from pre- to post- at the classroom level, and in Table 2 we even see a modest increase in support for R1, showing that in absolute numbers, more people co-activated the three causal-nodes in their pre-responses. To us, this indicates that R1 points to a robust change in students' thinking at the classroom level towards connecting the *location of typically higher-paying jobs* with the *choice* and *relative commute distances* of higher income.

Qualitatively, this is a particularly interesting rule to us, because it gets at the part of the question that most people find counter-intuitive: that high-income people often have longer commutes than low-income people. Indeed, one of the reasons we designed the simulation activity was to let students change the infrastructure and the zoning of the city – including the relative position of residential zones and workplaces – to see how common city design patterns (e.g. dense city centres, "green" suburbs far away) lead to this distribution of commute times. While any firm conclusions about causality in this change in thinking would rely on a closer analysis of students' activities during the simulation unit, we speculate that the focus on *placement of zones* in the model helped make this aspect of the phenomenon more salient to students.

Rule 2: Connecting Job Location & And the Desirability of Suburban Life

R2 speaks to the relationship between causal-nodes 1, and 0 and 8. We chose this rule for two reasons: first, R2 has the highest *lift* in post-responses that we identified in any rule, and second, it is interesting because while we see a positive change in lift, we see a drop in confidence between pre- and post-. R2 shows that students who said that jobs are located in cities and not in suburbs were more likely *than expected* to also say that wealthy people live in suburbs *and* that people want to live in places with green areas or good schools. However, as mentioned, we see an absolute drop in confidence, or predictive power, for this rule between pre- and post-. The drop in confidence on its own could mean that more people are activating causal-node 1 in the post-responses without an increase in the co-activation of causal-nodes 0 and 8 in post-, or that fewer people are co-activating causal-nodes 0 and 8 in the post-responses without a corresponding drop in the activation of causal-node 1. Table 1 shows that we do see an increase in the activation of causal-node 1 and a drop in causal node 8 between pre- and post-. However, we see an increase in causal node 0. To us, this is a good example of why looking at lift tells a more nuanced story, and why it is sometimes easier to read association rules backwards when trying to make sense of them: While slightly fewer people co-activated all three causal-nodes in the post-response, the strongest predictor of whether someone did was whether they activated causal-node 1. In other words, this shows a *convergence* across the classroom on the inclusion of causal-node 1 by those who also co-activated causal-nodes 0 and 8.

Qualitatively, this rule is interesting to us, because it shows how students reason not just about the relative position of jobs and residential areas or about high-income people having more choices. It also shows how they perform a kind of meta-reasoning or perspective-taking: reasoning about how *other people reason* about where to live, and how students then activate this reasoning with the rest of causal-nodes. While we do see a strengthening of this particular association rule, we see an overall drop in the activation of causal-node 8. We think perspective-taking is important when reasoning about policy outcomes, and had hoped that our design would have encouraged more of this thinking. We speculate that this might be due to how the underlying logic of the AI agents in the simulation activity was hidden from students, and will in future implementations explore how we can forefront the AI and make this aspect more visible to students.

Discussion, Limitations, and Conclusion

Using Association Rule Mining helped us better identify interesting patterns in changes in students' assembly of causal-nodes into larger explanatory structures, and gave us both some statistics and a vocabulary for measuring and discussing which of these changes were interesting and significant. We found that students' responses seemed converge around particular co-activations, and we speculated how the collaborative simulation activity might have influenced their thinking, and how we could improve on the design. We hope to have provided evidence for our assertion that Association Rule Mining can be a powerful addition to the qualitative researchers' toolbox when taking a Knowledge Analysis-inspired view of knowledge. In particular, we hope to have shown that ARM can be used in combination with manual qualitative coding to both validate the knowledge-pieces identified at the *level of individual students*, and show changes in the assembly at the *classroom level*.

In contrast to KA's focus on conceptual change at the level of individuals, we only looked at within-student changes when we validated the stability of individual causal-nodes. In future work, we hope to use ARM to first identify important changes at the classroom level, and then use this as a starting point for a more in-depth analysis of the reorganization of knowledge-pieces at the level of individual students. In additional contrast to KA, our knowledge pieces are somewhat less fine-grained, and as we mentioned previously, we took a very permissive approach to coding students' responses. We hope in the future to apply ARM to a both more fine-grained, and more conceptually coherently uniform set of knowledge-pieces.

Because this use of ARM is new, we have limited understanding of exactly how to interpret the stability and change in changes in thinking at the classroom level. Our baseline for comparing changes is always the expected outcome, i.e. the general frequency of a set of causal-nodes in the post-responses. However, we hope to do similar kinds of analyses on more data from this and other reasoning tasks

over longer periods of time, and start work on better understanding and measuring the change and stability in knowledge structures, and potentially develop more generally applicable measures.

An important limitation of this study is that students were all attending a private mid-western university, and almost all students reported to have grown up in suburbs. Consequently, the causal-nodes that we identified should be considered expressions of a particular, and fairly limited experience of the world. We hope to expand on this by collecting similar data from other socio-economic or geographic groups.

We do not wish to claim that an ARM can stand on its own as an analysis of student reasoning. But we believe that it provides a tool for measuring and discussing changes in thinking at the classroom level while still anchoring the unit of analysis in a KA-approach to knowledge and thinking that respects the multitude ways in which students can assemble their knowledge. Consequently, we hope that it will find an appropriate place in the computational methods toolbelt of the Learning Sciences.

References

- Adams, P. C. (1998). Teaching and learning with SimCity 2000. *The Journal of Geography*, 97(2), 47.
- Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD Record* (Vol. 22, pp. 207–216). ACM.
- Bereitschaft, B. (2016). Gods of the City? Reflecting on City Building Games as an Early Introduction to Urban Systems. *Journal of Geography*, 115(2), 51–60. <https://doi.org/10.1080/00221341.2015.1070366>
- Berland, M., Baker, R. S., & Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning*, 19(1–2), 205–220.
- Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge* (pp. 110–116). ACM.
- Devisch, O. (2008). Should Planners Start Playing Computer Games? Arguments from SimCity and Second Life. *Planning Theory & Practice*, 9(2), 209–226. <https://doi.org/10.1080/14649350802042231>
- diSessa, A. (1993). Toward an epistemology of physics. *Cognition and Instruction*, 10(2–3), 105–225.
- diSessa, A. (2002). Why “conceptual ecology” is a good idea. In *Reconsidering conceptual change: Issues in theory and practice* (pp. 28–60). Springer.
- diSessa, A., & Sherin, B. (2015). Knowledge Analysis. *Knowledge and Interaction: A Synthetic Agenda for the Learning Sciences*.
- diSessa, A., & Sherin, B. (1998). What changes in conceptual change? *International Journal of Science Education*, 20(10), 1155–1191.
- Dorn, D. S. (1989). Simulation Games: One More Tool on the Pedagogical Shelf. *Teaching Sociology*, 17(1), 1–18.
- Frasca, G. (2001). *Videogames of the Oopressed - Videogames as a Means for Critical Thinking and Debate* (Vol. 2001). Master's Thesis, Georgia Institute of Technology.
- Frasca, G. (2003). *The Video Game Theory Reader*. Routledge. Retrieved from <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0415965799>
- Gaber, J. (2007). Simulating planning - SimCity as a pedagogical tool. *Journal of Planning Education and Research*, 27(2), 113–121.
- Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment*, 1(1), 20.
- Gee, J. P. (2007). *Good Video Games and Good Learning* (1st ed.). Peter Lang Publishing.

Hjorth, A., & Wilensky, U. (2014). Re-grow Your City: A NetLogo Curriculum Unit on Regional Development. In *Proceedings of International Conference of the Learning Sciences, ICLS* (Vol. 3, pp. 1553–1554). International Society of the Learning Sciences.

Hjorth, A., & Krist, C. (2016). Unpacking Social Factors in Mechanistic Reasoning (Or, Why a Wealthy Person is Not Exactly Like a Grey Squirrel). Singapore: International Society of the Learning Sciences.

Hjorth, A., & Wilensky, U. (2014). Redesigning Your City-A Constructionist Environment for Urban Planning Education. *Informatics in Education*, 13(2), 197.

Kim, M., & Shin, J. (2016). The Pedagogical Benefits of *SimCity* in Urban Geography Education. *Journal of Geography*, 115(2), 39–50. <https://doi.org/10.1080/00221341.2015.1061585>

Kolson, K. (1996). The Politics of SimCity. *PS: Political Science and Politics*, 1996(1), 43–46.

Martin, T., & Sherin, B. (2013). Learning Analytics and Computational Techniques for Detecting and Evaluating Patterns in Learning: An Introduction to the Special Issue. *Journal of the Learning Sciences*, 22(4), 511–520. <https://doi.org/10.1080/10508406.2013.840466>

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis : an expanded sourcebook*. Thousand Oaks: Sage Publications.

NCSS. (2013). *The college, career, and civic life (C3) framework for social studies state standards: Guidance for enhancing the rigor of K–12 civics, economics, geography, and history*. Author Silver Spring, MD.

NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. Washington, DC: The National Academies Press.

Pahl, J. (1991). Finally, a Good Way to Teach City Government! A Review of the Computer Simulation Game “SimCity.” *The Social Studies*, 82(4), 165–166.

Shaffer, D., Squire, K., Halverson, R., & Gee, J. (2004). Video Games and the Future of Learning. University of Wisconsin-Madison. Retrieved from <http://www.discoverproject.net/italy/images/gappspaper1.pdf>

Shaffer, D. (2006). *How Computer Games Help Children Learn* (First Edition). Palgrave Macmillan.

Shaffer, D. (2006). Epistemic frames for epistemic games. *Computers & Education*, 46(3), 223–234.

Sherin, B. (2006). Common sense clarified: The role of intuitive knowledge in physics problem solving. *Journal of Research in Science Teaching*, 43(6), 535–555.

Sherin, B. (2013). A Computational Study of Commonsense Science: An Exploration in the Automated Analysis of Clinical Interview Data. *Journal of the Learning Sciences*, 22(4), 600–638. <https://doi.org/10.1080/10508406.2013.836654>

Sherin, B., Krakowski, M., & Lee, V. R. (2012). Some assembly required: How scientific explanations are constructed during clinical interviews. *Journal of Research in Science Teaching*, 49(2), 166–198.

Squire, K. (2003). Video Games in Education. *International Journal of Intelligent Simulations and Gaming*, 2003(2), 1.

Turkle, S. (1997). Seeing Through Computers: Education in a Culture of Simulation. *The American Prospect*, March-April 1997.

Wilensky, U. (1999). NetLogo: Center for connected learning and computer-based modeling. *Northwestern University, Evanston, IL*.

How High is the Ceiling? Applying Core Concepts of Block-based Languages to Extend Programming Environments

Sven Jatzlau, sven.jatzlau@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Ralf Romeike, ralf.romeike@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Since the emergence of block-based visual programming languages, they have been developed and improved to become increasingly accessible, intuitive, and easy to use. Over the course of this evolution, both uncommon and entirely new language concepts have been introduced, such as the cloning of objects, or the nesting of sprites. This paper provides a selection of core concepts and describes a categorization model. It is proposed that the concepts found in block-based languages are the reason they lend themselves to constructionist learning approaches. To illustrate this point, the fundamental computer science concept of image nesting will be selected, its background and origin explained, and its incarnation in current block-based languages outlined. Following this, the constructionist task of modifying a programming environment will be implemented. Using the aforementioned concept proves that even high-level solutions according to the “high ceiling”-principle can be implemented using the basic concepts of block-based languages.

Keywords

visual programming languages; Snap; Scratch; GP; block-based languages; language concepts; nesting; composite objects

Context

Introduction to block-based languages

In 2006, the programming language and environment Scratch made its debut. Developed by the Lifelong Kindergarten Group at MIT, it represented a new approach to introducing programming to learners: based on the ideas of “low floor, wide walls, high ceilings” by Seymour Papert (Resnick, et al., 2009), its popularity grew rapidly. At the time of writing, some 32 million projects, guides, animations, and games have been shared by a continuously growing user base of 28 million users. In schools, block-based languages display a similar success: Compared to text programs, block-based environments enable students to achieve better results on average (Weintrop & Wilensky, 2015), and while students like to shift to text sooner or later, they can nevertheless appreciate that block-based programming is equal to text-based programming (Bau, Bau, Dawson, & Pickens, 2015). Block-based languages are characterized by a number of common features, such as a stage, and code blocks that snap together to create scripts and replace the syntax of text-based languages. These features open up new possibilities and approaches to solving problems. In this paper, we give examples for these core concepts, and elaborate upon the origins of the image nesting concept in particular. Within this context, the proposition that block-based languages enable users to solve problems in new ways will be illustrated with an example: extending a programming environment, which may seem to be one of the most complicated tasks users could undertake. However, given the possibilities of the core concepts found in block-based languages, how complicated will it ultimately be?

The example reinforces the claim that block-based languages such as Scratch, *Snap!*, or *GP*, support constructionist ideas, and demonstrates the complexity of projects users can create with them. For this purpose, the *GP* environment itself will be modified and extended (using its easily-accessible source

code) to include a feature that makes it more meaningful to us: the ability to search for certain blocks within the program code.

Constructionism in programming

The ideas of constructionism and block-based languages have always been closely linked: the programming language Scratch was built on the constructionist principles of *Logo* and *Etoys* (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). The three basic principles “low floor”, “wide walls” and “high ceilings” laid the foundation for the way Scratch and its family members behave and are displayed to the user (Resnick, et al., 2009).

However, whether Scratch supports the principle of “high ceilings” (high threshold for more advanced programmers (Weintrop & Holbert, 2017)) is a point of frequent discussion: feedback shows that learners and educators alike feel as though Scratch is too simple to enable complex, high-level projects. For this reason, GP was designed with the idea of enabling users to create significantly more complex solutions (Maloney, 2018).

Core concepts in block-based languages

Block-based languages, and the environments used to program with them, enable new solutions to known problems, while creating the opportunity to produce entirely new tasks and problems. The reason for this is that in multiple ways, these languages are fundamentally different from typical text-based languages and environments used for teaching programming novices.

To understand the ways in which these languages are different, it is important to recognize the aspects that constitute a programming language. To characterize these aspects, the term “core concept” will be used; these core concepts embody the nature of programming environments and languages and make them both graspable. They should also be central for the way programming languages are taught. The concepts found in these languages separate them from text-based languages used for introductory teaching situations. In the following, several examples of new and promising, or less well-known concepts will be provided and outlined. These examples were identified inductively by analyzing user and developer-made demo projects, and deductively from the programming environments of Scratch, Snap!, and GP.

- **Manual control of program flow** (abbreviated as “Buttons”)

Most block-based languages enable the user to start, to stop, and (in some cases) even to pause the currently-loaded program. This results in a feeling of directness for the user, making it possible for them to stop and observe their program, which offers interesting strategies for debugging.

- **Broadcasting**

Sending messages between objects is based on the *one-to-many* communication model, i.e. a single object sends messages to all other objects in a program. Other objects can then react to the received message – or not. This type of messaging is commonly used for synchronization and signaling purposes, as it ties into the event-driven program paradigm (outlined below).

- **Sensing**

In many block-based languages, objects (or “sprites”) are able to detect a number of different factors, including their own position on the stage, the direction they are facing, whether they are overlapping with any other objects on the stage, or whether a color is touching another color.

This sensing ability makes objects able to react autonomously to different situations, alignments, or conditions.

- **Prototyping**

Instead of conceptually abstract classes, many block-based languages utilize a system of prototypes and clones of prototypes in order to create new objects. The concept of prototyping can be further extended to include prototypical inheritance or “delegation” (Lieberman, 1986), a form of inheritance that does not require classes. For this reason,

prototyping and delegation can be perceived as more intuitive and easy to understand than other forms of inheritance.

Event-driven programming

If an event occurs, the corresponding script is executed. If multiple events occur simultaneously, or if multiple scripts react to the same event, multiple blocks are executed at the same time. This behavior results in an implicit type of concurrency that is intuitive and simple for learners to grasp. Events can originate from the user (i.e. key presses, or mouse clicks), or from within the environment (i.e. an object sensing overlap with another object, or messages being received).

Weak typing

One variable can hold different types of data; the same variable can store strings of letters, boolean values, or integers without being limited to a single type of data that is set at the time of its creation.

Stage

Most block-based programming languages feature a stage that can visualize the world of the project to the user, showing objects, and their (inter-)actions. Possible applications include using it as a canvas for drawing graphs, or as a playing field for various games.

Delayed execution of code

Typically, block-based languages slow down the execution of blocks within scripts. This is done for visualization purposes: without an inherent delay, objects could fly off the stage instantly without the user being able to observe the process. The concept is based on a non-atomic interpreter, and is typically found in block-based programming environments for didactical reasons.

Drag & drop (abbreviated as “D&d”)

While most of these languages support keyboard editing and input, learners will typically utilize the concept of dragging-and-dropping blocks from the sprite palette into the scripting area to create scripts. The same process also deletes scripts, moves objects on the stage, and enables the user to resize parts of the graphical interface to fit their personal preferences.

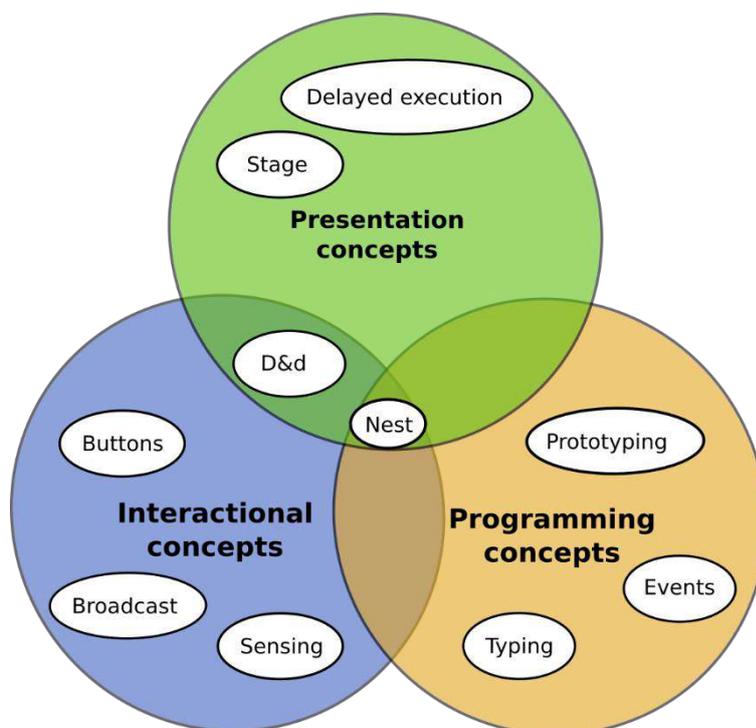
Image nesting (abbreviated as “Nest”)

The attachment of visual objects to other objects can be compared to the way objects are grouped in various software tools (such as Microsoft Powerpoint or OpenOffice Impress). Objects become parts of other, greater objects, while still retaining their individual identity. As a real-life example, a hand can be considered an attachment to the arm, of which it is a part. When the arm moves and rotates, so does the hand. However, both the hand can also still rotate and move (somewhat) independently. Therefore, while the hand is part of the greater object “arm”, it retains its individual nature, and can perform actions independently (such as rotation, and grabbing).

Note: this concept does **not** refer to the idea of nested loops, or code nesting through the use of syntactic elements such as brackets; it deals with the visual representation of objects and their composition into greater composite objects.

This is merely a non-exhaustive selection of core concepts found in block-based languages. As they make up the foundation of how programming languages function, their impact on teaching needs to be considered when designing tasks. These concepts are also the reason block-based languages lend themselves to constructionist solutions and tasks. It is important to note that not all of them are entirely new ideas; many (such as weak typing) have been part of programming languages and environments for decades.

This paper proposes that the core concepts of block-based languages characterize the way they are used and, in the same way, characterize the way they should be taught.



Concepts can be categorized into three overlapping categories:

Figure 1: Categorization model for core concepts found in block-based languages

1. Presentation concepts

“What do users see and **how**?”

This category includes all concepts regarding the presentation of the user interface. These design choices were made in order to support the learner's ability to interact with the environment, and to improve their understanding of the underlying programming concepts. The stage, for example, helps learners understand the nature of objects by offering a visual representation in the form of sprites.

While traditional environments make them seem abstract, or “ethereal” to many students, this visualization makes them more graspable and concrete. A common trait of most block-based languages is that for pedagogic reasons, several blocks (such as loops, all motion blocks and broadcasts) have an inherent delay associated with their execution. This delay was implemented to slow down program flow and further support its visualization.

2. Interactional concepts

“How do users interact with the environment?”

“How do objects interact with each other?”

The second category includes concepts concerned with the way users interact with the programming environment and its user interface, while also including concepts that deal with the way objects interact with each other.

An example within this category is the buttons a user can use to start, pause, and stop the program at any desired point. They enable the user to get a better feel for the program flow. The interaction of objects with each other includes concepts such as broadcasting, and sensing.

3. Programming concepts

“How do users implement their solutions?”

This category comprises both typical programming constructs (such as variables, loops and conditionals) and consciously-designed language traits that were implemented with accessibility, intuitiveness and ease of use in mind. For instance, an event-driven program flow (found in all members of the Scratch-family) enables users to implement projects that rely heavily on concurrency – intuitively, while encountering fewer of the typical problems and difficulties associated with concurrency (such as visibility, race conditions, etc.).

Certain concepts do not fit a single category. Dragging and dropping blocks to create scripts is a hybrid concept: while it fundamentally influences the way users interact with the programming environment to create meaningful projects, the possibility of doing so is suggested through the shape, shading, and highlighting of blocks. As another example, image nesting fits all three categories: presentation (due to the visual connection of nested sprites/objects), programming (after nesting, objects can refer to each other as “owner” and “part”), and interactional (as nesting creates a hierarchically-structured part-whole-relation between two objects).

The proposition is that these core concepts enable users to implement constructionist solutions easily – they are the reason block-based languages support constructionism to the extent they do. To illustrate this proposition, the concept of image nesting will be elaborated upon in more detail.

Image nesting: a core concept

The concept of nesting visual objects into others is based on the idea of *composition*. Naturally, the composition of individual, smaller parts to form a greater entity is by no means a novel concept; In the process of identifying fundamental ideas of computer science, Schwill deemed the idea of *structured dissection* a master idea (Schwill, 1994). Two of its sub-ideas are *hierarchization*, and the visualization as a *tree*. All these factors are closely related to the concept of nesting, leading to the conclusion that image nesting is an essential idea of computer science.

Despite its basic nature, nesting has the potential to enable very high-level solutions to complex problems (“high ceilings”), making it a core concept in block-based languages. As has been shown (Resnick, et al., 2009), block-based languages such as Scratch make it simple for users to create projects within the framework of constructionism. Commonly-identified traits of such frameworks are “low floors”, “wide walls”, and “high ceilings”. For Scratch, these three traits have been expanded to include “more tinkerable”, “more meaningful”, and “more social” (Resnick, et al., 2009).

How do these traits relate to image nesting, and other core concepts of block-based languages? Using the core concept of image nesting, a seemingly very complex, high-level task can be achieved: the programming environment GP will be modified to include a search feature.

This search feature, and the algorithm that is used to implement it, rely entirely on image nesting, therefore showcasing that block-based languages accommodate the ideas of *high ceilings*, *more tinkerable* and *more meaningful*.

In the following chapter, to gain a more substantial understanding of the concept in question, the origin of image nesting will be analyzed in more detail.

The origin of image nesting: Morphic

Morphic is a user interface construction environment. Originally developed as part of the language *Self*, it would eventually play a central role in the Scratch-language family: the first versions of Scratch used *Morphic* for the user interface creation, which may be the reason for the existence of nested sprites in Snap! and GP.

The first central principle that acts as the foundation for the entirety of *Morphic*’s functionalities is the *morph* (Greek for “shape”). Every visual object is graphically represented by a morph (Bouraqui & Stinckwich, 2007), which is the base class of everything that can be displayed in a “world”. As graphical entities, morphs are able show behavior and handle events: they can sense and react to user input, periodically perform actions, detect their own position and size, and sense overlap with other morphs. The second major concept found in *Morphic* is “composition”. Any morph can be made into a composite

morph through the attachment of others. The attached morphs are then considered submorphs, which keep a pointer to their owner they are part of. When the composite morph moves, turns, is copied, or deleted, the same action is applied to all its submorphs – and their submorphs recursively. While they depend on their owner in many ways, they retain their individuality: they are given a chance to handle events (such as button presses) before their owner does (Maloney & Smith, 1995).

The entire IDE of Morphic is based on this special structural relationship between morphs. As each morph has an owner (be part of a greater composite morph) or can have submorphs (be a composite morph), a compositional hierarchy is created. In this tree-like structure of part-whole-relations, the “world” is at the root, ultimately containing all the visible morphs within itself (Maloney & Smith, 1995).

How does Morphic relate to block-based languages?

Nesting in Snap! and GP is based on Morphic’s structure of composition, owner-morphs and parts. Comparing both systems yields the following central aspects:

- Nested objects behave the same way composite morphs do (act as a single entity when performing tasks).
- Individual parts know their owner, owners know all their parts; both can communicate with each other using these connections.
- Individual parts have a degree of independence from their owner (can perform actions and handle events with a higher priority than their owner).
- Both systems create a part-whole hierarchy between objects, and can therefore be visualized by a tree structure

Due to these similarities, it appears that composition in Morphic was used as the foundation of the way nested objects are handled in certain block-based languages; therefore, nested sprites are a new incarnation of composite morphs. The central ideas of composite morphs and image nesting are vital to understanding of how nesting works and what is meant by it.

GP particularly adheres to Morphic’s design philosophy. Each part of the user interface is a morph attached to an owner: the script editor, as an example, is a composite morph.

Its components are the individual scripts, which are composite morphs themselves. They are comprised of the individual blocks. Blocks consist of several parts: a colored frame, a text label, and typically one or more input slots. Blocks react to mouse clicks by running their associated code (the so-called “handler”). It becomes apparent that the hierarchical structure and functionality of morphs is part of GP’s design.

In the following chapter, these central aspects of image nesting will be utilized to add a search feature to the programming environment GP. The implementation of this feature will demonstrate how image nesting, and other core concepts found in these languages, enable new and intuitive solutions to previously complex problems.

How do concepts enable new solutions?

Block-based languages such as Scratch, Snap!, or GP, enable new solutions to problems due to the concepts they use. The very same concepts are also the reason block-based languages lend themselves to constructionist learning approaches. To give an example of this proposal, the programming environment of GP itself will be modified by utilizing its application of the nesting-concept. The implementation will highlight that image nesting enables three aspects in particular: “high ceilings”, “more tinkerable”, and “more meaningful”.

To illustrate this point, we will solve a two-fold problem outlined by the developers of GP: 1) Code represented by blocks typically takes up more space than it would with a text-based representation. This makes it harder to get an overview of the code at a glance (Mönig, Ohshima, & Maloney, 2015).

2) The visual stimuli, such as colors, borders and other graphical elements of blocks, can make it harder for the user to scan the actual labels of the blocks. These label texts, however, hold most of the meaning.

Text-based language editors typically solve this issue by providing a search function that can locate certain code snippets based on a given search term. For block-based language environments, such a feature does not yet exist – even if the user knows the block they need to find in the program, they must visually scan the entire scripting area to find it (Note: most block-based environments provide a search bar for the blocks palette. This searches the library of available blocks based on a search term; it does **not** search the code blocks in the scripting area).

The proposed solution to this problem, therefore, is the addition of a “search in class”-function that enables the user to find a block containing a given search term within the currently selected class. This should operate similarly to the “search”-feature of a text-based language editor, scanning the code contained in a project for a matching search term and pointing the user to its location.

Implementation of a search function in GP

In this step, the programming environment of GP itself will be extended to include a “search in class”-function. But what is the role of image nesting for this extension?

Image nesting as a core concept of several block-based languages applies not only to objects on the stage (so-called “sprites”): it can also be found within the graphical interface of GP itself. It is one and the same core concept – a hierarchical part-whole relationship between two objects.

Due to GP’s graphical interface structure relying on image nesting, searching for a specific block label only needs to follow a basic algorithm.

However, there are several terms and their relation to each other that need to be clarified (Figure 2):

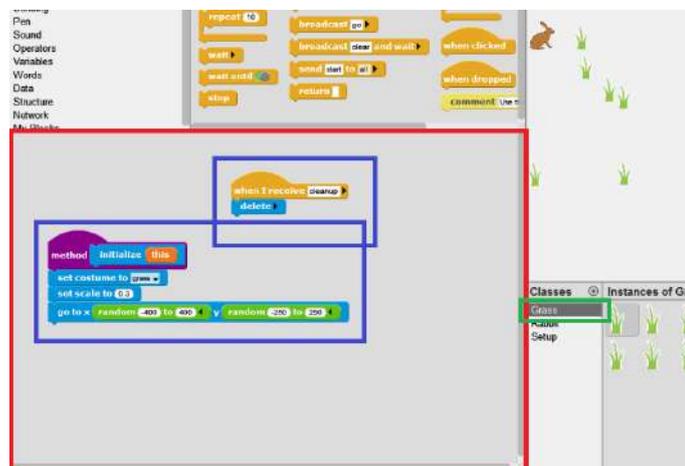


Figure 2: Interface of GP

ScriptEditor (red box): The area that contains all the code of a class. The ScriptEditor is a composite morph; its parts are the individual scripts (blue box).

Class (green box): GP uses classes and objects. The ScriptEditor always displays the code blocks for the currently selected class; this code is shared among all its objects (or “instances”). The currently selected class for the example in Figure 2 is “Grass”.

Scripts (blue boxes): Any number of blocks (≥ 1) attached to each other form a script. Every script is a composite morph; among its parts are the individual blocks. The selected class has **two** scripts: “when I receive”, and a method definition for the “initialize”-block.

Blocks: Parts of a script, internally attached by nesting. The “when I receive”-script has **one** block attached.

labelText: A part of a block, contains the actual readable text on the block. The labelText of the “delete”-block is the text string ‘delete’.

The hierarchical structure of the graphical interface that is created by image nesting can be visualized by a tree (Figure 3). This tree shows the position of the label texts in the hierarchy of the graphical interface: the texts are nested into the individual blocks, which in turn are nested into the scripts. Finally, the scripts are nested into the ScriptEditor, which is the part of the GP interface that holds all the scripts of a class.

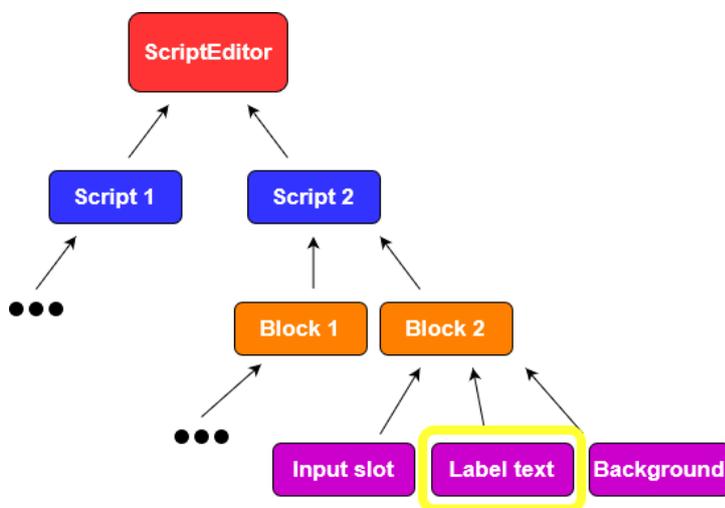


Figure 3: Hierarchical structure of the interface

The algorithm below (simplified) is outlined in pseudo code. This representation was chosen due to the inevitable simplification of the algorithm in place, although a block-based representation may have been more fitting.

To find the matching string, we need to compare the text labels of all visible blocks. To this end, we utilize the concept of image nesting. We will not, however, nest objects into other objects ourselves. Instead, we will utilize what happens when code blocks are snapped together: they are nested into one another.

This impacts our approach to a solution significantly: we merely need to descend the hierarchical structure (Figure 3) that makes up the graphical interface in order to find the label texts of blocks.

```

1 set answer to (ask "Enter search term")
2 set searchTerm to answer
3
4 for every script of ScriptEditor
5   for every block of scripts
6     for every part of block
7       if (searchTerm == part)
8         return true
  
```

After the user's search term has been stored (lines 1-2), the concept of nesting becomes important.

Descending the hierarchy of the graphical interface is done in lines 4, 5, and 6. In each of these lines, the composite object (ScriptEditor, scripts, and block, respectively) is prompted for a list of all its parts, that means other objects that have been nested into it.

This hierarchy is followed downwards until the label text becomes available (line 7). The algorithm is therefore no more complicated than parsing a tree

and its nodes.

As an example, breaking down the "delete"-block yields three parts – a blue puzzle-shaped image, the label text "delete", and an image of a black arrow. In this manner, every block of every script shown in the ScriptEditor is decomposed into its nested parts in order to find the label text.

This label text can then be compared to the *searchTerm* entered by the user. If both strings match, the result has been found. The search menu is accessed through the context menu by right-clicking the scripting area. After a matching block label has been found, the ScriptEditor scrolls to it and a highlight is added to signal its position.

Discussion

The reason for the simplicity of the algorithm described here is the architecture of the user interface: as each visible object (like the ScriptEditor, the scripts, the blocks, etc.) is a part of a larger (composite) object, and is in turn made up of parts (making it a composite object), a hierarchical structure is created. Descending this hierarchy downward to the point where the label text of blocks becomes accessible makes it possible to keep the search algorithm simple and intuitive. Therefore, the search algorithm makes use of the **nesting** of images that are used to make up the user interface of GP. Furthermore, it also makes use of three central ideas of constructionism in block-based languages:

- **High ceilings:** Image nesting enables simple solutions to complex problems: An example could be implementing an autonomous car: the task is to create a car that is able to follow a race course reliably without user input; the issue is that to turn appropriately, the car needs to be able to determine where exactly it touches the edge of the course. In block-based languages, this can be implemented simply by nesting sensor objects onto the car; these sensors notify their owner if they touch the edge of the race course, to which the owner reacts by turning according to the notifying sensor.
At the same time, however, nesting also enables solutions on a much higher level; as described in this paper, the seemingly complex task of modifying a programming environment can be done by utilizing its reliance on image nesting.
- **More tinkerable:** As GP is open to modifications, the search algorithm described above can be implemented entirely in blocks through the usage of GP's system class browser.
- **More meaningful:** By extending the programming environment of GP according to our individual needs, we modified it to become meaningful for us (Resnick, 1996).

The search feature is by no means indicative of the limits to image nesting: with some adjustments and extensions, this feature could also be used to parse blocks in a project for various purposes, for example, an analysis tool examining the block types users tend to use; the “high ceiling” has not yet been hit. Furthermore, while the language of this implementation is specific to GP, the algorithm itself is not; the algorithm is based on the usage of image nesting. Therefore, similar implementations are possible in other programming environments that utilize nesting to create their graphical interface, such as Snap!.

Conclusion

As the example illustrates, the graphical representation of elements in block-based languages enables the reduction of algorithms to their basic principles. For the user, this means that the implementation of personally-meaningful, constructionist projects is simplified. In the case of image nesting, which originated from Morphic's composite morphs, the unique part-whole hierarchy it creates between objects makes it possible to solve a wide range of different tasks. Among these tasks are typical programming projects implemented by novices in introductory courses, such as autonomous cars. Even on a higher level (“high ceilings”), however, block-based languages enable constructionist solutions to known problems, as has been demonstrated with an extension of the GP programming environment. In conclusion to the posed question, “how high is the ceiling?”: we have not hit the ceiling quite yet. With even basic concepts such as image nesting, high-level solutions to problems can be implemented in a simple manner. Indeed, the particular feature implemented in this paper may be part of future GP versions.

References

Bau, D., Bau, D. A., Dawson, M., & Pickens, C. S. (2015). Pencil Code: Block Code for a Text World. Proceedings of the 14th International Conference on Interaction Design and Children, 445-448.

Bouraqadi, N., & Stinckwich, S. (2007). Bridging the gap between morphic visual programming and smalltalk code. Proceedings of the 2007 international conference on Dynamic languages: in conjunction with the 15th International Smalltalk Joint Conference 2007, 101-120.

Krahn, R., Ingalls, D., Hirschfeld, R., Lincke, J., & Palacz, K. (2009). Lively Wiki A Development Environment for Creating and Sharing Active Web Content. Proceedings of the 5th international Symposium on Wikis and Open Collaboration , 9.

Lieberman, H. (1986). Using prototypical objects to implement shared behavior in object oriented systems. ACM Sigplan Notices, 21(11), 214-223.

Maloney, J. (2018). GP: A New Blocks Language for CS Education. Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 1110.

Maloney, J., & Smith, R. B. (1995). Directness and Liveness in the Morhic User Interface Construction Environment. Proceedings of the 8th annual ACM symposium on User interface and software technology, 21-28.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. ACM Transactions on Computing Education (TOCE), 16.

Mönig, J., Ohshima, Y., & Maloney, J. (2015). Blocks at Your Fingertips: Blurring the Line Between Blocks and Text in GP. Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE, 51-53.

Resnick, M. (1996). Distributed constructionism. Proceedings of the 1996 international conference on Learning sciences, 280-284.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . .

Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM 52(11), 60-67.

Schwill, A. (1994). Fundamental ideas of computer science. Bulletin-European Association for Theoretical Computer Science, 53, 274.

Weintrop, D., & Holbert, N. (2017). From Blocks to Text and Back: Programming Patterns in a Dual-modality Environment. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 633-638.

Weintrop, D., & Wilensky, U. (2015). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. 11th Annual ACM Conference on International Computing Education Research, 101-110.

Weintrop, D., & Wilensky, U. (2017). Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments. Proceedings of the 2017 Conference on Interaction Design and Children, 183-192.

The Direction and Possibility for Social Justice in Informatics Education based on Bebras Challenge in Republic of Korea

Ungyeol Jung, *puynagi@gmail.com*

Korea National University of Education, Republic of Korea

Young-jun Lee, *yjlee@knue.ac.kr*

Korea National University of Education, Republic of Korea

Abstract

In addition to high interest in computational thinking ability, the 2015 revised national curriculum of Republic of Korea has set as an emphasis on revising skills for the 4th Industrial Revolution era through the enhancement of the ability, and has made informatics mandatory for the middle school. As a result, regional, gender, teacher level, and students' prior knowledge levels are pointed out as important environmental variables affecting informatics education and raise of students' computational thinking of computing, and as a threat to the social justice of informatics education.

In this study, based on the tasks and the results of the Bebras Challenge, the applicability of Bebras challenge for social justice in informatics education was analyzed from two perspectives (internal and external) and suggested implications. As a result of the research, the tasks of the Bebras challenge reflected educational elements for social justice. Also, the results of the Bebras contest showed that informatics education based on Bebras challenge could contribute to the social justice.

The results of this study will provide the basis for the direction and possibility of Bebras challenge for social justice in informatics education.

Keywords

keyword; informatics education, software education; social justice; Bebras Challenge

Introduction

The 2015 revised national curriculum of the Republic of Korea was developed on the basis of the national and social needs for cultivating creative and convergent talents and the awareness of the problems of the 2009 revised national curriculum. Especially, the curriculum provides the education paradigm that cultivates talented people with core skills required by the future society and implements happy learning by improving the quality of experience for the students' real life. This means that our society is being innovated so as to bring about changes in education. Especially, for the future of the nation, the influence of artificial intelligence, internet of things, cloud computing, big data, and mobile technology on the social, political, economic, cultural, artistic and educational fields is inevitable.

Accordingly, in the 2015 revised national curriculum, software (SW) education aimed at cultivating computational thinking ability was imposed in the K-12 in order to strengthen the future capacity of the students who will live in the intelligent information society brought the 4th industrial revolution. On the other hand, it is not the first time that computational thinking was introduced in the 2015 revision curriculum in Republic of Korea. Computational thinking has been included as an educational goal through the 2009 revised informatics curriculum.

However, this is limited to secondary schools in in the elective courses, not mandatory. Of course, in the 2015 revised national curriculum, only the minimal time and contents of SW education were presented without independent subject for elementary school. However, SW education in middle schools and high schools is the essential education through informatics for professional and systemic education. In particular, since the first national curriculum has been implemented, it is the first case that the elective subject has become mandatory. Therefore, the software education as compulsory and

informatics as mandatory have great significance for educational history in Republic of Korea. The fact that the informatics has become a compulsory and mandatory subject leads to the expectation that it can cultivate various kinds of abilities such as collaborative problem solving ability and information culture literacy as well as computational thinking in various aspects. The role and impact of the national curriculum in Republic of Korea is extensive and powerful because the curriculum presents the characteristics, goals, content system, achievement standards (content elements, skills), teaching and learning methods as well as teaching times of the subject relatively in detail.

However, unlike this expectation, there is a concern that the effect of SW education varies depending on the infrastructure, the capacity of the informatics teacher, the students' gender and prior knowledge level. In particular, the Korean education market is growing because of the concerns of the Korean people, who have a high level of education and passion for education. These concerns can be a great risk to social justice, a very important value in education.

Therefore, in this study, we aimed to search and analyse for social justice in informatics education through the Bebras challenge, which is developed for the purpose of motivating K-12 students for informatics, and promoting their computational thinking ability. The results of this study are expected to provide the basis for the direction and possibility of Bebras challenge for social justice in informatics education.

Directions of Social Justice in Informatics Education based on Bebras challenge

Social Justice in Education

The concept of social justice is closest in meaning to fairness, emphasizing communication and understanding, humility, compromise, and tolerance rather than emphasizing one aspect of the public good or the private good. Justice is some similar and somewhat different from equality. In other words, everyone should be educated without discrimination, but the difference of results should be acknowledged only by the factor of the ability. On the other hand, it cannot be denied that the purpose of education is to realize social justice, but the ignorance and unawareness of these concepts and values have made the view that education is for social inequality.

Paulo Freire (1970, 2009), who advocated education for social justice, said that students in literacy education critically read and critically rewrite the world and that participation for individual liberation is important. Applying Freire's liberation to informatics education, students should be taught and prepared informatics to think critically about the oppressive part of the social structure or environment through the concept or technology of informatics and computational thinking. In this respect, D'Ambrasio (1999) stated that students should be taught to apply and infer knowledge of the subject within the social context. In other words, we need a curriculum for leading students' interest, curiosity, and creativity.

Skovsmose (1994) and Frankenstein (1983, 2001) presented the teaching and learning direction of social justice in mathematics education in two perspectives. It is 'mathematics for social justice' and 'critical mathematics' education. 'Mathematics for social Justice' means to understand the world and to improve problem-solving ability through mathematics. It can be interpreted as a goal and an effect of mathematics education. In addition, 'critical mathematics' means to cultivate positive cultural and social identity through learning of mathematics. It means to cultivate critical awareness through teaching mathematics.

Choi (2015) presented the teaching topics and contents for social justice. It is possible to teach 'critical mathematics' through the topics and contents focusing on social problems and it is possible to teach 'mathematics for social justice' the contents focusing on mathematical problem solving.

Through this literature analysis, 'Informatics for social justice' means that everybody should be able to gain the opportunity for computational thinking. In addition, 'critical informatics' means to have a positive viewpoint on different races, classes, genders, etc. through learning informatics.

Bebras Challenge, Informatics Education, and Social Justice

Bebras challenges are an educational model and an initiative that is designed to motivate K-12 students in informatics and to improve computational thinking. The challenge is based on tasks that anyone can participate and challenge regardless of gender, region, prior knowledge and so on.

All the tasks of the Bebras challenge involve interesting and familiar real-life problem situations, and participants experience 'abstraction' that analyzes problem situations and designs problem-solving models as well as automation that develop and apply problem-solving procedures. Through this process, logical thinking ability and computational thinking ability can be cultivated. This means that it is possible to educate 'informatics for social justice' through teaching and learning informatics based on Bebras challenge.

Jung and Lee (2017) analyzed the educational effects of the Bebras challenge and found its applicability to the 2015 revised national curriculum of Republic of Korea. According to them, the Bebras challenge can measure whether it is possible to deeply understand and apply the basic concepts of informatics beyond simply knowledge acquisition or memorization. It also improve students' affective ability through solving attractive and challenging tasks, motivation, attitude, interest and so on.

In addition, it is possible to diagnose computing thinking ability, which allows to provide appropriate feedback for students. Furthermore it can be applied to various teaching, learning and evaluation models based on competition and collaboration and it can improve students' ICT literacy and abstraction, algorithm design, and programming ability through interactive tasks based on IBT with visualized and simulated environment.

All the tasks of Bebras challenges include various multi-cultural elements as they are developed through collaborative development and reviewing from the experts of various countries.

In particular, there is no discrimination against specific countries, classes, and races because they place an emphasis on attractive problem situations that 'everyone' can challenge. This means that it is possible to educate 'critical informatics' education through Bebras challenge.

Possibilities of Social Justice in Informatics Education based on Bebras challenge

The purpose of this study is to analyze the effectiveness of informatics education based on Bebras challenge with the perspective of social justice. To do this, we set up the tasks and the test-results of Bebras challenge 2017 in Republic of Korea. The challenge was conducted by Bebras Korea, a nonprofit organization composed of Korean experts in the field of informatics education, and 7,208 students from five age groups, from Group II (Primary) to Group VI (Seniors), participated in the challenge.

The analysis of this research was done in two aspects. The first thing is about the social problem factors included in the tasks of Bebras challenge 2017 in Korea so that we explore the possibility of 'critical informatics education'. The second is about the participants' results of the challenge so that we find the possibility of 'informatics for social justice'. Therefore, the research problem to be solved through this study is as follows.

- Is Bebras challenge suitable for 'Critical Informatics' education?
- Is Bebras challenge suitable for 'Informatics for social justice' education?

Internal Analysis: Bebras Tasks for Social Justice

In order to figure out whether the Bebras challenge is suitable for critical informatics education, it is necessary to analyze whether the tasks of this challenge contain the elements for social justice. Therefore, this can be said that the internal analysis on Bebras challenge.

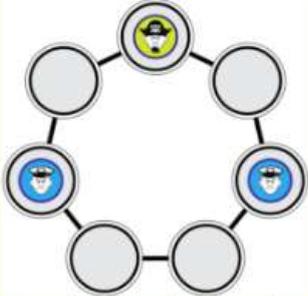
Choi (2015) and Lim (2017) suggested five important topics and contents for teaching and learning social justice. Therefore, it is necessary to search whether the tasks of the Bebras challenge includes the following elements for solving the 1st research problem.

No.	Topics or Contents
1	about Social Unjust
2	about Equity and Public Virtue
3	about Consideration of Social Minorities
4	about Students' Interests and Concerns
5	about Multicultural and Global Materials

Table 1. Topics or Contents for Education of Social Justice

Frist, The content about social unjust means to construct educational contents in such a way as to find out or solve unfair problem situations. The following is a task of Bebras challenge that explores the appropriate police actions to catch a pirate (a bad man) under given conditions.

Jane and Jill play a board game Pirate Hunters. At each move, one of the policemen (but not both) moves to a neighboring place. In the next move, the pirate, who is faster and always jumps for two places. Policemen always move to an unoccupied place – they cannot move to a place occupied by the pirate or his colleague policeman. The game is finished when the pirate is forced to jump onto one of the policemen ... which would be now (see the picture), except that it is currently the policemen turn. To win, the policemen must force the pirate into this position when it is the pirate's turn.



Jane, who plays the pirate is quite skilled at evading being captured. You are smart as well, though. If you help Jill play a perfect game, how many moves will she make before the pirate is caught?

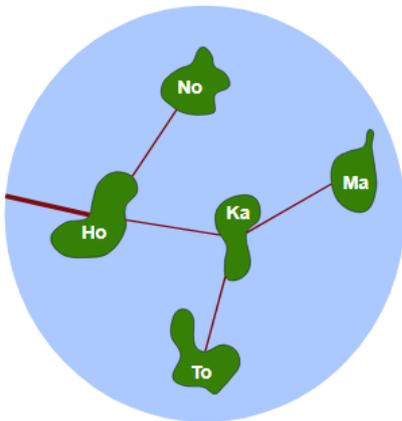
Figure 1. Pirates (2015-SI-07; Age group: Juniors; Difficulty: hard)

Students try to catch pirates who are unjust in the solution of this problem, and through this process, they become police officers and endeavor to realize social justice. The examples like this task can be found in "Find the Thief (2016-BE-02; Age group: seniors; Difficulty: hard)" to search the thief who stole a diamond and "Intrusion (2017-DE-03; Age group: cadets; Difficulty: hard) ' to prevent unauthorized intruders from the museum.

Second, the contents about equity and the public virtue refer to educational contents that encourage to recognize or solve the problem for the more people. The next task is to find a way to ensure that all islands are able to communicate even if one thin cable is cut off.

A Resilient Network for Honomakato

The Honomakato archipelago consists of the five beautiful islands Ho, No, Ma, Ka, and To. The big island Ho is connected to the internet by a big cable. In addition, small cables connect Ho and No, Ho and Ka, Ka and Ma, and Ka and To. Via these cables, all islands are connected to Ho and, therefore, the internet.



The Honomakato people want the network to be resilient: If one of the small cables breaks, no matter which one, every island shall still be connected to the internet.

If two more cables are laid to make the network resilient, which of the following is correct?

Figure 2. Honomakato MC (2017-DE-06a; Age group: Seniors; Difficulty: medium)

In solving this problem, students try to find ways to create a situation in which everyone can communicate equally. The examples like this task can be found in 'Soda Shoppe (2017-CA-07; Age group: seniors; Difficulty: easy)' to find ways to drink all your friends' favorite drinks and 'Bebragram; Age group: seniors; Difficulty: easy)' to find out how all your friends can buy music.

Third, the contents about consideration of social minority means the educational contents to recognize the problem situation for the handicapped, the elderly and children, or to induce them to solve their difficulties. The following is a Bebras challenge task to find a way to make delicious food (twigs) for a one armed Bebras.

The more leaves a branch has, the tastier it is. So, David intends to sort the branches based on their taste using the temporary storage beside him.



Question / Challenge

Please help David sort the branches by taste so that the tastiest branch is the closest to him.

Answer Options / Interactivity Description

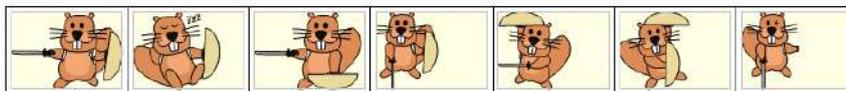
In the graphics above the six branches should be draggable to any empty place or the temporary place on the left. If he tries to move a branch to any other place (especially a place where another branch is in), it snaps back to its original position. There needs to be a "save" and a "reset" button.

Figure 3. One Armed Bebras (2017-CH-08b; Age group: Primary; Difficulty: medium)

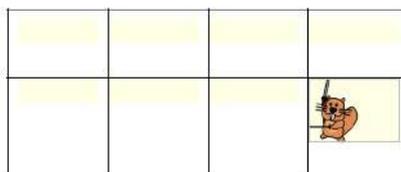
In the process of solving this problem, the students have a desire to help the one armed Bebras. The examples like this task can be found in 'Grandmother's jam (2017-RU-04; Age group: Primary; Difficulty: medium)' to find a way to help grandmother who want to make jam and 'Bird House (2017-RO-03a; Age group: Primary; Difficulty: easy)' to prepare a daughter's birthday present.

Fourth, the contents about students' interests and concerns are related to the use of familiar materials, pictures, or real-life problems. The following is a task to find a suitable picture combination with a Ninja Bebras which is attractive for students.

Lucia is playing stick and shield with 7 friends. The following diagram shows for each of the friends what their favorite pose is:



They want to have their picture taken in the school yard. In the picture every stick should point at another beaver, and every shield should block a stick. Lucia has already taken a spot in the picture.



Question / Challenge

Arrange the beavers according to the rules.

Figure 4. Stick and Shield (2017-JP-02; Age group: Cadets; Difficulty: medium)

Students are interested in the action and facial expressions of cute ninja Bebrass holding a stick and a shield in solving this problem. The examples like this can be found in "Irrigation system (2017-AT-05; Age group: Cadets; Difficulty: easy)" to fix the real-life problem, and "Color the flowers (2016-SK-04 ; Age group: Primary; Difficulty: easy) ' with beautiful flowers.

Fifth, contents about multicultural and global material means constituting educational contents in the way of problematic situation of various countries and races as material. The following is a tasks to find ways to change the lighting conditions of skyscrapers.

Skyscrapers art

There is a new skyscraper in the town. It has 26 windows that can be illuminated (yellow) or not (purple); the building has 7 floors (0-6) and 4 units (A-D).

The architect has put some switches in the basement and the skyscraper can be used to show beautiful light pictures:

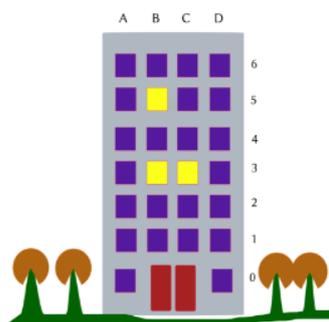


Figure 5. Skyscraper art (2017-IT-06; Age group: Cadets; Difficulty: hard)

In the process of solving this problem, students in Korea will be questioning the beginning of the building from '0' (not 1), and realize that this is a difference between culture and custom. There are many other problems like this, but the students especially have looked at Beavers' different shapes, facial expressions and color, and clothes from different countries in the tasks and imagine the people of the country.

These results indicate that the elements for critical informatics are included in the tasks of Bebras challenge. Therefore, 'critical informatics' education based on Bebras challenge is possible.

External Analysis: Results of Bebras Challenge 2017 in Korea

In order to find out whether the Bebras challenge is suitable for informatics education for social justice, the focus should be on whether students will be given opportunities to improve their computational thinking through the challenge. Therefore it is necessary to analyze the participants and results of the participation. This approach can be called the external analysis on Bebras challenge

Dagienė & Futschek (2008) and Dagienė & Stupuriene (2016) said that everybody can be participants to Bebras challenge regardless of gender, region, or prior knowledge and all tasks of this challenge are based on interesting and familiar real-life problems. Thus, participants in the Bebras challenge can be motivated for informatics and can enjoy their computational thinking regardless of their different backgrounds.

On the other hand, Jung et al. (2018) presented the points to consider for the informatics by analyzing the background factors influencing the secondary school participants' results (N = 5,874). The students participated in Bebras challenge 2017 in Korea. As a result, the following conclusions were obtained.

- First, as a result of analyzing the difference in the percentage of correct answers of all participants according to their genders, the result of male students was higher than female students. However, as a result of analyzing the difference for each group, there were statistically significant differences only in Group VI (Seniors), and there was no difference between Group IV (Cadets) and Group V (Juniors)
- Second, as a result of analyzing the difference according to the region, it was found that 'Rural > Big city > Small city'. These results are somewhat different from those of previous studies that worried about the educational gap between urban and rural areas (City > Rural).
- Third, there was no statistical difference in the percentage of correct answers according to the evaluation areas (ALP, DSR, CPH, COM, ISS) and task-types (Multi-choice, Interactive). However, we should pay attention the result that the percentage of correct answers (N = 5874, m = 51.77, df = 36.41) in the CPH area was very lower than the overall average (N = 5874, m = 34.49, df = 18.89).

In this study, to find out the difference of the percent correct according to the region in more detail, we analyzed the difference based on the results of the students of each. The results are as follow.

- First, the average (avg.) of the big city (N = 221) was 69.05 and the standard deviation (SD) was 39.84 among the students in the group IV (Cadets). The avg. in the small city area (N = 1443) was 55.05 and the SD was 37.95. The avg. of rural (N = 91) was 81.14 and the SD was 35.39. The F-value was 30.46 and p-value was .00 when the significant level of this research was .05. The results of the post-hoc analysis (Scheffe) are as follows. This result implies that a statistically meaningful interpretation can be made for 'Rural > Big city > Small city'.

Comparative Group (GroupIV)	mean difference	SE	p
Big city vs. Small city	14.01	2.75	.00
Big city vs. Rural	-12.09	4.74	.04
Small city vs. Rural	-26.10	4.11	.00

Table 2. Results of Post-hoc analysis (GroupIV)

- Second, the avg. of the big city (N = 256) was 40.82 and the SD was 35.10 among the students in the group V (Juniors). The avg. in the small city area (N = 574) was 48.94 and the SD was 39.21. The avg. of rural (N = 23) was 44.74 and the SD was 26.77. The F-value was 4.12 and p-value was .02 when the significant level of this research was .05. The results of the post-hoc analysis (Scheffe) are as follows. This result implies that a statistically meaningful interpretation can be made for 'Small city > Big city'. However, there was no difference between 'Big city and Rural' and 'Small city and Rural'.

Comparative Group (Group V)	mean difference	SE	p
Big city vs. Small city	-8.12	2.84	.02
Big city vs. Rural	-3.92	8.21	.89
Small city vs. Rural	4.20	8.03	.87

Table 3. Results of Post-hoc analysis (Group V)

- Third, the avg. of the big city (N = 964) was 53.76 and the SD was 34.33 among the students in the group VI (Seniors). The avg. in the small city area (N = 2258) was 47.94 and the SD was 34.46. The avg. of rural (N = 44) was 54.05 and the SD was 26.70. The F-value was 10.06 and p-value was .00 when the significant level of this research was .05. The results of the post-hoc analysis (Scheffe) are as follows. This result implies that a statistically meaningful interpretation can be made for 'Big city > Small city'. However, there was no difference between 'Big city and Rural' and 'Small city and Rural'.

Comparative Group (Group VI)	mean difference	SE	p
Big city vs. Small city	5.82	1.32	.00
Big city vs. Rural	-.28	5.29	.99
Small city vs. Rural	-6.10	5.22	.51

Table 4. Results of Post-hoc analysis (Group VI)

These results show that it is difficult to generalize the test-results of Bebras challenge according to specific background factors. Instead, it emphasizes the importance of detailed analyzing the individual's computational thinking ability or misconceptions and providing appropriate feedback to help them. Therefore, 'informatics for social justice' education based on Bebras challenge is possible

Conclusion and Discussion

The purpose of this study is to find out how Bebras challenge can contribute to social justice in informatics education based on the analysis of the tasks and the results of the Bebras challenge 2017 in Republic of Korea. Based on the results of this study, the following conclusions can be drawn.

First, there are two educational directions for social justice in informatics education. The 'Informatics for social justice' education is to provide all students with opportunities to develop computing thinking skills that understand the world and solve problems from the perspective of informatics. 'Critical informatics' education is aimed at improving positive social and cultural identity for all students.

Second, the Bebras challenge includes various elements necessary for 'Critical informatics' education. This is based on the developing principles and processes of the tasks that allows for consideration of diverse cultural and ethical issues. Therefore, it is possible to educate 'critical informatics' education based on Bebras challenge.

Third, the goals and characteristics of Bebras challenge are appropriate for 'Informatics for social justice' education. The results of the challenge 2017 in Korea also show that informatics education is not always beneficial to a particular group or subject. In particular, through detailed analyzing, we can practice 'Informatics for social justice' education.

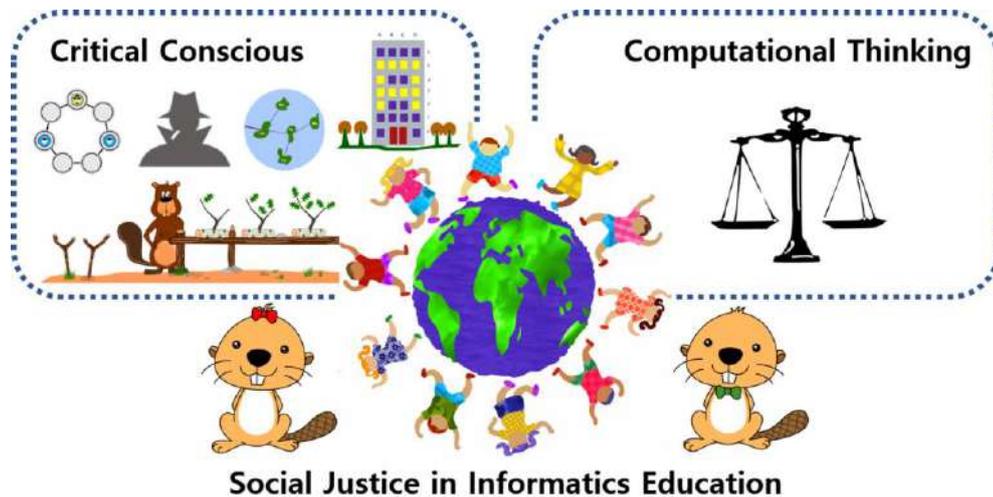


Figure 6. The directions and possibility for social justice in informatics education based on Bebras challenge

As the number of countries and students participating in Bebras challenge grows, various possibilities using Bebras challenge are being discussed. However, considering the goals and characteristics of the challenge, we should more focus on the social justice in informatics education based on this challenge. Therefore, we hope that this study will be the basis for the direction and possibility of Bebras challenge for social justice in informatics education.

References

- Bebras Challenge. (2018). <http://bebras.org>
- Choi, J., An, S. and Lee, Y. (2015). Computing Education in Korea-Current Issues and Endeavors. *ACM Transactions on Computing Education (TOCE) - Special Issue II on Computer Science Education in K-12 Schools*, 15(2), Article 8.
- Choi, S. (2015). Study on the development of elementary school mathematics program with a focus on social issues for the mathematically gifted and talented students for fostering democratic citizenship: based on the teaching mathematics for social justice. Ph.D. thesis, Ewha Womans University, Republic of Korea.
- D' Ambrasio, U. (1999). Ethnomathematics and its first international congress. *Zentralblatt fur Didaktik Mathematik*, ZDM, 31(2), 50-53.
- Dagienė, V., & Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*, 19-30.
- Dagienė, V., & Stupuriene, G. (2016). Bebras-a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education-An International Journal*, 15(1), 25-44.
- Frankenstein, M. (1983). Critical mathematics education: An application of Paulo Freire's epistemology. *The Journal of Education*, 165(4), 315-339.
- Frankenstein, M. (2001). Reading the world with math: Goals for a critical mathematical literacy curriculum. *Proceedings of the Eighteenth Biennial Conference of the Australian Association of Mathematics Teachers*, Canberra, Australia.
- Freire, P. (1970, 2009). *Pedagogy of the oppressed*. New York: Continuum.
- Jung, U. and Lee Y. (2017). The Applicability and Related Issues of Bebras Challenge in Informatics Education. *The Journal of Korean association of computer education*, 20(5), 1-14.

Jung, U., Kim, H., Lee, M., Lee, H. and Ahn S. (2018). A Study on the Factors Influencing Computational Thinking Ability of Secondary School Students in Bebras Challenge 2017. *The Journal of Korean association of computer education*, 21(3), 21-33.

Kim, J. and Park, M. (2015). The Influences of Teaching Mathematics for Social Justice on Students' Interest towards Mathematics and Perceptions of Mathematical Values. *Journal of Elementary Mathematics Education in Korea*, 19(3), 409-434.

Korea Ministry of Education. (2015). The 2015 revised national curriculum.

Lee, M. (2017). Computational Thinking: Efforts in Korea. In: Rich P., Hodges C. (eds) *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations*. Springer, Cham.

Lim, H. (2017). A Study on the Realization of Contents of Mathematics Education for social justice. Master thesis, Ewha Womans University, Republic of Korea.

Park, S. C. (2012). A Study on Social Justice in Multicultural Education. *Center for Educational Research*, 1-26.

Seo, K. (2017). A Narrative Inquiry on a Teacher's Struggle to Teach for Social Justice. *The Journal of Curriculum Studies*, 35(3), 129-156.

Skovsmose, O. (1994). *Towards a philosophy of critical mathematical education*. Dordrecht, Netherlands: Kluwer.

Interconnection between Computational Thinking and Digital Competence

Anita Juškevičienė, anita.juskeviciene@mii.vu.lt
Vilnius University, Lithuania

Valentina Dagiene, valentina.dagiene@mii.vu.lt
Vilnius University, Lithuania

Abstract

The European Commission Science Hub has been promoting Computational Thinking (CT) term as an important 21st century skill or competence. However „despite the high interest in developing computational thinking among schoolchildren and the large public and private investment in CT initiatives, there are a number of issues and challenges for the integration of CT in the school curricula⁴⁶.“ From the other side, the Digital Competence (DC) Framework 2.0 (DigCom) is promoted in the same European Commission Science Hub portal⁴⁷. It shows that both topics have many things in common. Thus there is the need of research on CT relationship with digital competence.

The goal of this paper is to analyse and discuss the relationship between DC and CT, and help educators as well as educational policy makers to make informed decisions about how CT and DC can be included in their local institutions. We begin by defining DC and CT and then discuss the current state of both phenomena in education in multiple countries in Europe. By analysing official documents, we try to find the underlying commonness in both DC and CT, and discover all possible connections between them. Possible interconnections between both approaches components groups is presented in Fig. 1. Fig. 1. The interconnections between DC and CT

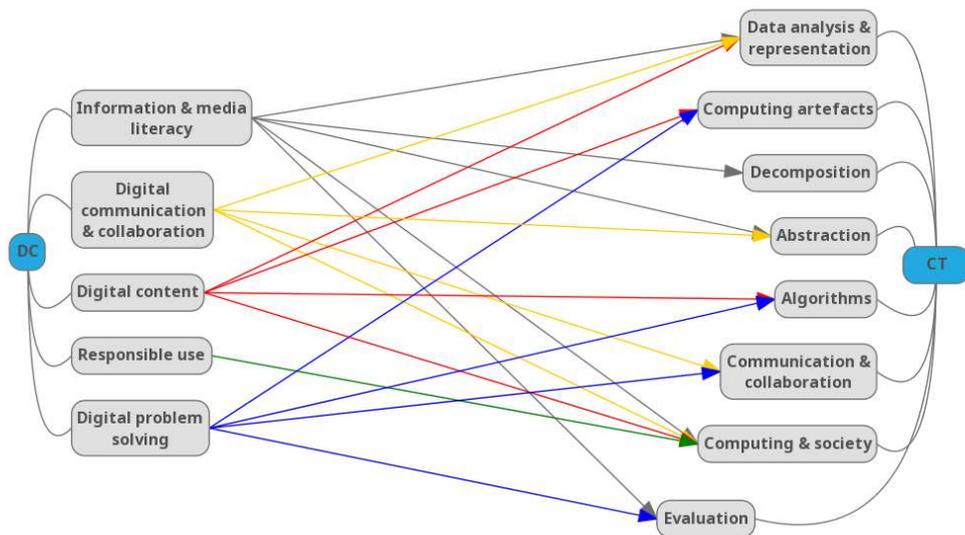


Fig. 1. The interconnections between DC and CT

Keywords

Computational thinking, digital competence, education

⁴⁶ <https://ec.europa.eu/jrc/en/computational-thinking>

⁴⁷ <https://ec.europa.eu/jrc/en/digcomp/digital-competence-framework>

Background

Computational Thinking (CT) and Digital Competence (DC) are indicated by many education policy makers as important 21st century skills. The European Commission Science Hub has promoted CT and has launched the Digital Competence (DC) Framework 2.0 (DigCom) in its portal. Nowadays CT and DC are essential skills and young generation should learn them for life.

During the last years, a lot of research work was devoted to the both topics, and enormous amount of studies and practical experience have been implemented. Nevertheless, there is the huge need of research in these topics on many aspects. One of them is the computational thinking relationship with digital competence.

Digital Competence

The invention of computer and internet has changed our lives and education sector. It requires individuals to improve their competencies, especially be digitally literate.

Digital competence is the most recent concept describing technology-related skills. Digital literacy is often seen as a synonym of digital competence, however there are some stages in development of this concept, for example: computer skills -> ICT skills -> digital skills -> digital competences (Ilomäki et al., 2011; Laar et al., 2017). Some researchers argue that digital skills concept, as a more holistic phenomenon, involves more features than digital competence (Pérez-Escoda, Rodríguez-Conde, 2015).

The number of content (e. g. media, tools, technologies) in the internet is growing every day. Thus, it requires individuals to deal with more abilities. JRC technical report (2012) analysed fifteen frameworks on DC and developed the following definition: “the set of knowledge, skills, attitudes (thus including abilities, strategies, values and awareness) that are required when using ICT and digital media to perform tasks; solve problems; communicate; manage information; collaborate; create and share content; and build knowledge effectively, efficiently, appropriately, critically, creatively, autonomously, flexibly, ethically, reflectively for work, leisure, participation, learning, socialising, consuming, and empowerment”. Being digitally competent implies the particular abilities, such as, understanding media, searching for information and be critical about what is retrieved, and communication with others using a variety of digital tools. Seven areas of DC were identified: (1) information management, (2) collaboration, (3) communication and sharing, (4) content and knowledge creation, (5) ethics and responsibility, (6) evaluation and problem solving, and (7) technical operations. The results of that report contributed to the DigCom project. Lately, contribution to the better understanding and development of DC in Europe was presented as DC framework involving five competence areas (DIGCOMP, 2013): (1) information, (2) communication, (3) content creation, (4) safety, and (5) problem solving. These areas consist of 21 competences. It was meta-framework for existing frameworks, initiatives, curricula and certifications. DigComp 1.0 has become a reference for many DCs initiatives at European and Member State levels. DigComp 2.0 keeps the same overall structure of 5 competence areas however slightly renamed (DigComp, 2016): (1) information and data literacy, (2) communication and collaboration, (3) digital content creation, (4) safety, and (5) problem solving.

One of the best known frameworks among educators and academics is the European Framework for the Digital Competence of Educators (DigCompEdu) which has six areas focusing on different aspects of educators' professional activities: (1) Professional Engagement – organisational communication, professional collaboration, reflective practice, digital continuous professional development, (2) Digital Resources – selecting, creating, modifying, managing, protecting and sharing digital resources, (3) Teaching and Learning – teaching, guidance, collaborative and self-regulated learning, (4) Assessment – strategies, evidence analysis, feedback and planning, (5) Empowering Learners – accessibility and inclusion, differentiation and personalisation, actively engaging learners, and (6) Facilitating Learners' Digital Competence (DigCompEdu, 2017).

DC involves the confident, critical and responsible use of, and engagement with, digital technologies for learning, at work, and for participation in society. It includes information and data literacy, communication and collaboration, digital content creation (including programming), safety (including

digital well-being and competences related to cybersecurity), and problem solving. The concept “digital technologies” is employed as an umbrella term for digital resources and devices, thus comprising any kind of digital input: software (including apps and games), hardware (e.g. classroom technologies or mobile devices) or digital content/data (i.e. any files, including images, audio and video).

Computational thinking

In 2006 Wing presented the concept of CT: “Computational Thinking involves solving problems, designing systems, and understanding human behaviour by drawing on the concepts fundamental to computer science” [Wing, 2006]. Based on this it can be concluded that CT involves three key components: algorithms, abstraction, and automation. However, the rise of this concept can be related to Seymour Papert by introducing it in the context of suggesting an alternative, computationally-based mathematics education [Papert, 1996]. Thence the researchers are very interested in CT approach and its application in education. However, it is still at an early stage of maturity [Lockwood & Mooney, 2017] and the steady definition of CT is not provided [Voogt, 2015]. In the attempt to define CT, researchers often focus on the core components of CT. As argued in [Voogt, 2015] it is more important try to find similarities and relationships in the discussions about CT rather than try to give an ultimate definition. They provide the discussion about the definition and core concepts of CT only in the Computer Science domain (excluding other domains), examined by covering literature from 2008 to 2013. Similarly, the analysis made in 2016 [Kalelioglu et al, 2016] provide the word cloud of CT definitions used in analysed papers covering 2006-2014. The generated ‘Wordle’ by Kalelioglu and others was based on the definitions provided by analysed researchers and not included some core concepts of CT. Additionally, CT definitions explained in the analysed papers were analysed and presented in percentage form of the words used to describe the meaning of the CT: problem solving (22%), abstraction (13%), computer (13%), process (9%), science (7%), data (7%), effective (6%), algorithm (6%), concepts (5%), ability (5%), tools (4%) and analysing (4%). Later, 59 definitions were analysed and classified into 7 themes in [Haseski et al., 2018] and they concluded that current limitations in the CT definition is that it is shaped by technology-aided problem solving. Thus further dimensions needed to be explored especially studies on personal, environmental, social, affective, psychological and ethical factors needed to be investigated.

CT definition challenge was analysed also in others domains, such as mathematics and science by [Weintrop et al., 2016]. They review the literature on CT and situated it historically till 2013. Thus they propose a definition of CT in the form of a taxonomy consisting of four main categories: data practices, modelling and simulation practices, computational problem solving practices, and systems thinking practices. Additionally, the set of ten CT skills were proposed:

1. Ability to deal with open-ended problems
2. Persistence in working through challenging problems
3. Confidence in dealing with complexity
4. Representing ideas in computationally meaningful ways
5. Breaking down large problems into smaller problems
6. Creating abstractions for aspects of problem at hand
7. Reframing problem into a recognizable problem
8. Assessing strengths/weaknesses of a representation of data/representational system
9. Generating algorithmic solutions
10. Recognizing and addressing ambiguity in algorithms

This set was the developed based on literature review on CT with a focus on applications to mathematics and science.

The focus on CT skills rather than definition meaning was presented in [Curzon et al, 2014]. They used a simplified set of skills: algorithmic thinking, evaluation, decomposition, abstraction, generalisation and implemented them in the developed workshops for teachers on CT themes in order to fill teachers’ knowledge gaps about CT and as a useful practical way that they can teach computing to school students. Very similar set of CT concepts were used to develop the relationship of CT concepts, student activity and curriculum subjects example [Catlin& Woollard, 2014]: abstraction, decomposition, algorithmic design, evaluation, and generalizations.

However the researchers and educators were searching for more CT elements in order to find out how CT definition can be interpreted. The literature review made by [Grover& Pea, 2013] showed that there are nine core elements widely accepted as comprising CT: Abstractions and pattern generalizations (including models and simulations); Systematic processing of information; Symbol systems and representations; Algorithmic notions of flow of control; Structured problem decomposition (modularizing); Iterative, recursive, and parallel thinking; Conditional logic; Efficiency and performance constraints; Debugging and systematic error detection.

Most of the researchers argue that CT is an activity, often associated with, but not limited to, problem solving [Beecher, 2017, p.8; Haseski et al., 2018]. For example, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) define CT as problem solving process that (but is not limited to) the following characteristics [CSTA&ISTE, 2011]:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them
- Logically organizing and analysing data
- Representing data through abstractions, such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analysing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem-solving process to a wide variety of problems.

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT, including:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open-ended problems
- The ability to communicate and work with others to achieve a common goal or solution

Additionally, they defined the vocabulary for CT in the purpose to explain the operational definition by listing nine CT concepts implicit in the operational definition such as: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, simulation and parallelisation.

As mentioned previously, CT can be seen as a problem solving process. It is in line with many aspects of 21st century competencies such as creativity, critical thinking, and problem- solving [Lye & Koh, 2014]. Thus it can be seen as one of the constructionist method (like the problem-based learning) which allows students to learn about a subject by exposing them to multiple problems and asking them to construct their understanding or objects of the subject through these problems. The pioneer of the constructivist theory, S. Papert, has expanded the theory of constructivism by stating that learning is best when the learner actively develops objects of the real world (for example, a sand castle, an automatic watering system), and not just ideas or knowledge that deliberately engages in design. The learning process itself is improved by improving the conditions that learners can construct.

Additionally, constructionism can be seen as the theory of particular relevance when considering lifelong learning. Lifelong learning is defined as "all learning activity undertaken throughout life, with the aim of improving knowledge, skills and competences within a personal, civic, social and/or employment-related perspective" [CEC, 2001]. Similarly, the CT is nowadays fundamental and children should learn it for life because it involves essential digital age competencies.

CT frameworks and existing practical implementation solutions

A framework for studying and assessing the development of CT was developed by (Brennan & Resnick, 2012). Three key dimensions based on studying activity in the Scratch online community and in Scratch workshops were identified: computational concepts (the concepts designers engage with as they program, such as iteration, parallelism, etc.), computational practices (the practices designers develop as they engage with the concepts, such as debugging projects or remixing others' work) and computational perspectives (the perspectives designers form about the world around them and about themselves). Based on these three dimensions, the suggestions how to assess learning programming were developed.

The College Board (2017) developed the CT framework for a Computer Science Principles course for high schools in the USA. They identify six CT practices: Connecting computing, Creating computational artefacts, Abstracting, Analysing problems and artefacts, Communicating, Collaborating. It is believed that these practises help students make sense of knowledge in order to accomplish the task, learn collaboration and communication principles.

Computing at School (CAS) presented the conceptual framework of CT by identifying key concepts (logic, algorithms, decomposition, patterns, abstraction, evaluation) and approaches (tinkering, creating, debugging, persevering, collaborating) involved in CT process as well as techniques (reflecting, coding, designing, analysing, applying) employed to demonstrate and assess CT (CAS, 2015). Each of the concepts of CT can be identified with approaches and techniques. This gives the opportunity to implement CT in classroom.

Framework for CT as a Problem-Solving Process was developed by (Kalelioğlu et al., 2016). It has five main categories of process that consist of the actions extracted from literature analysis. These categories are as follows: identify the problem; gathering, representing and analysing data; generate, select and plan solutions; implement solutions; assessing solutions and continue for improvement. They argue that this framework could help to teach, learn and practice CT and informatics concepts within many courses.

In the book by Krauss & Prottzman (2017), the CT framework of four categories: decomposition, pattern matching, abstraction and automation, is presented. Each has subcategories. This framework could help plan computer science lessons. The detailed descriptions of lesson plans for each category is given as well.

Scratch, App Inventor, LegoMind Stroma, CS Unplugged activities and various games are widely available tools and resources for CT development. CS Unplugged activities are adopted in different ways: videos, shows, outdoor activities, competitions. Another adoption is an internationally recognized challenge on informatics and CT called by *Bebras* (Beaver). Recently there have been over 60 participating countries. The *Bebras* challenge is an informatics education community-building model designed to promote informatics learning and CT at schools by solving short informatics concept-based tasks (Dagiene & Stupuriene, 2016). Alongside the initial goal of the *Bebras* project is to motivate students to be more interested in informatics topics, there is a strong intention to deepen algorithmic and operational thinking and extended to CT (Dagiene, Sentence, Stupuriene, 2017).

Interconnection between Digital Competence and Computational Thinking

The characteristics of CT and DC approaches based on results of literature review were analysed. The analysis showed that CT skills are overlapping and relatively broad in context. The eight CT's components groups were identified. Additionally, the list of abilities was identified.

The first CT concepts group - Data analysis & representation (DatAnaRep) involves processes of data collection, analysis and representation. The concept of generalization is also included in this group. Generalization is a way of gaining extra information by finding similarities between items (Krauss, Prottzman, 2017). Thus the concept of pattern recognition also belongs to this group. Because generalization can be defined as an activity that identifies patterns among individual sub-problems and simplifies them (Beecher, 2017). This group is mostly related to (CSTA & ISTE, 2011) vocabulary and framework for CT proposed by (Atmatzidou & Demetriadis, 2016).

The second group - Computing Artefacts (ComA) involves the creative aspects of computing and is developed based on six CT practises framework. It means the process of designing and developing computational artefacts as well as applying computing techniques to creatively solve problems. Decomposition (Decom) is the third group. This concept is identified in the most of the literature on CT and simplified means the process of breaking down the task into smaller manageable parts.

The fourth group – Abstraction (Abst) can be defined as the solution for a more general problem by ignoring certain details (Krauss, Prottzman, 2017). It also involves developing and representing models of the real world (Yadav & Hong & Stephenson, 2016).

The core concept of CT is Algorithms (Algo) and mainly devoted to the process of algorithm design. It also involves automation concept as it can be defined as the process of plugging pieces into an algorithm to help with a result (Krauss, Prottzman, 2017). Similarly, these two concepts were grouped together in (Duncan, Bell & Atlas, 2017).

The sixth group is Communication & collaboration (ComCon), the ability to communicate and work with others to achieve a common goal or solution are essential dimensions of CT. In order to successfully design, build, and improve computational artefacts the application of teamwork and collaboration, based on effective team practices, is important (College Board, 2017).

Practice that relates to the influence of computing and its implications on individuals and society is called Computing & Society (ConSoc). It can be seen also as responsible use of technologies by understanding the impact of computing, connection between society and computing. It involves concepts such as cybersecurity concern, privacy, self-protection in the Internet, potential beneficial and harmful effects of computing innovation.

The Evaluation (Eval) group involves the process of ensuring that solution, whether an algorithm, system, or process, fit for purpose (CAS, 2015). Simplified: evaluating the appropriateness of the proposed solutions and artefacts (College Board, 2017). It can be done systematically (through criteria and heuristics) make substantiated value judgements (Catlin & Woollard, 2014).

Digital competencies selected from the European Framework for the DC of Educators (DigCompEdu, 2017) were analysed. Facilitating Learners' Digital Competence area consist of five groups: Information and media literacy (InfMedLit), Digital communication and collaboration (DigComCol), Digital content (DigCon), Responsible use (ResUse), Digital problem solving (DigProSol).

Visualization of the interconnection between CT and DC components groups is presented in Fig. 1. These interconnections were developed from a matching of relationships between the details of components and features of CT and DC gathered from their respective literature reviews. First, component groups were identified and definitions collected for CT and DC. Then the related abilities of each component group were listed. Based on definitions and abilities, main concepts involved in each group were identified.

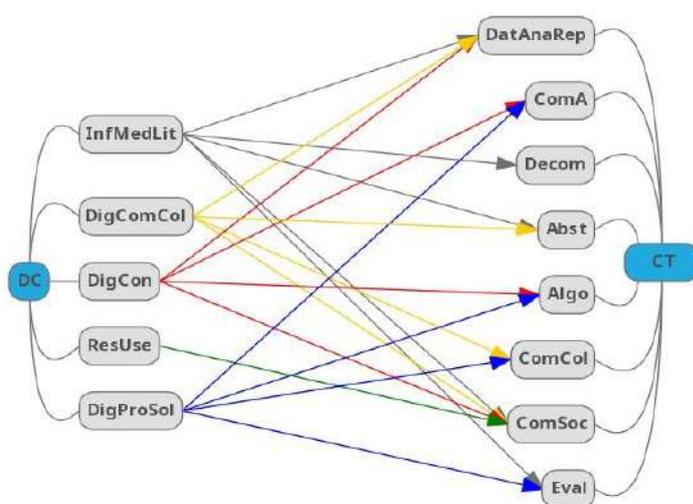


Fig. 1. The interconnection between DC and CT components

The interconnections presented in Fig. 1 are a visual portrayal of matches identified from analysis of interconnections between DC and CT components. It was quite clear that the Information and media

literacy group that involves abilities to manage information has interconnections with CT groups that involves similar abilities: Data analysis and representation, Decomposition, Abstraction, and Evaluation. It is interesting to note that it has interconnection with the Computing and Society group that relates to the influence of computing and its implications on individuals and society (College Board, 2017). As it was thought, the second group of Digital communication and collaboration which requires learners to effectively and responsibly use digital technologies for communication, collaboration and civic participation has interconnection with CT groups related to information exchange: Data analysis and representation, Communication and collaboration and, Computing and Society. Additionally, it connects to Abstraction group that is a way of expressing an idea in a specific context while at the same time suppressing details irrelevant in that context (Beecher, 2017) and looks like has nothing in common to communication or collaboration at a first sight. The Digital content group involves abilities to deal with content development strategies and licenses. It is not surprise that it is interconnected with four CT groups related to content procession, development strategies and social wellbeing: Data analysis and representation, Computing artefacts, Algorithms and, Computing and Society. Fourth DC group, Responsible use involves abilities to empower learners to manage risks and use digital technologies safely and responsibly has relation to very similar CT group - Computing and Society. Finally, the last Digital problem solving group is interconnected with four CT groups of Computing artefacts, Algorithm, Communication and collaboration, and Evaluation, involve abilities that are essential for problem solving. Thus it is quite clear that the last, Digital problem solving group, is interconnected with these four CT groups.

The presented interconnections of DC and CT groups were identified from similarities between abilities in each group. Patterns were spotted by looking for concrete descriptions, nouns and verbs that appeared in both cases. This enabled a simplification of them in order to identify interconnections. The findings from the literature review, generalisation and analyses are described below.

Information & media literacy group has 6 abilities. First Information & media literacy ability is to search information, thus ability to gather information is very similar as well as Abstraction ability to information filtering for solution development, and in order to articulate information needs the Computing and society ability of understanding connections between computing is needed. Second Information & media literacy ability to develop search strategies has relation with ability to gather an appropriate information and making sense of data. Additionally, in order to update search strategy, the expansion of existed solution to cover more cases could be used. Information filtering is needed for personal search strategies updating as well as ability to explain connections between computing concepts. The third Information & media literacy ability to adapt search strategies has the same interconnections with DC abilities as the second Information & media literacy ability. Fourth Information & media literacy ability is to evaluate the content. Thus, the analysis and comparison of content is related to patterns finding and conclusions making, as well as evolution of proposed solution and appropriateness, correctness justification. To organize content in digital environment (sixth Information & media literacy ability) is directly related to ability to depict and organize data in graphs, charts, images. The last listed Information & media literacy ability - information processing and organizing in structured environment has interconnection with two CT abilities. To identify the patterns and commonality between content could be one of the steps for information organizing as well as breaking down the content into smaller parts to easier manage them.

The second DC group Digital communication & collaboration has twelve abilities. Eleven of them is related to two CT groups, mostly to Communication & collaboration. Of course, ability to interact and ability to share content has connection to ability to exchange knowledge. Ability to understand appropriate digital communication means for a given context is could be related to ability to explain the meaning of a result in context. It is clear, that ability to know about referencing and attribution practices ability to participate in society through the use of digital services, ability to create and manage one or multiple digital identities, ability to protect one's own reputation) are connected to Computing & Society group ability to identify impacts of computing and describe connections between people and computing. This ability deals with privacy and security concerns, self protection in Internet, the way people connect, computing innovation (social, economic, cultural) effect, beneficial and harmful effects of computing (College Board, 2017). The sixth Digital communication & collaboration practise to seek opportunities

for self-empowerment and participatory citizenship could be related to practise of sharing the workload to collaborative effort by providing individual contributions. Seventh Digital communication & collaboration ability to collaboratively create content is interconnected with ability to collaborate in producing content and problem solving. Ability to foster a constructive collaborative climate by resolving conflicts and facilitating the contributions is related to eighth and ninth Digital communication & collaboration abilities because in order to foster the constructive collaborative climate one must be aware of behavioural norms and cultural diversity. Ability to deal with the data that one produces through several digital technologies, environments and services has connection to CT Data analysis & representation group's ability to deal with data in appropriate formats and Abstraction group's ability to explain how content is represented for computation.

The third group Digital content has six abilities. First is ability to manage digital content in different formats. It is related to ability to create content with a personal intent. Ability to express themselves through digital means and to be aware of copyright and licenses application is connected to ability to describe people and computing connection. Third ability is partly related to Data analysis & representation group's ability, because in order to modify, refine, improve and integrate information into an existing body of knowledge one could find similarities between items as a way of gaining extra information. On purpose to create original and relevant content, knowledge (fourth ability) the abilities to select appropriate techniques or information-management principles are useful. Sixth ability to perform a task or solve a problem by developing a sequence of instructions for computing system has four interconnections with abilities to use appropriate algorithmic principles, to identify sequence of events, to plug pieces into algorithm and to control a process by automatic means.

Responsible use group's all abilities that deal with safety and security understanding, protection and privacy, wellbeing in digital environment are interconnected with ability to identify impacts of computing, describe connections between people and computing. Digital problem solving group has seven abilities that are interconnected with nine CT abilities. First ability to identify and solve problems when using devices connected to ability to identify the processes of events. On purpose to identify, select, use and evaluate possible technological responses to solve task all Evolution group abilities are needed: evaluate proposed solution, correct errors, explain solution functions and justify appropriateness and correctness. Fourth ability to create knowledge by using technologies in innovative can be seen as a use of technologies with practical, personal or societal intent and appropriate management principles. Additionally, in order to create a high quality content, the review and revise of own work is needed. Also it is related to competences self-improvement needs recognition and self-development regulation. Sixth ability to support others in competence development has connection to ability to facilitate the contribution of a team member in a constructive way thus enable their improvement.

Conclusion and Discussion

The analysis of possible interconnections shows that both Digital Competencies (DC) and Computational Thinking (CT) have a lot of in common. Many abilities and competencies are overlapping. Only one of DC listed abilities (Digital problem solving - to adjust and customise digital environments to personal needs) has no direct connections to CT abilities. This ability is from personalization research area thus it could be concluded that it is out of the CT focus area at the moment. Digital problem solving is very important and huge area, so more detailed investigation in connection to CT is needed.

Five CT abilities were left without connections: ability to explain how abstractions are used in computation or modelling; ability to identify abstractions; ability to describe modelling in a computational context; ability to describe computation with accurate and precise language, notations, or visualizations; ability to summarize the purpose of a computational artefact. These abilities are related to abstraction and computation processes, actually, that are included in DC framework indirectly.

The discussed framework of CT components group and interconnections between DC and CT could be further developed, especially the quality research based on interviewing experts is needed. Evaluation study of experts' opinions of different countries for the proposed framework is in our future plans.

Limitation of this work is that there was no focus on age groups or educational course subject. As argued in (Shailaja & Sridaran, 2015) work, CT skills could be grouped by age, e.g. visualization, pattern

recognition and generalization can be learnt in K-2, abstraction and critical thinking in grade 6 to 8. Additionally, deeper investigation should be paid to integration of CT in different subjects. CT can benefit students studying in any area, academic and work lives and can be successfully taught in any subjects (Lockwood & Mooney, 2017). The similar questions can be applied to DC.

Acknowledgment

This research is/was funded by the European Social Fund under the No 09.3.3-LMT-K-712 “Development of Competences of Scientists, other Researchers and Students through Practical Research Activities” measure.

References

- Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, 55, 832–835.
- Atmatzidou, S., & Demetriadis, S. (2016) Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661-670.
- Beecher, K. (2017) *Computational Thinking: A Beginner's Guide to Problem-Solving and Programming*.
- Brennan, K., & Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada, 1-25.
- Catlin, D., & Woollard, J. (2014) Educational robots and computational thinking. In *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, 144-151.
- College Board (2017) *AP Computer Science Principles. Course and Exam Description*. College Board, NY.
- Commission of the European Communities (CEC) (2001) *Making a European Area of Lifelong Learning a Reality*. EUR-Lex.
- Curzon, P., McOwan, P. W. (2017) *The Power of Computational Thinking: Games, Magic and Puzzles to Help You Become a Computational Thinker*. World Scientific.
- Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014) Introducing teachers to computational thinking using unplugged storytelling. In *ACM Proceedings of the 9th workshop in primary and secondary computing education*, 89-92.
- Dagienė, V., & Stupurienė, G. (2016) Bebras - a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 5(1), 25-44.
- Dagiene, V., Sentance, S., Stupurienė, G. (2017) Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics // *Informatica*, Vol. 28, No 1, 23-44.
- Duncan, C., Bell, T., & Atlas, J. (2017) What Do the Teachers Think?: Introducing Computational Thinking in the Primary School Curriculum. In *ACM Proceedings of the Nineteenth Australasian Computing Education Conference*, 65-74.
- Ferrari, A. (2013) DIGCOMP: A framework for developing and understanding digital competence in Europe.
- Ferrari, A. (2012) *Digital competence in practice: An analysis of frameworks*. JRC technical report.
- Google for Education. Exploring computational thinking. Retrieved from <https://edu.google.com/resources/programs/exploring-computational-thinking/#!home>
- Grover, S., & Pea, R. (2013) Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Haseski, H. I., Ilic, U., & Tugtekin, U. (2018) Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. *International Education Studies*, 11(4), 29.
- Illomäki, L., Kantosalo, A., & Lakkala, M. (2011) What is digital competence? In *portal: Brussels: European Schoolnet*.
- International Society for Technology in Education (ISTE) (2016) *ISTE standards for students*. Eugene, OR.
- ISTE, CSTA. (2011) *Computational Thinking in K–12 Education leadership toolkit*.
- Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, 4(3), 583.
- Krauss, J. & Prottzman, K. (2017). *Computational Thinking and Coding for Every Student. The Teacher's Getting-Started Guide*. Corwin Press Inc.

- Lockwood, J., & Mooney, A. (2017) Computational Thinking in Education: Where does it fit? A systematic literary review. arXiv preprint arXiv:1703.07659.
- Lye, S. Y., & Koh, J. H. L. (2014) Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, 41, 51-61.
- Papert, S., (1996) An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1), 95–123.
- Pérez-Escoda, A., & Rodríguez-Conde, M. J. (2015) Digital literacy and digital competences in the educational evaluation: USA and IEA contexts. In *ACM Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality*, 355-360.
- Redecker, C. (2017) European Framework for the Digital Competence of Educators: DigCompEdu (No. JRC107466). Joint Research Centre (Seville site).
- Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. Retrieved from <http://royalsociety.org/education/policy/computing-in-schools/report/>
- Shailaja, J. and Sridaran, R., 2015. Computational Thinking the Intellectual Thinking for the 21st century. *International Journal of Advanced Networking & Applications*, May 2015 Special Issue, pp.39-46.
- Van Laar, E., van Deursen, A. J., van Dijk, J. A., & de Haan, J. (2017). The relation between 21st-century skills and digital skills: A systematic literature review. *Computers in human behavior*, 72, 577-588.
- Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715-728.
- Vuorikari, R., Punie, Y., Gomez, S. C., & Van Den Brande, G. (2016) DigComp 2.0: The Digital Competence Framework for Citizens. Update Phase 1: The Conceptual Reference Model (No. JRC101254). Joint Research Centre (Seville site).
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.
- Wing, J. M. (2006) Computational Thinking. *Communications of the ACM* 49(3), 33-35.
- Wing, J. (2011) Research notebook: Computational thinking—What and why? *The Link Magazine*, Spring. Carnegie Mellon University, Pittsburgh.
- Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6), 565-568.

AI Programming by Children

Ken Kahn, *toontalk@gmail.com*

Department of Education, University of Oxford, UK

Niall Winters, *niall.winters@education.ox.ac.uk*

Department of Education, University of Oxford, UK

Abstract

The idea of children constructing artificial intelligence programs goes back to the early days of Logo (Papert & Solomon 1971; Kahn 1975; Kahn 1977). After decades of little activity recent efforts to support students in making AI programs has come from Stephen Wolfram (Wolfram 2017b), Google (Google 2018a, 2018b), the Machine Learning for Kids website (Lane 2018), and the eCraft2Learn project (eCraft2Learn 2018b). Two technological developments underlie the feasibility of these efforts: (1) AI cloud services and (2) mainstream laptops and desktop computers that now can run sophisticated machine learning algorithms. And all of these developments can be made accessible in a web browser, thereby running on many platforms without the need to install software.

Given appropriate programming tools children can make apps and intelligent robots that rely upon speech and image recognition services. They can add custom capabilities to their programs by using machine learning training and prediction. In doing so they may learn about perception, language, psychology, and the latest empowering technologies.

We describe the addition of new programming blocks to the Snap! visual programming language (Harvey & Mönig 2010) that provide easy-to-use interfaces to both AI cloud services and deep learning functionality. Interactive learning materials have been developed and preliminarily trialled with students. We anticipate in future trials to observe children creatively using these new blocks to build very impressive programs. Children are likely to be even more motivated to program when the results are such capable programs.

Keywords

Visual programming; machine learning; block languages; Snap!, AI services; cloud services; speech synthesis; speech recognition; image recognition

Introduction to AI programming by children

From the early days of Logo research (Papert & Solomon 1971; Kahn 1975; Kahn 1977) there was interest in supporting children in creating artificial intelligence programs. The decision making, perception, learning, and natural language understanding in these programs was very simple due to the inabilities of the computers of those days. In the process of programming AI systems one naturally reflects on one's own thinking processes. This provided a very good fit for the constructionist ideas of learning through construction and reflection.

Among the many reasons for supporting AI programming by children are

1. Students may become better motivated and empowered to produce very capable artefacts
2. Students may learn about perception, reasoning, psychology, and animal behaviour in the process of building perceptive robots and apps
3. Students may learn about cloud services, machine learning, artificial intelligence, and other advanced technologies
4. Students may reflect more deeply upon their own abilities to hear, see, learn, and respond appropriately.

AI cloud services

Today's AI cloud services provide a new opportunity to support a new class of student AI projects – those that rely upon state-of-the-art AI “subroutines” (Kahn & Winters 2017). These services include speech synthesis and recognition as well as image recognition. Children can design and build impressive intelligent artefacts by composing and customising components provided by world-class AI teams.

Several companies are offering AI cloud services via a web connection. These include Google's machine learning services, IBM Watson cloud services, Microsoft cognitive services, and Amazon AI services. Many of these services include recognition services for obtaining descriptions of what is being spoken or seen. Other services analyse text for content, tone, and sentiment. They all include machine learning services that find patterns in data.

These are commercial services that cost a few dollars for a thousand queries. Fortunately for schools with limited budgets, free quotas are provided which allow a hundred recognition queries per day or a few thousand a month.

The service providers support accessing these services from many programming languages. Unfortunately, these are complex interfaces designed for use by professional programmers. In this paper, we show how to provide easy-to-use interfaces to these services, opening up their potential to children who are learning to program.

Students today are often doing physical computing projects involving micro-controllers such as Raspberry Pi, Arduinos, or Micro:bits. They are also often programming pre-built robots. In many cases these projects could benefit significantly from the ability to recognize what is being spoken or what is in front of a camera. For example, a student could build a robot that when it hears “push the red ball” will move to the ball and push it. This could be accomplished by sending the output from a microphone to an AI cloud service, picking out the keywords in the response, then repeatedly turning and sending images from a camera to a service until the response is that a red ball is in the image and then heading in the direction the camera is facing.

There is a long tradition of children programming language-oriented programs. In the early days of Logo children programmed poetry generators, silly sentence makers, chatbots, and more (Papert & Solomon 1971; Kahn 1975). The appeal of these kinds of projects increases when speech input and output replaces reading and typing, an area of research that has been neglected but can now be revisited to using the power of cloud-based AI services. In addition, student projects can use other AI services including sentiment analysis of what is spoken to respond in appropriate or amusing ways. This opens up the potential of AI to children in a simple and interactive manner, an emerging area of research we are exploring.

Machine learning

AI cloud services are created by large teams of experts using complex algorithms, massive data, and very large collections of servers. Systems are trained to recognise speech, images, video, sentiment, and more. Clients of these services have no control over the training. The services are black boxes to their users. In contrast, when children use machine learning software (Wolfram 2017b; Lane 2018) they need to provide the training examples before the system can do any classification or recognition. Programs can be created that respond to hand gestures, individual faces, or odd sounds that AI cloud services are not trained to recognise.

While there are cloud services for machine learning, we are not aware of any that have free quotas or are inexpensive enough to be used by school children. The Machine Learning for Kids website (Lane 2018) has a special arrangement with IBM for limited use of their machine learning cloud services. Consequently, the site requires registration and limits the number of projects allowed.

An alternative to cloud services appeared recently (tensorflow.js 2018). Tensorflow.js (formerly named deeplearn.js) is an open source JavaScript library that is able to support both training and prediction using deep neural nets. It runs in a browser and is accelerated hundreds of times by its use of graphical processing units (GPUs). These are now common on modern laptops, desktop computers, tablets, and

smartphones. For example, a high-end laptop can process up to ten images per second during training and make up to twenty predictions per second. Programs using tensorflow.js on devices that run at one tenth of that speed can still support a wide range of applications.

Machine Learning for Kids

The Machine Learning for Kids website (machinelearningforkids.co.uk) provides an interface where users can define a number of labelled buckets and fill them with images, text, or numbers.

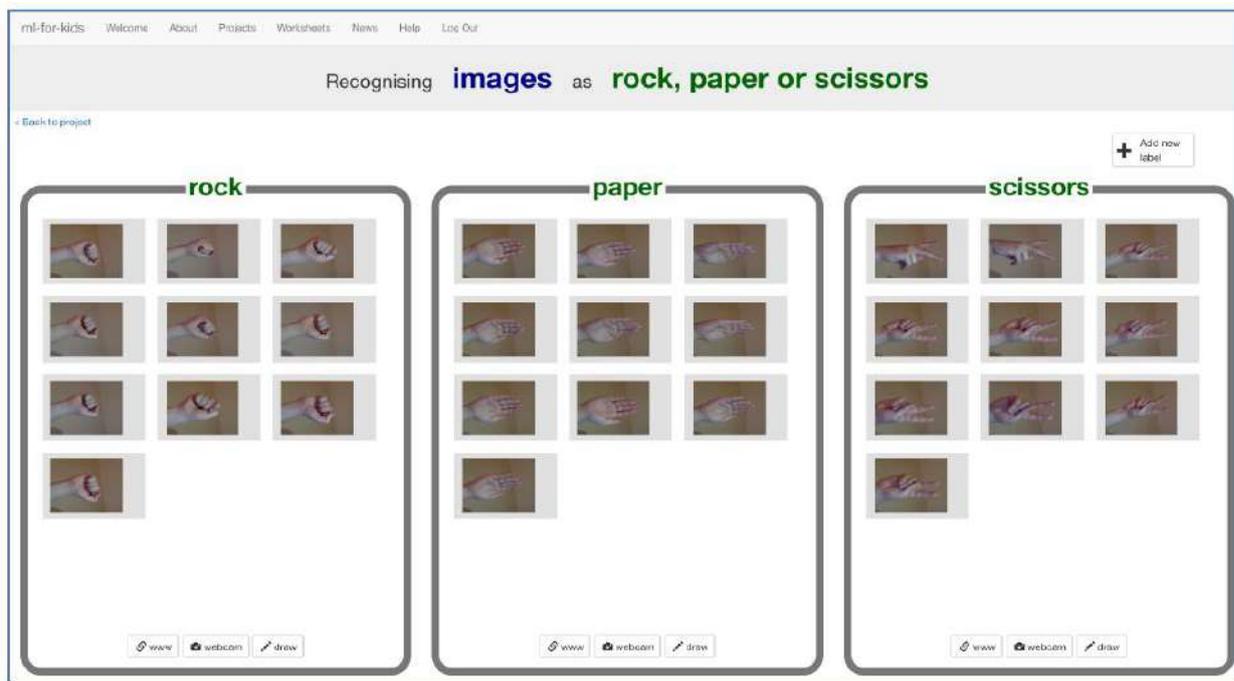


Figure 1. Training to classify a hand as rock, paper, or scissors at Machine Learning for Kids

After the buckets have been filled they are sent off to the IBM Watson servers to generate a model for classifying new images. It then adds new blocks to the Scratch programming language (Resnick et al 2009) that can be used to determine which category a new costume image belongs to. Another Scratch block is available to send new data for additional training.

The Machine Learning for Kids website has many suggested projects. For example, one implements sentiment analysis using textual data. Other projects involve learning to play Pac Man or Tic-Tac-Toe (Noughts and Crosses) using numerical training data. The Pac Man program learns to play better by adding additional training data whenever a human playing the game makes a move. The Machine Learning for Kids website and project ideas inspired the machine learning work described in this article.

Machine Learning in Wolfram Language

Stephen Wolfram wrote a blog post entitled *Machine Learning for Middle Schoolers* (Wolfram 2017b) about the new extensive section of his Wolfram Language textbook (Wolfram 2017a). (The Wolfram language is the language of the Mathematica system.) In this blog post he describes a variety of machine learning programming exercises including classification, clustering, text recognition, image recognition, feature extraction, feature spaces, and training. The textbook explains how neural nets can be defined and visualised.

Teachable Machine and Model Builder

Google recently released Teachable Machine (Google 2018c) which demonstrates image training and classification in a web page. It directly inspired the machine learning work reported below.

Google also recently released the Model Builder (Google 2018b). This is a website where one can design and train a deep neural net and run experiments on it. While one can learn a great deal about machine learning from this site, it doesn't provide a programming interface for creating apps that use the trained models.

AIY Projects

Google has begun to release a series of do-it-yourself AI kits they call AIY kits (Google 2018a). The kits are based upon Raspberry Pi computers. The first kit enabled one to build a box with a button that can do speech recognition and synthesis. It can answer questions the same way "OK Google" does. The second kit is for building a box that can do image recognition. Sample programs written in Python are provided.

Creating Child-friendly Programming Interfaces

Our efforts as part of the eCraft2Learn project are not to create a new programming environment for children, but instead to enhance existing ones. We have added speech input and output to ToonTalk Reborn (Kahn 2014) and to Snap! (Harvey & Mönig 2010). We have also added image recognition and machine learning to Snap!. This paper reports on our Snap! efforts.

Why was Snap! Chosen?

Snap! is a superset of the very popular children's blocks-based programming language Scratch (Resnick 2009). It is well-suited to our efforts because

1. It is a powerful language that supports first-class data structures and functions
2. It is easy to define new blocks using JavaScript without touching the source code
3. It runs in every modern browser
4. There are versions that connect to Arduinos and Raspberry Pis

Speech synthesis

All the popular browsers except for Internet Explorer support the Speech Synthesis API. (There is a problem with Chromium that no voices are installed.) The API utters the provided text with control for the pitch, rate, volume, language, and voice. An issue arises because the API instructs the browser only to begin speaking. The new Snap! block that invokes this API returns at once and yet some projects need a way to respond to the speech finishing. This was accomplished by passing to the Snap! block an optional function that is called when the browser signals the event that the speech has ended.



Figure 2. Simple speak command takes a text argument and an optional continuation function

More advanced users can use a full-featured 'speak' command that exposes most of the Speech Synthesis API functionality. While it is more difficult to use, all but the first parameter are optional and can be ignored. During testing students were clearly amused by entering different values for the pitch, rate, and voice. The underlying API expects the language parameter to be given as a BCP 47 language tag. This is a string that specifies the language and dialect. E.g., fr-FR is French as spoken in France while fr-CA is the Canadian dialect. This is not very user friendly. We addressed this by providing a Snap! reporter that supports search terms for installed voice names (e.g., "English", "UK", and "Female" in Figure 3). We also defined a way to change the default language for speech synthesis and recognition by giving the name of the language in the language or in English.

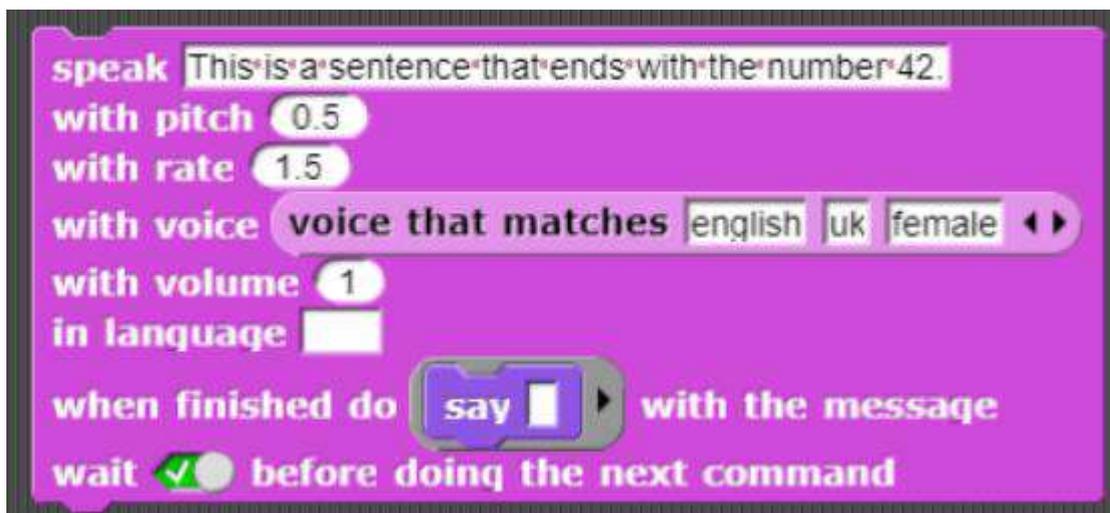


Figure 3. Advanced speak command that exposes nearly all the functionality of the Speech Synthesis API

Speech recognition

While speech recognition is part of the Web Speech API, as of this writing only Chrome and Opera support it. However, AI cloud services for speech recognition are available from Google, Microsoft, IBM, and others.



Figure 4. Simple speech recognition block for receiving text recognised and errors

(Empty fields here are shorthand for the argument passed to the command. The first one is what was heard and the second is the error message.)

The asynchronous nature of recognition services forces a reliance upon continuations (also called “callbacks”). Continuations are ideal from a technical point of view; however, we are concerned that student programmers may find them difficult. However, when students learn that the two fields of the listen command are simply places to put commands that run when and if the speech is recognised or when an error occurs. The combination of handling failures, errors, multiple outputs, and the asynchrony make the use of continuations more appropriate than Snap! reporters.

As an easier alternative to the use of continuations, we also implemented a block using the ‘listen’ block that supports event broadcasting and uses global variables:



Figure 5. A block that broadcasts speech recognition results

The underlying complexity is hidden (but available for ambitious students) so that one needs only to call the ‘broadcast speech recognition results’ block and then receive broadcasts when something was heard and read a global variable containing the last thing spoken. For example, the following program

repeats what was spoken, prefaced with “I think I heard”. Besides a questionable reliance upon global variables this technique makes it very difficult to respond to an utterance that wasn’t expected with a response such as “I don’t understand ...”.



Figure 6. Using broadcasting to respond to recognised speech

For more advanced uses of speech recognition, a user may want access to partial results as one is speaking, to receive alternative interpretations of what was said along with their confidence scores. And one may want to specify which language is expected. The following block provides this functionality. In Figure 7 while speaking partial results are displayed in a thought bubble (unless there is an error). French is expected. Up to five alternative interpretations of what was said will be displayed.

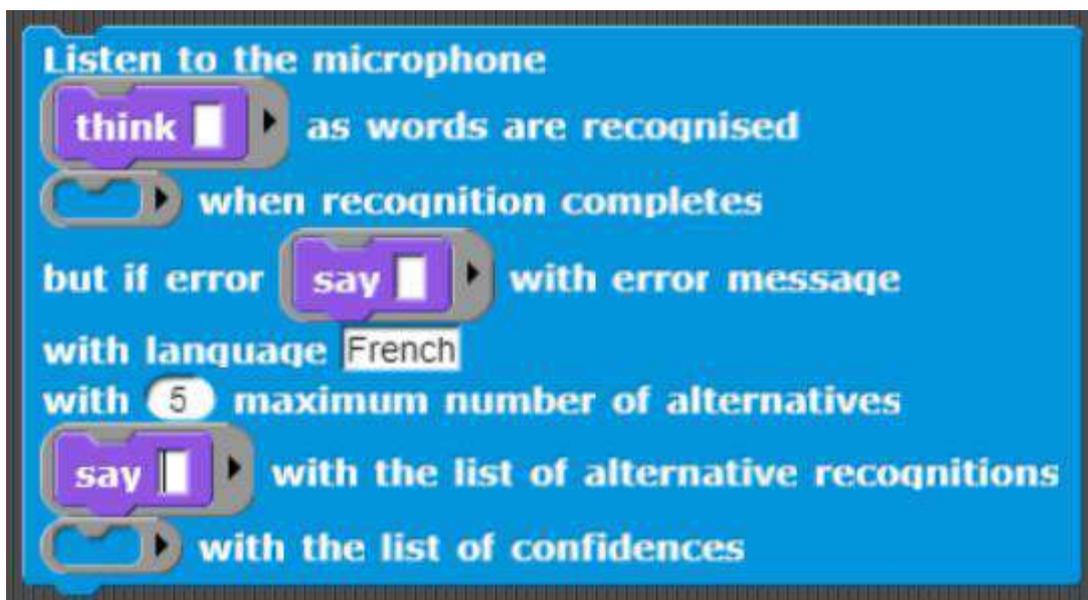


Figure 7. A full-featured speech recognition block

Image recognition

There are no standard APIs for image recognition so the block we implemented supports the Google, IBM, and Microsoft APIs. The simple image recognition block has a parameter specifying which cloud provider, a continuation that will receive a description of the image, and a flag as to whether the image should be displayed.



Figure 8. A simple block for obtaining a list of labels of what is in front of the camera

The responses from the vision recognition services are tables containing tables sometimes containing tables. And each provider has different structures. We provide blocks for accessing all the information

a vision service provides. For example, The Google vision API supports more than labelling what is in the image. One can access text recognition, face detection, and image properties results as well.

API Keys

All the AI cloud vision services require that requests be accompanied by API keys. These keys are easy to obtain and entitle the user to a moderately generous free quota of requests. It is not wise to share widely a project that contains API keys since the free quota will be exhausted quickly. The first solution we implemented was that the keys are provided as URL parameters. This wasn't very child friendly and became too clumsy for projects that rely upon more than one AI service provider. We eventually settled on using Snap! reporters to hold the keys. Our Snap! blocks use keys if provided and if they are missing offers to take the user to a page documenting how to obtain keys.

Machine Learning Snap! Blocks

At the time of this writing the machine learning Snap! blocks are limited to images (and a crude learning algorithm for audio) The underlying technology can be enhanced to support audio, video, text, and data. The interface to machine learning requires a second browser tab in addition to Snap!'s. Technically this is necessary since the machine learning software uses the GPU and Snap! uses it for graphics via WebGL. It is also motivated from a user interface perspective since the second tab is well-suited for both training and prediction feedback.

The following command launches a tab for training whether you are leaning to the left or the right. It overrides the default text on the page.

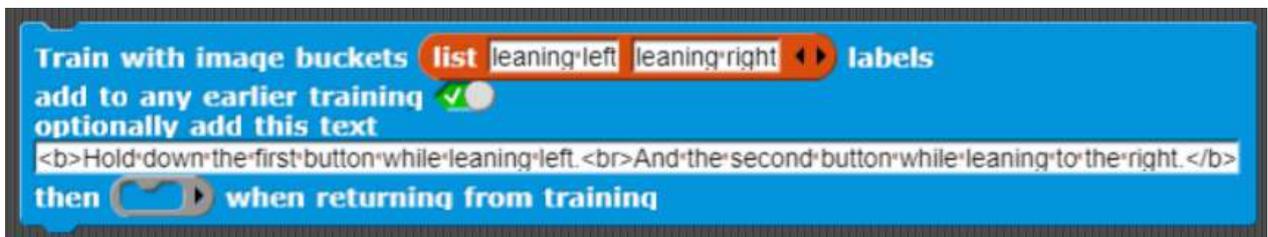


Figure 9. Launching a training tab

The tab created includes buttons for adding new images to the training. It also provides feedback as to how confident it is that it can label the current image.

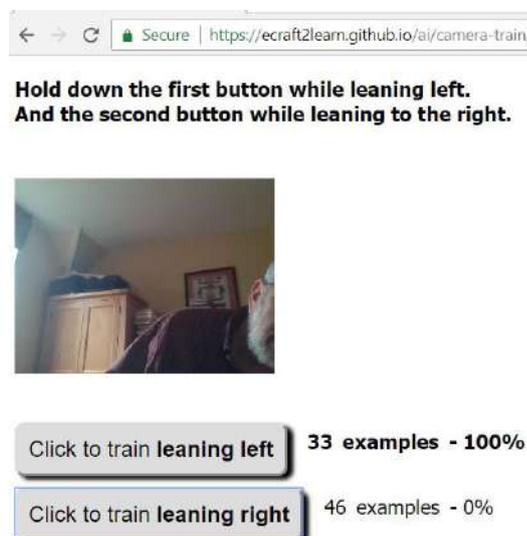


Figure 10. A machine learning training window

Upon returning to the Snap! tab one can send an image from the camera to the training tab to obtain a list of the confidences that each label applies to the image.

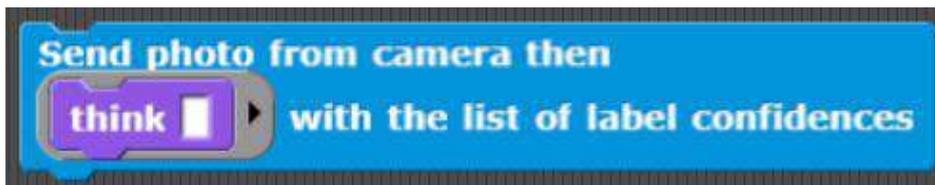


Figure 11. A block to obtain label confidences

An alternative to training using the camera is to use blocks that use the costumes of the Snap! sprites.

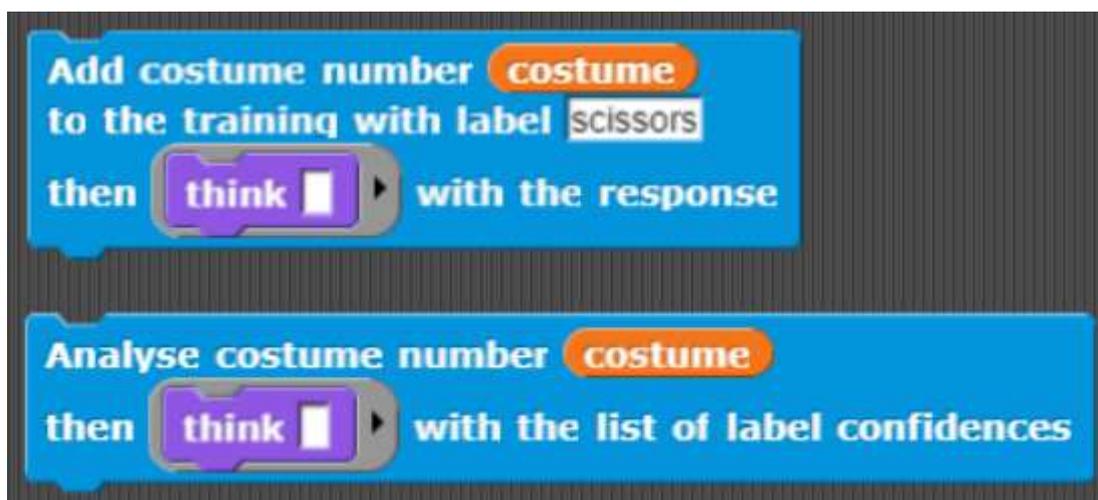


Figure 12. Blocks to use costumes for training and prediction

A block has been defined that uses the 'Add costume ...' block to send all the costumes of a sprite to the training tab. These costumes can be drawn, imported, or captured by a camera. Currently the underlying library providing the machine learning capabilities (tensorflow.js 2018) has no general way to save a model after training so applications need to run training before doing classification.

Student and teacher guide to AI programming

An extensive interactive guide to using all of these new Snap! blocks has been created. There is a student and a teacher version (though they are over 90% identical) (eCraft2Learn Project 2018b). The guides are web pages with dozens of Snap! projects embedded as iframes. Some are simple interactive introductions to a single block while some are more complex demonstration programs.

In addition to the technical content the guides include discussions of how the underlying technology works as well as societal impact. They also include exercises and project ideas.

The guides can be viewed via a local web server. This is useful when the Internet connection is slow or intermittent. If there is no Internet connection about half the guide's interactive elements are still functional. The machine learning chapter works fine locally without a network connection.

Internationalisation

Google Translate has been integrated with the AI programming guides, documentation, and the training tab interface. The speech synthesis and recognition blocks by default use English but there is a block for setting the default to any of the many languages that Chrome supports. Snap! itself supports several

dozen languages. Unfortunately, the labels and help messages of the AI blocks themselves are not translated.

Sample programs using AI blocks

Currently ten sample programs are available that use these new blocks. They are designed to be readable and editable by students. Many of them implement only part of what a full app would contain leaving the missing parts for students to add.

1. Speak with random pitch, rate, voice, and language (Speech synthesis)
2. Speak commands to a sprite (Speech synthesis and recognition)
3. Generate funny templated sentences (Speech synthesis and recognition)
4. Generate template stories (Speech synthesis and recognition)
5. Talk to Wikipedia (Speech synthesis and recognition)
6. Report what is in front of the camera (Speech synthesis and image recognition)
7. Speak what is in front of the camera when a cloud service is asked to (Speech synthesis, speech recognition, and image recognition)
8. After training watch the turtle move depending on which way your finger is pointed or sounds you make (Machine learning)
9. After training watch the turtle move left or right depending on which way you lean (Machine learning)
10. Play a Rock Paper Scissors game by configuring your hand in front of the camera (Speech synthesis and machine learning)

All of these programs and the new blocks are available at (eCraft2Learn Project 2018b).

Using NetsBlox (NetsBlox 2018), a Snap! variant that supports multi-player projects, and together.js (together.js 2018), a library for creating multi-player JavaScript programs, we created a two-player version of Rock Paper Scissors. In the first phase both players connected via the Internet can train the same model to recognise whether a hand is showing rock, paper, or scissors. In the second phase the program says “1, 2, 3, go” and then the trained model and cameras on both computers are used to classify each players moves. Speech synthesis is used to inform the players of the outcome.

Field testing

The speech input and output blocks have been tested with 25 Singaporean undergraduate students, 18 children (aged 7 to 13, Sri Lankan and Singaporean) in afterschool programs and 40 Indonesian high school students. This preliminary testing has been very encouraging. The new Snap! blocks were easily understood and the users indicated that they enjoyed adding speech to their programming projects. 12 of the 18 children and all the high school students were also introduced to machine learning Snap! blocks and mastered several machine learning programming exercises.

A formal study of 40 Indonesian high school students using speech synthesis, speech recognition, and machine learning has been submitted for publication. Teachers in Finland and Greece have reported successfully introducing the Snap! AI blocks.

Conclusion and Discussion

We began by describing the history and current efforts to support AI programming by children. We then presented programming constructs we have developed that are suitable for use by beginners for speech synthesis, speech recognition, image recognition, and machine learning (of images). Artificial intelligence is much broader than this and includes common sense, planning, reasoning, understanding language, unsupervised learning, translation, and more. The research reported herein is just the start of an effort to give children the ability to construct AI applications.

The goal of this research is to demonstrate that AI programming need not be limited to those with advanced degrees. Even young children can be empowered to creatively use AI programming to make apps that speak, listen, see, and learn.

Acknowledgements

This research was supported by the eCraft2Learn project funded by the European Union's Horizon 2020 Coordination & Research and Innovation Action under Grant Agreement No 731345.

References

- Tensorflow.js (2018) <https://js.tensorflow.org/>.
- eCraft2Learn Project (2018a) <http://www.project.ecraft2learn.eu/>.
- eCraft2Learn Project (2018b) <https://ecraft2learn.github.io/ai/>.
- Google (2018a) AIY Projects. <https://aiyprojects.withgoogle.com/>.
- Google (2018b) Google Model Builder. <https://deeplearnjs.org/demos/model-builder/>.
- Google (2018c) Teachable Machine. <https://teachablemachine.withgoogle.com/>.
- Harvey, B., Mönig, J., (2010) Bringing “No Ceiling” to Scratch: Can One Language Serve Kids and Computer Scientists? In Proceedings: Constructionism, Paris, France.
- Kahn, K. (1975) A Logo natural language system. Technical report, MIT AI Lab, LOGO Working Paper 46.
- Kahn, K. (1977) Three Interactions between AI and Education. *Machine Intelligence 8*.
- Kahn, K. (2014) ToonTalk Reborn: Re-implementing and re-conceptualising ToonTalk for the Web. In Proceedings: Constructionism, Vienna, Austria.
- Kahn, K., Winters, N. (2017) Child-friendly programming interfaces to AI cloud services. In Proceedings: EC-TEL 2017: Data Driven Approaches in Digital Education, 10474, 566-570.
- Lane, D. (2018) Explaining Artificial Intelligence. *Hello World 4*. Raspberry Pi Foundation. <https://helloworld.raspberrypi.org/issues/4>.
- NetsBlox (2018) <https://netsblox.org/>.
- Papert, S., Solomon, C. (1971) Twenty Things to Do with a Computer, MIT AI Lab, <http://hdl.handle.net/1721.1/5836>.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009) Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67 DOI=<http://dx.doi.org/10.1145/1592761.1592779>.
- Together.js (2018) <https://togetherjs.com/>.
- Wolfram, S. (2017a) *An Elementary Introduction to the Wolfram Language, Second Edition*. Wolfram Media.
- Wolfram, S. (2017b) Machine Learning for Middle Schoolers. Stephen Wolfram blog. <http://blog.stephenwolfram.com/2017/05/machine-learning-for-middle-schoolers/>.

VISURATCH: Visualization Tool for Finding Characteristics of Teaching and Learning Process of Scratch Programmers

Nobuko Kishi, *kishi@tsuda.ac.jp*,

Mari Yoshida,

Minori Yoshizawa,

Tsuda University, Japan

Aoi Yoshida, *aoi@si.aoyama.ac.jp*

Aoyama Gakuin University, Japan

Abstract

We are developing an interactive visualization tool, VISURATCH, for Scratch projects using Python and D3.js. This tool collects a set of programs, or projects, from the Scratch site and reveals each program's characteristics, such as the number of blocks, as a line chart and a heat map. By visualizing a set of programs in chronological order of creation, it enhances an instructor's or a learner's ability to quickly find the characteristics of their teaching and learning process. The tool's visualization model is based on the Visual Information-Seeking Mantra: Overview first, zoom and filter, then details-on-demand. We observe that the tool shows some characteristics of a teaching plan when applied to a set of sample programs shared in a Scratch studio. Moreover, the tool shows some characteristics of a learner, such as the pattern of programming blocks frequently used, when applied to a set of programs specified by the learner's id.

Keywords

information visualization; Scratch programming; computer science education

Introduction

Teaching children to program or encouraging them to learn to program is becoming a major trend in education. However, there are no simple methods for assessing children's ability with programming or computational thinking. In software development classes for adults, we can often evaluate students' skills based on the programs they create. In programming classes for children, we need to focus on what they learn, rather than what they create. In the constructionist approach to learning programming, we encourage children to create programs meaningful to them, but we need to assess various aspects of children's learning, not just the programs that they create.

Brennan and Resnick (2012) proposed a general framework for the assessment of computational thinking. In their paper, they propose three key dimensions: computational concepts, computational practices, and computational perspectives for assessing computational thinking. They also described three approaches—project analysis, artifact-based interviews, and design scenarios—and six suggestions for further research.

Our work is motivated by their work, based on their suggestions #3: Illuminating processes. Conversations with the learner and real-time observation of their activities are mentioned in their paper as possible methods for studying the learning processes. However, it is time-consuming to use those methods with many children, or when instructors have limited time.

We have decided to try a well-known technology, information visualization, to ease the task of finding the characteristics of the process of children learning to program. Scratch is an online programming tool for children, or for people of any age, developed at the MIT Media Lab. (Resnick, 2009). Scratch projects are shared by Scratch users at the Scratch site <http://scratch.mit.edu>. Various project metadata such as a project creation date, modification date, publication data, and the user id of the creator are also

available at the Scratch site. By obtaining a set of projects created by a single user from the Scratch site and putting them in chronological order, we can get a series of programs in order of creation date by the same user.

In this paper, we first describe the information visualization model we used with VISURATCH. Next, we describe its system design and its main function. After that, we report on the findings we obtained with the tool in the case studies. Although VISURATCH is still in the development stage, we made several interesting observations with the sample programs used in Scratch books and the programs created by young learners. Finally, we discuss the possibility of program visualization in assessing the programming skills of young learners and future works.

Information Visualization Model for Scratch Projects

Information visualization or data visualization is a technology to explore large amounts of abstract data by creating images or diagrams to represent a fact or communicate a message. The term *information visualization* originates from the works at Xerox PARC (Card, Mackinlay, Shneiderman, eds., 1999). There are a variety of techniques for information visualization, and Shneiderman summarizes it with the Visual Information-Seeking mantra: *overview first, zoom and filter, then details-on-demand* (Shneiderman, 1996). We have decided to apply this mantra to a set of Scratch projects that are created by a single learner, or to those that are included as sample programs in a Scratch book or online tutorial..

A Scratch project

The Scratch 2.0 programming environment is available as an online programming tool at the Scratch site: <http://scratch.mit.edu>. The Scratch projects, programs written in Scratch, are created by snapping graphical programming blocks together at the Scratch site. As of March 31, 2018, the number of Scratch projects shared was 30,302,556. These Scratch projects are stored in JSON format and available through the Scratch API. Figure 1 shows some of Scratch's graphical programming blocks, which are part of a Scratch project. Figure 2 shows JSON data corresponding these blocks. Each Scratch project also contains graphics data and sound data, in addition to the Scratch program. Figure 3 shows an online programming screen for a Scratch project.



Figure 1. Scratch programming blocks

```

{"objName": "Sprite1", "scripts":
  [[85,84],
    ["whenGreenFlag"],
    ["say duration:elapsed:from","Watch me dance!",2],
    ["doRepeat",10,
      [
        ["forward",10],
        ["playDrum",1,0.25],
        ["forward",-10],
        ["playDrum",4,0.25]
      ]
    ]
  ],
  []
}

```

Figure 2. Scratch programming blocks in JSON format

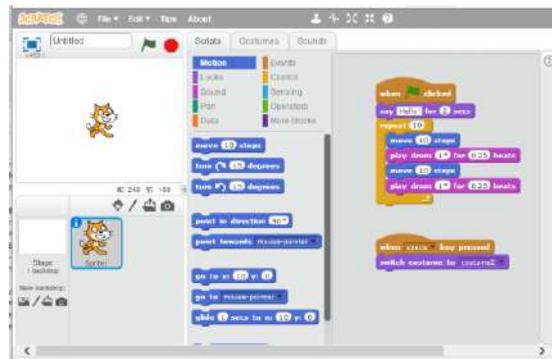


Figure 3. A Scratch project

The Scratch site and studio

At the Scratch site, the metadata for Scratch projects are also available. By metadata, we mean the data about the program author, the program creation date, the modification date, and the date when the program was shared. The site also offers mechanisms for sharing projects. When a user id is specified, the site shows a list of programs created by that user and made publicly available by the user himself/herself. Figure 4 shows an example of the screen that is returned after a user id is specified. It shows one featured project and five additional projects from 18 projects shared by a single user. The Scratch site also has a studio mechanism that can hold a set of projects created by different users.

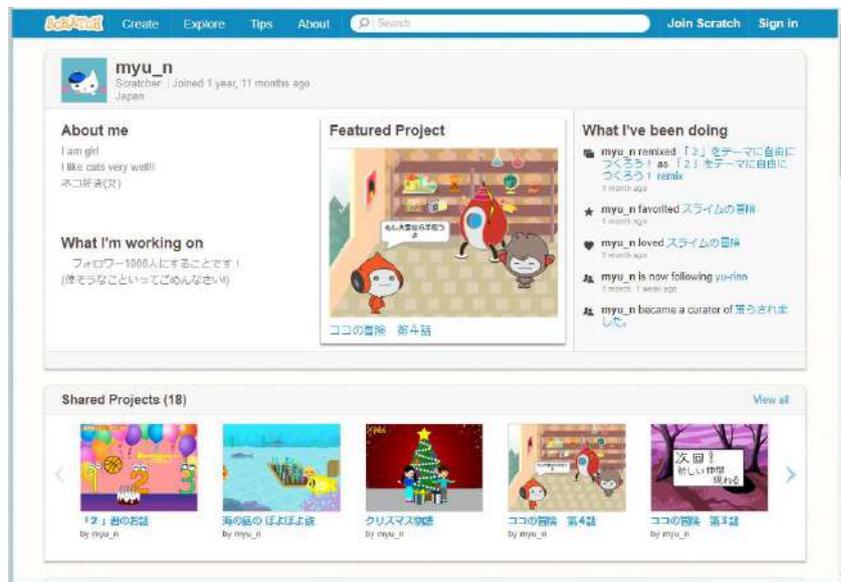


Figure 4. Projects created by a single Scratch user

Overview of Scratch projects.

By using the Scratch API and the Scratch site, we get two sets of Scratch projects. With a Scratch user id, we get a set of projects created by a single user corresponding to that user id. With a studio id, we get a set of projects created by a group of users with a common interest, such as creating sample programs for a workshop.

Each Scratch program has the following metadata:

- “created” date: the date and time when the project was first created
- “modified” date: the latest date and time when the project was modified
- “shared” date: the date and time when the project was made publicly available on the community site

To observe the changes among programs over intervals of time, we decided to extract the following statistics to characterize the program.

- The number of each type of block
- The number of blocks in each category of block
- Total number of all blocks
- Total number of sprites

To get an overview of these data, line charts are used. A line chart shows a trend in the total number of blocks, or the number of blocks in each category of block over the dates of creation, modification, and sharing. To give a detailed view of each project, a heat map is used to show the use of each type of block. In addition, a list of the project title and the thumbnail image of the screen is shown so that the user can run the Scratch program easily. Left part of Figure 6 shows an example of the VISURATCH main screen. It shows a visualization of the Scratch projects created by a single user, corresponding to the ones shown in Figure 4. It shows a line graph in the upper part, a heat map in the bottom left, and a project list in the bottom right. When a project in the list is double-clicked, a Scratch project window, as shown in Figure 3, is opened and the program can be run in that window.

System Design

VISURATCH comprises two main components (Figure 5). One is an analysis tool written in Python. This tool obtains Scratch program data by scraping the Scratch site and by using the Scratch API. It then analyses the program data and sends the statistics of the program, such as the number of blocks and the program creation date, to a web browser in JSON format. The other is a visualization tool implemented in JavaScript. It runs on a web browser and uses D3.js, a JavaScript library for visualizing data (Bostok, 2017). The term D3 comes from Data-Driven Document. When the statistical data from the analysis tool are obtained as a document, the images of the data can be interactively modified. For example, scrolling and zooming of a line graph can be done interactively using the brush function (Figure 6).

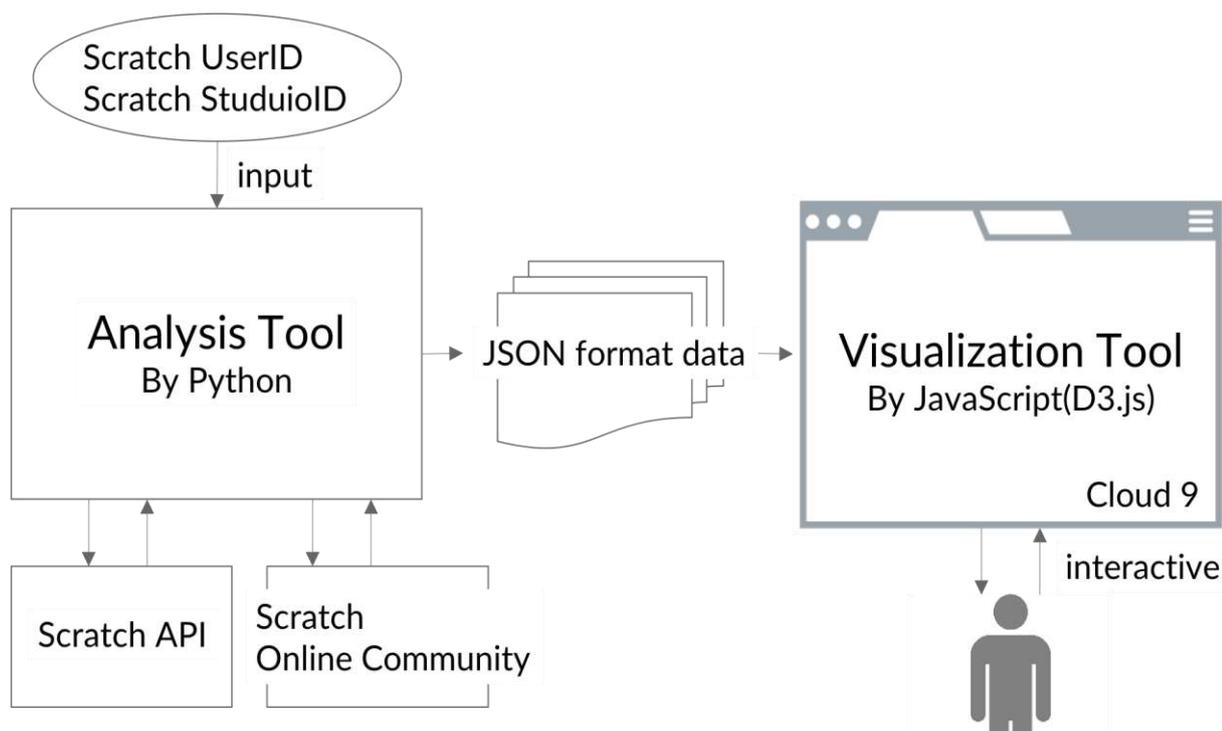


Figure 5. VISURATCH system design

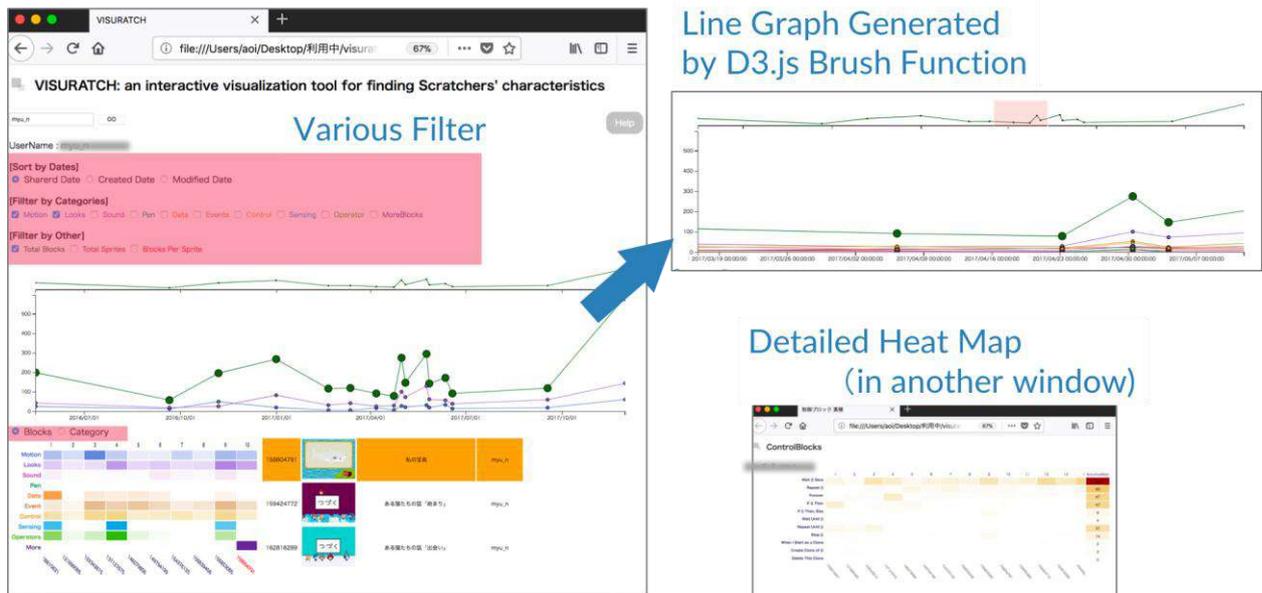


Figure 6. Zooming and scrolling of line charts using brush function in D3.js

Case Studies

To evaluate the effectiveness of VISURATCH, we have conducted three case studies. We first studied a model user, a hypothetical learner who creates sample programs exactly as given in a website or in a book. We then studied Scratch projects created by real learners of Scratch. Although we are aiming for visualization of a large set of Scratch programs in the future, we need to start with a small set of about 20 programs so that we can study each program in detail.

Case 1: Scratch tutorial at the Scratch site

The Scratch site has an online tutorial for beginners, which has a series of sample projects (Figure 7). We created all the sample programs and put them in a single Scratch studio. Figure 8 shows the visualization of these sample programs specified by its studio id. With the line graph we can see that the number of blocks increases over time, but each project is not always larger than the previous project. We also notice that some blocks are used rarely. The Pen category blocks are not used at all, and the More category blocks are not used until the end of the tutorial. This means the tutorial excludes a sample program for drawing graphics using x-y coordinates. It also means that the blocks for creating a function, which are the More category blocks, are not used in the beginning part of the tutorial.

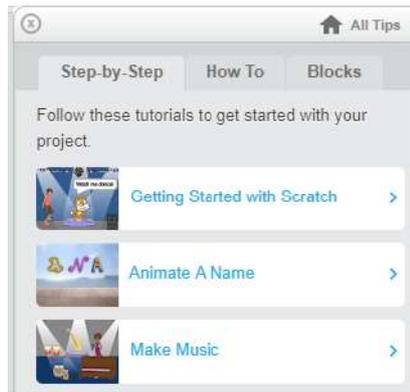


Figure 7. Scratch Tutorial at <http://scratch.mit.edu>



Figure 8. Visualization of sample projects in Scratch Tutorial

Case 2: Scratch book on creating a shooting game

Creating a shooting game is very popular among Scratch learners. A book was published in Japan aimed at having the reader learn Scratch by creating a shooting game (Sugiura, 2015). The book author made 16 projects from the book available in the Scratch site. Figure 9 shows the visualization of these 16 projects. We found that the number of the total blocks gradually decreased over time in the line chart, and the use of category blocks also gradually decreased in the heat map in chronological order of the date of sharing. Figure 10 shows the visualization based on the creation date and modification date. We presume that the author uploaded the final project, a complete shooting game, first, followed by the rest of the programs in reverse order of their appearance in the book. We also presume that the author was very careful that each sample project be created by adding a small set of blocks to the preceding sample projects.

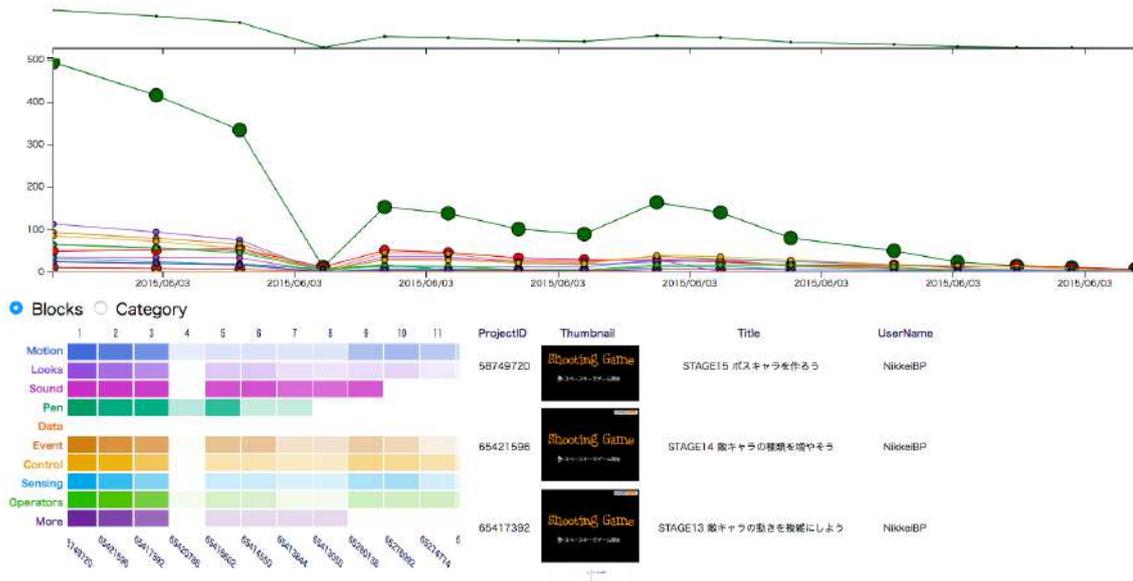


Figure 9. Sample projects in the order of their being shared

Sort by "Modified" Date

Sort by "Created" Date

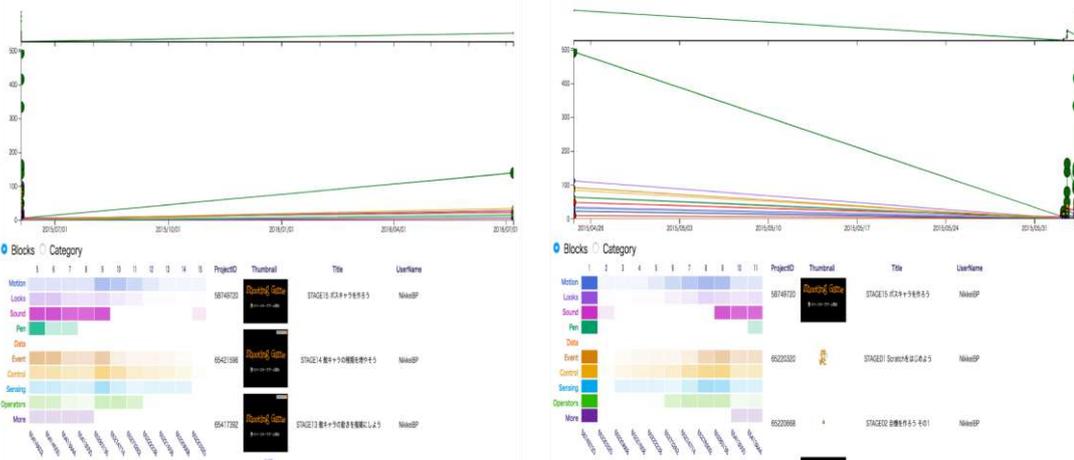


Figure 10. Sample projects in the order of "modified" date and "created" date

Case 3: A real learner of Scratch

A Japanese TV station, NHK, broadcasts a TV program to introduce Scratch to children in Japan (Japan Broadcasting Corporation, 2016). On that program, contests are regularly held. One of the contest prize winners was a girl with two years of experience with Scratch. She put 18 shared projects on the Scratch site. Figures 11 and 12 show the visualization of her projects. We applied a filtering function to the line chart to create three line graphs: a purple line for for the number of Looks category blocks, a orange line for Control category blocks, and green line for the total number of blocks (Figure 11). We observed that these three lines show almost the same trend. This means she used the blocks in the Looks and the Control category together quite often. Next, we used the detailed heat map of the blocks in these two categories. We found one block, *wait x secs*, in the Control category (Figure 12), and three blocks: *say x for y secs*, *show*, and *hide* in the Looks category (Figure 13) quite frequently. These four blocks were often used to change the scenes of the stories in Scratch projects. With this information, we can presume that she frequently created stories with Scratch. We confirmed this by running her projects on the Scratch site and by observing that many of her projects were story-oriented.

[Filter by Categories]

Motion Looks Sound Pen Data Events Control Sensing Operator MoreBlocks

[Filter by Other]

Total Blocks Total Sprites Blocks Per Sprite

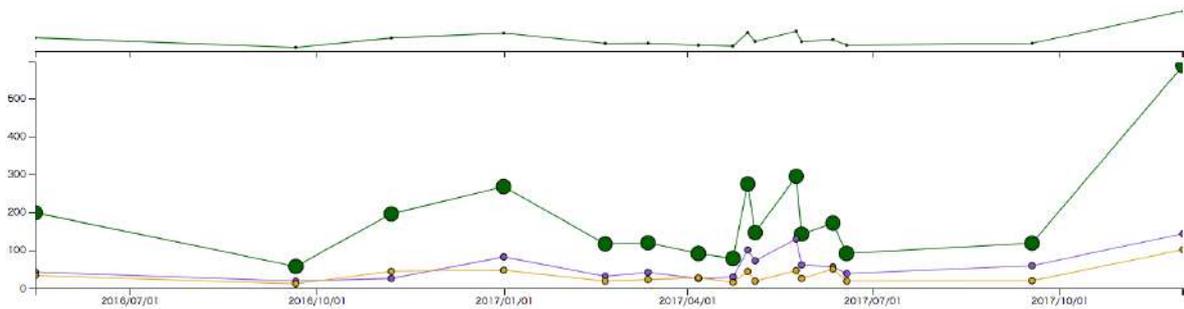


Figure 11. Filtered line charts

Detailed Heat Map "Control" Category Blocks

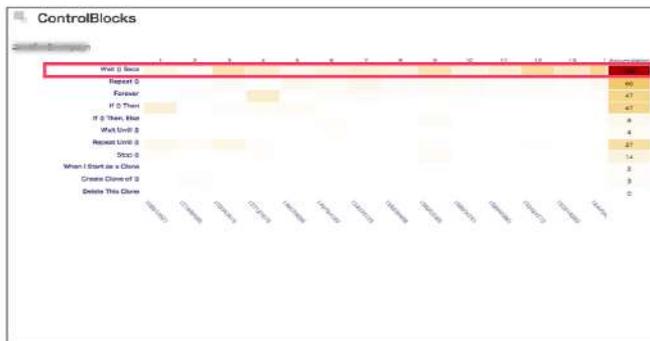


Figure 12. Control Category block found by the detailed heat map

Detailed Heat Map "Looks" Category Blocks



Figure 13. Looks Category blocks found by the detailed heat map

Conclusion and Discussion

We have developed a preliminary version of an interactive visualization tool, VISURATCH, for Scratch projects using Python and D3.js. We also conducted case studies to evaluate its effectiveness. The initial case studies suggested that VISURATCH could help an instructor or a learner of Scratch to identify the characteristics of teaching and learning processes.

After conducting several more case studies, we plan to improve VISURATCH and make it publicly available as an online tool. By doing so, we can conduct a data-based evaluation of our visualization model at large scale. We hope to improve VISURATCH so that it can be used easily by both instructors and learners to understand the learning process.

We also hope that it can serve as a basis for research on educational data mining in the future. Scratch is excluded in the literature review and case studies paper on educational data mining and learning analytics (Ihantola, P. et al., 2015), probably because Scratch is not a textual language and is intended for children to learn computational thinking, not programming. However, it can be a good starting point for educational data mining because it offers a large set of program data with an IDE available to everyone.

References

- Brennan, K., and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the American Educational Research Association (AERA) annual conference
- Card, S.K, Mackinlay, J.D. and Shneiderman, B. (Eds.), (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
- Ihantola, P. et al. (2015). Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies, Proceedings of the 2015 ITiCSE on Working Group Reports, July 04-08, Vilnius, Lithuania
- Japan Broadcasting Corporation, NHK (2016). Why? Purograminngu (in Japanese), <http://www.nhk.or.jp/gijutsu/programming/>
- Resnick, M. et al. (2009). Scratch: Programming For All. Communications of the ACM, Vol. 52 No. 11, Pages 60-67
- Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of the IEEE Symposium on Visual Languages, pages 336-343, Washington. IEEE Computer Society Press, <https://www.cs.umd.edu/~ben/papers/Shneiderman1996eyes.pdf>
- Sugiura, M. (2015). Scratch de hajimeyou, purograminng nyumon (in Japanese), Nikkei BP, Tokyo, JAPAN

Curriculum Intervention for Learning Programming in Python with Turtle Geometry

Eva Klimeková, klimekova@fmph.uniba.sk

Department of Informatics Education, Comenius University, Bratislava, Slovakia

Abstract

Programming as part of computing education at upper secondary schools in Slovakia has a long tradition. When a new trend of teaching programming in Python emerged, many schools adopted it. In this paper we focus on deeper understanding of teaching programming basics in the Python language. Our aim is to find out how we can help to teacher for easier transformation to new language. We designed a curriculum intervention for learning programming in Python in which we decided to exploit the motivation of turtle geometry. This supports learning by exploring, creating and discovering. By means of it pupils learn the basic concepts and constructs as variables, loops, functions and conditions. We developed, implemented and verified our materials iteratively, using design research. In this paper we also present the process of development of our curriculum intervention from the first steps of collecting initial insight to implementing the materials into practice at upper secondary schools in Slovakia.

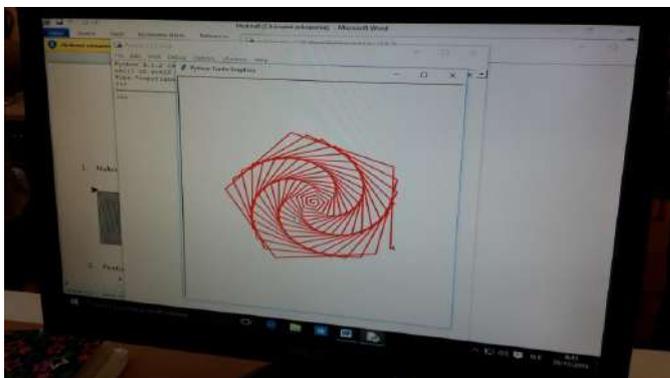


Photo of a pupil's "rose" created by using turtle geometry in Python

Keywords

Python; turtle geometry; curriculum intervention

Introduction

Computing in upper secondary school education (pupils aged 15 to 18) in Slovakia has a relatively strong tradition. Despite the fact that educational reforms often modified the content of the general-academic subject Informatics, programming is an integral part of this subject for more than twenty years. The aims of teaching programming are the development of algorithmic thinking, problem solving skills and the acquisition of programming skills. In Slovakia, National curriculum does not explicitly specify a particular programming language, therefore the teachers choose the programming language. Currently, the most widely used programming language in Slovakia is Pascal (Delphi or Lazarus), previously Logo was used. In the last few years Python is more commonly amongst the languages recommended for teaching programming basics. Research shows that Python is suitable as the first programming language for pupils at upper secondary school (Atteq, 2014; Goldwasser and Letscher, 2008; Krpan and Bilobrk, 2011). Manilla and De Raadt (2006) after extensive analysis of programming languages concludes that "...the most suitable languages for teaching, Python and Eiffel, are languages that have been designed with teaching in mind". According to Grandell (2006), the benefits of Python fully support the aims of teaching programming, the development of algorithmic thinking, and that the characteristics

of this language are consistent with the requirements for the programming language for upper secondary schools. Therefore, more and more universities and secondary schools switch to teaching programming basics in Python (Hromkovič, 2016; Blaho, 2016; Belan, 2013). However, when we started our research there weren't any official textbooks for learning programming basics in Python in Slovak language. Teachers had to develop materials for their own use, where these materials are aimed for specific group of pupils (Belan, 2013), or using very similar or identical examples as the ones used for Pascal (Delphi / Lazarus) (Kučera, 2016) based on working with the graphical canvas using components as an input fields or buttons. However, Python also offers other techniques, such as turtle geometry, which promotes learning by exploration and discovery or using command mode which gives immediate feedback for pupils.

Before designing a curriculum intervention we examined the currently most used textbooks in our country, textbooks of the Logo and Pascal (Delphi) languages: Algorithms with Logo and Programming in Delphi and Lazarus (Varga, 1999; Blaho, 2012). Both textbooks' major topic is the work with graphics while other terms and concepts are explained through it so pupils can visually see the solution of the tasks. We have found out that the order of some of the first topics coincide in both textbooks, which means that loops, conditions and procedures can be taught both by turtle geometry and graphical canvas. When designing our materials for teaching programming basics in Python for upper secondary schools, we will base it on the examined textbooks as much as possible and use the tasks created by the experienced and respected authors of these textbooks.

When designing methodological materials, it is advisable to keep in mind the stages of the cognitive process defined by Hejný (2009), and also the advice of Linn and Dalbey (1988), whom suggests the following learning process of programming: 1. learn elements one at a time; 2. learn design skills by combining old and new strategies, and 3. learn general skills to solve problems. In our design process we also keep in mind the advice for designing the right materials for learning. Salanci (2011) points out that the correct textbooks emphasize that the learner collects the experience, while textbooks are not encyclopaedic they gradually explain the basic concepts of programming; emphasizing motivating examples and tasks, tracing and the gradual discovery of the algorithmic rules. As we follow the Fuller's spiral taxonomy of cognitive learning (2007) we proceed from basic programming concepts.

Methodology

In our research we focus on the teachers. Our aim was to find out how we can help to teacher for easier transformation to a new language. The pre-survey showed us that teachers need material in Slovak language compatible with our learning system (Mészárosová, 2015). We decided to develop a comprehensive curriculum intervention for the Python programming language to teach the basics of programming. Our materials will include remarks, tips, and notices that will make it easier for teachers to transition from the Pascal programming language, which they have experience with, to Python. Our materials will differ from most of the materials for Python by focusing on algorithms and concepts, but not on characteristics of the language. Our goal was to minimize the amount of commands pupils need to learn and to gain new programming concepts and knowledge so we chose turtle geometry. We also wanted to create material that uses didactic methods and tasks that teachers already know and make pupils learning easier by using the smallest number of commands.

Our curriculum intervention consists of methodical materials for teachers and worksheets for pupils. These materials contain educational goals, developed competencies by students, selected methods, content of lessons with solutions of the tasks and recommendations for the teacher and they are in accordance with the National Educational Program of Slovakia. Therefore, we had a research question: *What should be specific objectives, content and form of curriculum intervention for teaching programming basics in Python to help teachers with transformation to this language?*

We used design based research with four iterative cycles (Creswell, 2002) as a main research strategy and we used qualitative methods of data collection and data analysis (Lichtmann, 2012), including semistructured interview, observation (transcriptions and field notes), and audio-visual materials (photographs and recorded videos of pupil's work and their products). We conducted our research in ordinary upper secondary school in capital city of Slovakia. In the first and third iteration of the

verification we cooperated with the teacher who taught their pupils within compulsory school subject Informatics according to our curriculum intervention. The pupils from these iterations haven't met with a textual programming language before. The teacher was experienced with teaching programming in Pascal (Lazarus) and C++, but had no experience with the Python programming language. During the verification we observed the teacher's questions about the course and the Python language, as well as pupil's questions, mistakes and misconceptions. The second iteration we performed during our teaching activity at an upper secondary school with the pupils of first grade. This group consisted of pupils who were selected at the admission process of the school on the basis of stricter acceptance criteria (success rate about 1:10). Pupils were also encouraged to work independently, which allowed them to learn even faster and solve more challenging tasks. They had experience with childrens programming environments of Imagine or Scratch. One pupil had experience with Python (but not turtle geometry) and one of them had experience with C#.

In one group there were from 9 to 11 students, aged from 14 to 16. These groups included boys and girls in different ratio, see Table 1.

Iteration	Girls	Boys	Total
1.	4	5	9
2.	4	6	10
3.	9	2	11

Table 1. Pupils participated in our research

First steps of designing the curriculum intervention

Before designing materials, we have gone through several steps of acquiring experience with Python. Since, we also transformed to Python language from Pascal, these steps could be in part similar to those of teachers when they switched to Python. By having experienced it ourselves, it helped us to better understand the situation of the teachers. (Mészárosová, 2016)

Pedagogical activity

A PhD study also includes teaching activities of students at the university. During our research which lasted about a year and a half (4 terms) we conducted exercises for the subject Programming in Python for students of the Applied Informatics unit, each semester in two groups of around 30 students. Thanks to this, we gained not only programming skills in Python, but we saw lectures, led by an experienced university pedagogue right in practice (Blaho, 2016). Also gained valuable observations about what kind of mistakes and bugs that pupils make. Our pedagogical activities continued during further iterations, where we continued in practice teaching programming in Python for Computer Science Teacher students as well as pupils of an upper secondary school.

Overview of teaching programming in Slovakia

Obtaining an overall view of the current state of teaching programming in Python should have helped us to better conceive the possibilities of teaching programming in Python at upper secondary schools. Within, we talked with teachers who shared their experience of teaching programming basics in Python. In this iteration of our research we also implemented a case study aimed at describing of a teachers practice while she switched to teaching in Python. Obtaining a general overview of the current state of teaching programming in Python in Slovakia should help us to imagine and to better understand the possibilities of teaching programming basics in Python language at upper secondary schools in Slovakia as well as teacher's requirements for methodological materials. (Mészárosová, 2015)

Selecting the motivation of turtle geometry

After creating a comprehensive overview of teaching programming basics in Python, we have a variety of motivations to choose from: 1. Turtle geometry; 2. Using graphical canvas (Tkinter); 3. Task with mathematical context without graphical motivation. We have chosen the motivation of turtle geometry

that supports learning by discovering and exploring. Our goals were to create a curriculum intervention for learning programming basics in Python for the pupils in the first year at upper secondary schools. We have based on the assumption that pupils coming from lower secondary schools have experience with different programming environments for children as Imagine or Scratch. Our goal is that they meet a higher level programming language in the upper secondary school, which is suitable for learning the programming basics. At the same time, we have tried to find ideas for our materials so that the pupils get acquainted with more demanding algorithms in this programming language.

Selection of the primary resources

We selected some existing textbooks as guidelines to follow in developing our materials. As the primary resource we chose the textbook Algorithms with Logo (Varga, 1999), which is intended for upper secondary schools and is using turtle geometry. When choosing the didactic procedure and tasks we were motivated by lectures on programming in Python and C# from the Comenius University in Bratislava; also by the textbook for Delphi (Blaho, 2012). While these materials are not intended for our target group or they focus on another programming language, we were motivated by the selection of didactic procedures, methods, approaches and motivations used in these materials. Also we have to follow the National Education Curriculum which defines the knowledge and skills pupils have to acquire before they graduate.

Developing and verifying our curriculum intervention

The process of the materials development contained three iterations of verification in three different groups of pupils at upper secondary schools. In one of the groups we took lessons by ourselves and also cooperated with a computing teacher who taught the lessons according to our curriculum intervention in two different groups. During the verification we followed the teacher's questions about the course and the Python language, as well as pupil's questions, mistakes and misconceptions.

1st iteration

The design of the pilot version partly went on simultaneously with its integration to practice and validation, so we were able to respond more flexibly to the needs of the pupils and teachers, and to the encountered deficiencies and problems as well. When designing our materials, we mostly followed the textbook for the Logo language. However, we encountered a problem with the order of the topics: loops and variables. In the pilot design we decided to devote ourselves to the topic of loops first. We prefer them because of the lesser level of abstraction than variables, and also because we followed the process of learning programming with turtle geometry used in practice. The problem of the variable inside of the looping construct still remains even though we were trying to not use the looping variable in tasks. After the evaluation we decided to change the order of the topics: variables and loops. A similar problem was encountered by researchers Hromkovič et al. (2016), who stated in their publications that they have designed a course of programming basics using Logo tutorials with turtle geometry. As one of the positive attributes of the Logo language they mentioned the repeat loop, which according to the authors, is the simplest loop to learn the looping construct without the looping variables or conditions. They solved the problem of how to position the topic of loops without the topic of the variables, by implementing the repeat loop into their own programming environment for Python.

We found out that the materials need to be in two parts: materials for teachers and worksheets for pupils. The teachers need an extensive description of the lessons and also to solve the tasks before the lessons, so we added the solutions of the tasks to the materials to make this process easier. Materials also include the aims of each lesson according to the National Education Curriculum to help teachers with school plans and documents. In order to simplify the preparation of the teacher before lessons, we added a list of used commands as a brief reminder of commands the pupils already know and the syntax of the language.

We also found out that the teacher who does not have experience with teaching programming in a language that is new for her has difficulty finding errors in pupil's code. Therefore, we have added recommendations to the materials on what kind of mistakes pupils can do, so the teacher after reading the material was ready to find errors in pupil's codes.

2nd iteration

This iteration of the validation gave us valuable experience with the organization of the lessons and provided an insight of the teaching in a group of pupils who have different programming experience and also different cognitive abilities than the first group. In this group there were pupils who progressed faster than most of the group. This is the reason why we adjusted the pupil's worksheets to include examples intended for a common solution with teacher, while more skilled pupils (in our case, pupils who attended other programming courses) did not have to wait for slower pupils. They were also able to work independently and when they encountered a problem and needed to know a new command or knowledge, they asked the teacher. In this group we met with such internal motivation of pupils, where we did not have to give homework, some of the pupils solved the remaining tasks from the worksheets themselves at home.

We were also enriched with an experience of teaching without a projector, using one part of the whiteboard for code and the other part for drawings to support explanation. However, we noticed that pupils had more negative reactions to the error messages when they ran the code, compared to the pupils from the first iteration. In this group, we noticed that pupils asked for help immediately (or faster) after the error report, while in the first group they asked for help just after they couldn't fix the problem. These differences between reactions on error messages could be attributed to the fact that while in the first group pupils saw on the projector how the teacher corrects bugs in the code, in the second group they did not see the teachers' reaction on the error messages because she wrote the code on the whiteboard. In this group, we had to positively support pupils when they encountered error messages in order to avoid being scared and individually explained to them how to interpret the error message and fix the program.

3rd iteration

According to the analysis of frequent mistakes, we have adapted the methodological material so as to avoid similar mistakes and misconceptions of the pupils. We have added explanations, reminders and also changed the order of some tasks, in which we firstly failed at the determination of the grading. During the verification, we had to devote more space to the tasks of using nested loops. Among the common mistakes of pupils was that they executed the program only after they implemented the whole solution of the problem even in the case of difficult tasks. That is why the program often did not draw the desired picture and it was not possible to see directly from the picture in which step the error is (see Fig. 1). (Jašková et al., 2017)

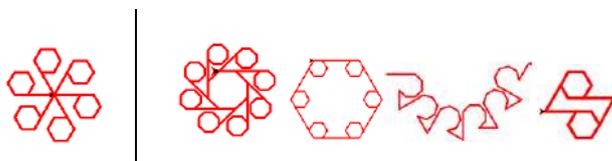


Figure 1. Picture on the left: the task; Pupils most common mistakes on the right.

In case of these errors, pupils needed help from the teacher, but mostly only needed to correct a value of a parameter in a command or the number of repetitions. We tried to focus their attention to solving difficult tasks by dividing them into smaller parts, to gradually solve and verify their solutions. After analysing these observations, we came to the conclusion that it would be more appropriate for the pupils to learn functions and solve these or similar tasks using them, instead of using nesting loops. In this way, the components of informatics thinking – decomposition and pattern search – would be accessible and developed in programming lessons. The solutions themselves would be more comprehensive, understandable and would help pupils to find errors easier. In such tasks, where they already used functions in solutions, we didn't meet such problems.

After analysing the observations from these iterations, we finally enriched the materials with recommendations for teachers to eliminate pupil's errors, for example, specifically, in Python, we need to carefully explain the indentation of the commands in loops, conditions and functions. We present frequent pupil's mistakes in the syntax of the language, such as forgetting to enter command parameters or skipping parentheses for some commands. We also warn them to keep in mind the work of pupils to

be able to orient themselves in an environment with three separate windows, using REPL and programming mode, or to teach pupils how to read the error messages.

Results and Discussion

Before the design of the curriculum intervention we needed to take some steps to get the necessary knowledge and collect experience. Without these steps, we may have overlooked some important elements. We consider these steps to be important before designing a textbook or methodical materials for teaching programming:

1. **Learn the language** as much and deep as possible, to recognize the positive and negative aspects of the language. Without this step we can easily overlook the strengths of the language and make a mistake where we just adapt a book to another programming language without taking advantage of the programming language options that would make coding easier and thus focus on algorithms.
2. **Pedagogical experience** is necessary to design lessons for programming in a new language.
3. **Literature overview** is needed to have insight into what materials exists, to draw inspiration and motivation from the best, which is used in our schools, country and abroad for programming courses. Didactic knowledge is also necessary for designing valuable materials.
4. **Co-work with teachers** and monitor their needs in order to be able to develop a material that is helpful for them. During the verification, teachers can point out such problems and mistakes of which the creator wasn't aware. It's also necessary to verify the materials in groups of pupils with different skill level and teachers with a different amount of experience.

We identified some key elements and features a material should contain to help teachers transform to a new language:

- materials for pupils (worksheets)
- recommendations for teachers what mistakes are being made by pupils, as one of the most challenging elements of teaching in a new language is to find mistakes in the pupil's code
- recommendations for the assessment
- key differences in the new language vs. language used before (in our case Python and Delphi)
- list of used commands helps the teacher especially at the beginning of the course

The output for teachers is the created curriculum intervention which we describe in the following section of the paper.

Curriculum intervention

The materials are divided into six parts by topic (introduction to Python, variables, loops, functions, randomness and conditions) and consists of a methodical material for teachers, supplemented with worksheets for pupils. In each part of the material we defined the required knowledge and skills of pupils. The structure of the lessons (for 45 minutes), pedagogical procedures and the description of its agenda are also part of the material, along with tasks for pupils, their solutions and recommendations for teachers. When designing tasks, we also tried to select programs that do not need a large set of commands such that we don't unnecessarily burden the students. We focused on the concepts and constructions defined in National educational curriculum. In terms of the process of lessons, we follow the scheme: *Motivation* → *Collecting experience* → *Exercising the knowledge*, while in the Motivation section the teacher, along with the pupils, solves an example at the board. In the other parts the pupils work independently on their own computer (that. Tasks had a constructionist approach, pupils got pictures which they had to draw using turtle geometry).

Introduction to Python and turtle geometry

The aim of these lessons is to familiarize pupils with the Python programming language and its development environment IDLE, as well as with the idea of turtle geometry. The first two lessons of the course are aimed at learning to work with IDLE – command and programming mode too. They are also designed to help pupils to learn basic commands for navigating the turtle. We want to achieve pupils be able to control the basic commands automatically in order to make the following lessons, when they will meet new constructions and concepts, easier. Working with the three separate windows they encounter

in this environment can be challenging for some pupils. That is why we have reserved two lessons for this topic so that students have enough time to practice and the teacher has the opportunity to make sure all pupils are able to navigate through this environment. Another aim of these lessons is to make pupils think about the properties of the performer – turtle. Pupils draw simple pictures in these lessons which are intended to automate the basic commands for the movement of the turtle (see Fig. 2).

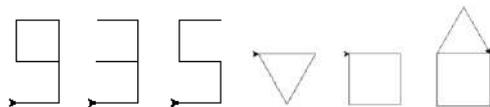


Figure 2. Examples of the tasks for practicing the navigation of the turtle.

Variables

For the topic of variables, we designed only a single lesson, but pupils use them in the following lessons as well. In explaining variables, we used a didactic method widely used in practice, representing the variables by drawing boxes that contain the values stored in these variables. It is important to show the process of assigning a value to a variable step-by-step. Our goal was that pupils understand the meaning of a variable and its use in Python, learn to assign a value to a variable and later use it, and to be able to identify from the task, which data must be remembered, respectively vary and therefore require the use of variables.

For loops

We divided the topic of for loops into five lessons, ranging from loops with enumerated values for the control variable, for loops with the function `range(value)` to loops with the control variable used in the body of the loop. The aim of these lessons is that pupils are able to recognize recurring patterns, and to generalize and write a solution using loops. Similarly, to the topic of variables, we make reckoning of a thorough tracing of the program, with which pupils can search for and find certain patterns. It can also be helpful in generalizing. This topic contained two lessons devoted to nested loops, but after analysing pupil's mistakes, we have decided to leave out nested loops from the course and give prepared tasks to pupils after the lessons of functions.

Functions

The aim of these lessons is that pupils can define and call functions, and also to use functions with simple parameters. New concepts and knowledge gained at these lessons are: definition of the function, function call, function name, parameters, function body, how the parameters work and the use of function parameters inside the function body. We proceed from functions, loops in the function body to functions with parameters. In the tasks for functions with parameters we focus on explaining parameter delivery, creating a local variable for the function parameter, and the disappearance of the variable after the function body has been executed.



Figure 3. Examples of the tasks for practicing functions.

Turtle position and randomness

In this topic, we used the motivational example of drawing a starry sky. During this exercise, we taught the pupils to change the turtle's position using the `setpos(x,y)` command and to generate random values using the `random` module. Including this topic in the curriculum intervention made available the use of motivational tasks with random values.

Conditions

The aim of these lessons is to understand how branching the program behaves, to recognize situations and conditions when branching is needed, and what part of the algorithm has to be performed before, within, and after the condition. After these lessons, pupils should be able to solve tasks that combine loops and branching. Among the tasks, we included known ones as throwing coins or generating a large number of dots on the canvas at random positions and colouring them according to their location.



Figure 4. Example of a pupil's project

At the end of the course, we included a project to retain the acquired knowledge and for the summative assessment by the teacher. We also give recommendations for teachers for the formative and summative assessment over the course.

Discussion

The use of Python and turtle geometry have been discussed in the literature rather extensively. The authors of papers (Hromkovič, 2016; Cho, 2016; Ayock, 2015; Vidal Duarte, 2016) highlighted the strength of learning programming basics using turtle geometry. According to Cho (2016), “*Python that starts with turtle modules is linked to the exploration of microworlds at primary and middle schools, thus serving to lower the entry barrier for students in performing exploration at high schools*”. Researchers also mentioned CS1 courses starting with turtle geometry, continues with game engine (Ayock, 2016) or extensive projects (Vidal Duarte, 2016). In Slovakia, however, turtle geometry is used in earlier education (using Imagine Logo), but it is also appropriate for higher education, upper secondary schools, where pupils are learning programming in textual programming languages. In our research we met with increasing motivation among pupils to learn programming, confirming the suitability of our choice of motivation with turtle geometry.

There are several books and online courses for Python using turtle geometry, but these materials are rather suited for self-study, e.g. (Payne, 2016). In the learning process we prefer the scheme: *Motivation* → *Collecting experience* → *Exercising* the knowledge, where we need a motivational example that the teacher solves together with pupils by discussion. Consequently, the pupils solve the tasks to exercise the new knowledge, where we need a lot of graduated tasks. Our materials also differ from most of the materials for Python by focusing on new algorithms and concepts, but not on language characteristics. Materials are aimed for teachers and the worksheets for pupils include only tasks without a huge amount of text. Our goal was to minimize the amount of commands pupils need to learn and to gain new programming concepts and knowledge, we used just a fraction of the commands compared to other materials (Belan, 2013; Kučera, 2016). This goal is also confirmed by the fact that during the verification, pupils easily remembered the used commands. A similar material exists in Switzerland (Kohn, 2016), where researchers created an environment that offers a repeat loop instead of a for loop, therefore the material follows a different approach in teaching.

We distributed the curriculum intervention to computing teachers in Slovakia. We extended the materials to teacher's awareness thanks to the PyCon SK 2017 conference, where we organized a section for teachers. On this conference, we gave teachers copies of the materials, and also made it available on the internet. Then we asked them to comment the materials. We have received positive feedbacks from teachers, such as “I appreciate the performance of the curriculum intervention in my class very positively, even though the breaks they have completed their solutions without any other motivation, although they did not want to code before.” or “it works! especially in girls I saw increased motivation

for programming". Also the fact, that some teachers chose this material for their teaching is positive confirmation too. It should be noted though, that they choose the motivation according to the characteristics and possibilities of the group of pupils, their style of teaching and habits. This feedback is very valuable for us for further development of the materials.

Conclusion

In our work, we focused on teaching programming basics in Python. Our goal was to develop a curriculum intervention for learning programming basics for upper secondary schools helping teachers to transformation to a Python programming language. In it we targeted basic concepts and constructions as variables, loops, functions and conditions. We developed the methodical material in iterations, realized and verified through the development of three groups of pupils at upper secondary schools with teachers who haven't experienced with teaching in Python before. The material has been distributed to teachers and we will continue to collect feedback from them and monitor their experiences with the use of the material in practice. When we were developing the curriculum intervention, we cared that we did not overburden pupils with too many Python commands, but rather let them learn the basic concepts and constructions, and motivate them to further learn programming. Our materials include recommendations and notices that will make it easier for teachers to transform from the Pascal programming language, they have experience with, to Python. We believe that the materials created by us will serve the development of pupil's algorithmic thinking, important knowledge and skills, and last but not least the fulfilment of the educational goals of computing.

References

- Aycock, J., et al. (2015) A Game Engine in Pure Python for CS1: Design, Experience, and Limits. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15). ACM, New York, NY, USA, p. 93-98.
- Ateeq, M., et al. (2014) C++ or Python? Which One to Begin with: A Learner's Perspective. In Proceedings of the International Conference on Teaching and Learning in Computing and Engineering (LaTiCE '14). IEEE, 64-69. ISBN 978-1-4799-3592-5. DOI: <https://doi.org/10.1109/LaTiCE.2014.20>
- Belan, A. (2013) *Python, eUčebnica pre septimu osemročného alebo 3. ročník štvorročného gymnázia* (1st. ed.). Druska Books, Bratislava, Slovakia. ISBN 978-80-89646-35-7.
- Blaho, A. (2012) *Informatika pre stredné školy: Programovanie v Delphi a Lazaruse* (2nd. ed.). Slovenské pedagogické nakladateľstvo - Mladé letá, s.r.o., Bratislava, Slovakia. ISBN 978-80-10-02308-0.
- Blaho, A. (2016) *Programovanie v Pythone 1 (prednášky k predmetu Programovanie 1)*. Retrieved September 30, 2016 from <http://python.input.sk/>.
- Creswell, J.W. (2002) *Educational Research: Planning, Conducting, and Evaluating Quantitative Research* (4th. ed.). Pearson Education, New Jersey, NJ, USA. ISBN 978-81-203-4373-3.
- Cho, H. H. et al. (2016) Math-based Coding Education in Korean School. In *Proceedings of Constructionism 2016*. Bangkok, Thailand, 167-174. ISBN 978-616-92726-0-1.
- Fuller, U, et al. (2007) Developing a Computer Science-specific Learning Taxonomy. *SIGSE. Bull.* 39, 4, 2007, s. 152-170.
- Goldwasser, M. H. and Letscher, D. (2008) Teaching an object-oriented CS1 — with Python. In Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08). ACM, New York, NY, USA, 42-46. DOI: <http://dx.doi.org/10.1145/1384271.1384285>
- Grandell, L. et al. (2006) Why complicate things?: introducing programming in high school using Python. In Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06), 52. Australian Computer Society, Inc., Darlinghurst, Australia, 71-80. ISBN 1-920682-34-1.

- Hejný, M. and Kuřina, F. (2009) *Dítě, škola a matematika: konstruktivistické přístupy k vyučování* (2nd. ed.) Portál, Praha, Czech Republic. ISBN 978-80-7367-397-0.
- Hromkovič, J., et al. (2016) Combining the Power of Python with the Simplicity of Logo for a Sustainable Computer Science Education. In *Informatics in Schools: Improvement of Informatics Knowledge and Perception* (ISSEP '16). Springer International Publishing, 155-166. ISBN 978-3-319-46746-7. DOI: https://doi.org/10.1007/978-3-319-46747-4_13
- Jašková, L., Mészárosová, E. and Winczer, M. (2017) Skúsenosti z úvodného kurzu programovania v Pythone na gymnáziu. In: *DidInfo and DidactIG 2017-* Banská Bystrica : Univerzita Mateja Bela, 2017. - S. 174-178. ISBN 978-80-557-1216-1
- Krapan, D. and Bilobrk, I. (2011) Introductory Programming Languages in Higher Education. In: *Proceedings of the 34th International Convention* (MIPRO '11). IEEE, 1331-1336. ISBN 978-1-4577-0996-8.
- Kohn, T. (2016) *Python: Eine Einführung in die Computer-Programmierung*. Retrieved from <http://jython.tobiaskohn.ch/PythonScript.pdf>.
- Kučera, P. (2016) *Programujeme v Pythone* (1st. ed.). Bratislava, Slovakia. ISBN 978-80-972320-4-7.
- Linn, M. C. and Dalbey, J. (1988) Cognitive consequences of programming instruction. In *Elliot Soloway and James C. Spohrer* (ed.) *Studying the novice programmer* Lawrence Erlbaum Associates Inc., New Jersey, NJ, USA, 57-82.
- Lichtman, M. (2012) *Qualitative Research in Education: A User's Guide* (3rd. ed.). SAGE Publications, Thousand Oaks, CA, USA. ISBN 978-1-4129-9532-0.
- Mannila, L. and de Raadt, M. (2006) An objective comparison of languages for teaching introductory programming. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006* (Baltic Sea '06). ACM, New York, NY, USA, 32-37. DOI: <http://dx.doi.org/10.1145/1315803.1315811>
- Mészárosová, E. (2015) Is Python an appropriate programming language for teaching programming in secondary schools? [ed.] Pavel Kapoun. 2, Ostrava : University of Ostrava, 20. Máj 2015, *International Journal of Information and Communication Technologies in Education*, Zv. Volume 4, s. 5-14. ISSN 1805-3726.
- Mészárosová, E. (2016) First steps of developing a methodology for teaching programming fundamentals in Python . Ostrava. In: *Proceedings of Information and Communication Technology in Education*, 2016. ISBN 978-80-7464-850-2.
- Payne, B. (2015) *Teach Your Kids to Code: A Parent-Friendly Guide to Python Programming* (1st. ed.). No Starch Press, San Francisco, CA, USA. ISBN 978-1-59327-614-0.
- Salanci, L., et al. (2011) Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika - Didaktika programovania pre SŠ 2. Štátny pedagogický ústav, Bratislava, Slovakia. ISBN 978-80-8118-090-3.
- Varga, M., et al. (1999) *Informatika pre gymnáziá: Algoritmy s Logom* (1st. ed.). Media Trade, spol. s r. o., Bratislava, Slovakia. ISBN 80-0802965-X.
- Vidal Duarte, E. (2016) Teaching the First Programming Course with Python's Turtle Graphic Library. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITICSE '16). ACM, New York, NY, USA, 244-245. DOI: <https://doi.org/10.1145/2899415.2925499>.

The Impact and Effectiveness of Technology Enhanced Mathematics Learning

Einari Kurvinen, *emakur@utu.fi*
University of Turku, Finland

Valentina Dagiene, *valentina.dagiene@mii.vu.lt*
Vilnius University, Lithuania

Mikko-Jussi Laakso, *milaak@utu.fi*
University of Turku, Finland

Abstract

What happens inside the classroom is crucial for pupils' learning. Mathematics is mostly hard work and only small portion is accounted by talent. However emotions and enjoyment can facilitate the hard work and make it easily done. We can use technology to help pupils to complete more exercise, make practicing more interesting and motivate pupils, which in turn will improve learning performance and learning results.

A 15-week research study was conducted by research team of University of Turku, Finland and Vilnius University, Lithuania. Three schools were selected for this study. From each school we had two 3rd grade classes, one with control group and one treatment group. There was altogether 140 pupils (N=140) of which 69 (N=69) formed the treatment group and 71 (N=71) formed the control group. Figure 1 describes the research setup. The test started with a pre-test and was followed by 15 week research phase. The research was concluded with a post-test. During the research period, pupils from treatment group had one mathematics lesson per week transformed into technology enhanced mathematics lesson using a virtual learning platform called ViLLE. The control group followed typical teaching without any changes.

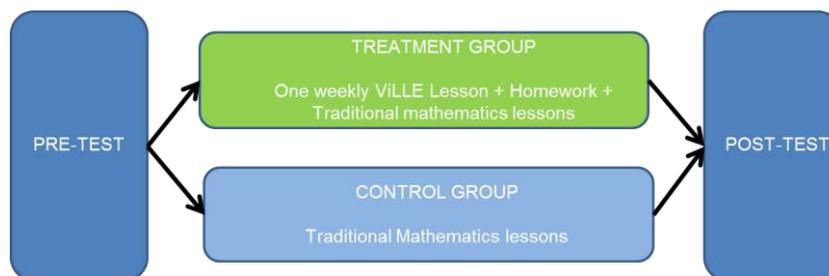


Figure 1. Research setup

Identical tests for pre-test and post-test were used in the study. The test consisted of two parts. The first part measured mathematics performance in topics taught in 3rd grade. The second part was an arithmetic fluency test that measures how fast pupils can solve basic arithmetic facts. The study clearly showed that the treatment group outperformed the control group pupils with statistical significant difference both in mathematics performance and in arithmetic fluency. When we combine these results with our previous findings there is solid evidence that the digital learning path in ViLLE makes a radical improvement to pupils' math learning. What makes it more amazing that we only replaced one lesson in a week.

Keywords

gamification; mathematics; primary education; technology enhanced learning; TEL

Introduction

Using technology to help students learn is complex. Technology enhanced learning (TEL) provides an interesting addition to traditional teaching methodologies (Goodyear & Retalis, 2010). We are going to use the term TEL as umbrella, but other terms, such as computer-aided learning (CAL), or e-learning carry similar connotations. There are many tasks where computers excel against humans. For example, assessing unambiguous tasks, asking repetitive random questions or giving feedback to multiple pupils at the same time. Using TEL enables educators to duplicate parts of their pedagogical knowledge and take the burden of simple time consuming tasks from teachers whose time is valuable and scarce.

Mathematics is at the core subject of science, engineering and technology. The 2015 results of the OECD Programme for International Student Assessment (PISA) show that many countries need to improve mathematics education in schools (Gurria, 2016). Too many students in schools still perceive mathematics as an educational stumbling block. How can educational technologies influence mathematics education and improve pupils' mathematical skills? First, technologies can improve students' technical skills in mathematics. By technical skills, we consider the know-what and the know-how. For example, students recognize formulas and notation (*know-what*) and apply procedures to solve different types of problems (*know-how*). How could be possible to improve the traditional learning outcomes in mathematics that are expected by most policy makers, and, at the same time, develop other important skills for innovation, such as reasoning, understanding, posing problems?

In this paper, we provide evidence that TEL strategies are effective both for traditional and for complex, unfamiliar and non-routine math problems. According to Goodyear and Retalis (2010), TEL activities that promote active learning are considered constructivist. When pupils are active, they are involved in the learning process, more motivated and interactive with the material provided. These aspects make learning deeper instead of superficial strategies.

We report the experiences in Lithuania utilizing a digital learning path in mathematics in Finnish learning platform called ViLLE. The aim of this research was to find out how regular, weekly technologically enhanced ViLLE-lessons affect the learning performance and arithmetic fluency of Lithuanian 3rd graders compared to pupils receiving traditional instructions. ViLLE provides ready-made weekly lessons with interactive and gamified exercises supporting the Finnish math curriculum that can be easily adapted into any curriculum. We have conducted multiple studies in Finland (e.g. Kurvinen et al., 2015a, 2015b) with success.

This paper is organized as follows: First we introduce previous research in the field of TEL and previous findings using ViLLE in Finland. Methodology, tests and content used by the pupils are introduced in research setup, which is followed by results. Finally, this paper is finished with conclusions and discussion.

Previous Work

The research community has studied TEL for years. Most researchers seem to agree, that using Information and Communication Technologies (ICT) in teaching and learning can have a positive effect on learning. Still, there is room for doubt, and some researchers demand for more robust methodology and evidence for benefits of TEL (e.g. Drijvers, 2016).

There are multiple meta-analyses conducted on TEL in education, all showing a positive impact on learning, motivation and confidence of the pupils (e.g. Edwy & Vodanovich, 2017; Vogel et al., 2006; Li & Ma, 2010). Li and Ma (2010) found out that there is a statistically significant positive effect of using educational technology, especially in primary education. However, the researchers remind us that introducing technology does not automatically yield for better results, but there must be a well-planned strategy on how to utilize them in teaching and learning. For example, in drilling-based games it is typically easy to just try different answers without thinking. Pupils relying on this superficial "learning strategy", will probably not learn much. Li and Ma divided technology enhanced solutions into two categories. First category is traditional approach, which means teacher-centered whole-class instruction. The second category is constructivist approach, which is student-centered. This approach puts more emphasis on active learning of the individual student through problem-based and discovery-

based learning. They found out that both approaches lead to positive effect, but the constructivist approach leads to significantly and consistently higher impact on learning.

Repetition and immediate feedback are two strong factors which are easily implemented in TEL and can clearly provide great improvement in learning. Brosvick, Dihoff, Epstein and Cook (2006) studied automated feedback and its effects on learning. They used paper-based automatic feedback and compared that to immediate feedback given by an educator. Both were found to be effective and there was not significant difference between the two methods. Based on this, we can draw a conclusion that the feedback does not have to come from a human being to be effective, and the same effect can be provided automatically by software. Humans are of course able to give more in-depth feedback in various topics but on the other hand computers are tireless and will offer feedback for as long as you require. The researchers concluded that immediate feedback increases interest and involvement in assessment, helps drilling to be more effective and promotes active learning by maximizing time on task (Brosvic et al., 2006). In addition to this, Attard and Curry (2012) found that immediate feedback helps building pupils' self-confidence, thus making pupils feel safer to make mistakes and try again. This helps building pupils' persistence.

Pilli and Aksu (2013) conducted a three month study on fourth graders in North Cyprus. They assigned 26 pupils into a control group and 29 into a treatment group. Their research setups are similar to the research setup presented in this paper. They also compared TEL to traditional teaching and verified their results with identical pre- and post-tests. In final results, the pupils using technology outperformed the control group in multiplication and division of natural numbers. In fractions the difference was not statistically significant. They concluded that computer-assisted instruction, or TEL in this context, is a good supplementary method compared to traditional mathematics instruction.

In previous studies conducted in Finland, we have found similar positive effects on learning by using TEL. We have conducted several studies in primary schools of using ViLLE for one weekly lesson for mathematics teaching. We have followed a similar research setup in all cases. The test starts with a pre-test, which is followed by the research period, where one technology enhanced mathematics lesson in each week (accompanied with homework) is utilized. This technology enhanced lesson is a traditional mathematics lesson transformed into an electronic lesson, thus the treatment group is not provided any extra instruction. After the research period, a post-test is conducted and the results are determined by comparing the treatment group and control group between the two tests.

One of our earliest studies is on first graders. Two classes participated in a 10 week research period. The improvement of the treatment group in post-test was statistically significant (Kurvinen et al., 2014). Another 18-week long study was then conducted on four first grade classes from two different schools. Again, the treatment group achieved statistically significantly better results in the post-test, while no significant difference in the pre-test was seen (Kurvinen et al., 2015b). During the same time, we also conducted a similar study on third graders. Two classes participated in the 18 week long study. Again, without a statistically significant difference in the pre-test, the treatment group achieved statistically significantly better results in the post-test (Kurvinen et al. 2015a).

Virtual learning environments for learning mathematics

There are many virtual learning environments for learning and some are specialized in mathematics. Many publishers have their own learning environments to supplement exercise books. One of the most famous non-profit platforms is Khan Academy (<https://www.khanacademy.org/>). Another famous system is Mathletics from Australia (Stephan, 2017; Nansen et al., 2012). Mathletics did not provide statistical significant improvement on standardized tests, but it was said to improve pupils' motivation and make differentiation easier for teachers (Stephan, 2017). Brasiel et al. (2016) compared 11 platforms to supplement mathematics teaching in K12 education in Utah. They found out, that all the platforms, except for one, had a positive impact on learning. However, only two platforms (ALEKS and i-Ready) had a statistically significant impact, but only when pupils were using the platforms on recommended level.

The problem of using virtual learning environments is more about choosing a proper one than just finding existing solutions. Many solutions provide only the platform without any content. Others provide some

content but the content might not be customizable or compatible with local curriculum. Because there are so many solutions already available, the choice is usually made by seemingly small features that in teachers' everyday life make huge impact. For example, compatibility to national curriculum, localization or the scope of the content are crucial for teachers.

ViLLE is an exercise-based digital learning platform, developed at the University of Turku, with over 100 different exercise types. Most of the exercise types are automatically assessed and give immediate feedback to pupils. The progress made by pupils is stored by ViLLE and comprehensive statistics and learning analytics based on pupils' answers are presented to the teacher. ViLLE provides all necessary tools for teachers to create their own content in any subject they choose in ViLLE (Laakso et al., 2018b.)

We have created a gamified digital learning path for mathematics for grades 1-9 (ages 7-15). The path has one digital mathematics lesson for each school week and it covers all the topics in the national curriculum in Finland (POPS, 2014). Altogether, the path contains more than 15 000 exercises. Each lesson contains 25-30 exercises that are customizable by the teacher. This path can be easily customized and tailored to meet needs of different national curriculums. Exercises in the digital learning path have been created in close co-operation with teachers, to ensure the suitability and quality of the exercises.

The path is gamified ie. the pupils collect points from the exercises and we give trophies for achieving certain amount of points. The general guideline is that the pupils' goal is to achieve at least 50% of the points available in each lesson. If the goal is not met during the lesson at school, pupils continue the work at home. We provide four different trophies as markers for goals for pupils. By default, the limits are set as follows: bronze is the first trophy (50%), then silver (75%), gold (90%) and diamond (100%). Limits can be customized by the teacher either for the whole class or for individual pupils. Customizing limits per pupil offers an easy way to differentiate pupils. We also provide two exercise categories, called warm-up exercises and bonus exercises to further differentiate pupils. If a pupil is added to either of previously mentioned groups, they will see also differentiated tasks in addition to the normal lesson content. In some exercises pupils have the opportunity to choose from three difficulty levels (easy, normal, hard). Selected difficulty level does not affect scoring but the choice is recorded for the teacher.

Typically each exercise contains 10-15 problems or tasks. Most of the exercises have adaptive capabilities. If a wrong answer is given, the same question is asked again from the pupil. Besides remembering wrong answers, all answers are recorded and automatically analyzed to identify typical misconceptions in mathematics.

ViLLE has build in mechanism for group work. One or more pupils can work together on one device collecting points for their own account. After learning session, pupils can continue their work individually. We have very good results working in pairs from university level programming courses (e. g. Kaila et al., 2016). Anecdotal evidence from observation in class rooms show that working in pairs is also beneficial for primary education. Pupils are able teach one another and discuss about the problems.

Exercises used in the mathematics learning path can be divided roughly into two categories: (1) game-based exercises and (2) more traditional exercises. Figure 2 shows an example of both of these exercises. The first one is gamified multiple choice question, which has a time limit. Pupils choose the correct lane to answer the question shown in the blue container. The question in the figure translates into "Which number is greater than six", and the pupil should select the rightmost option "8". On the right side of Figure 2 is a traditional columnar addition. The exercise has 6 steps, each step containing a new problem to be solved.

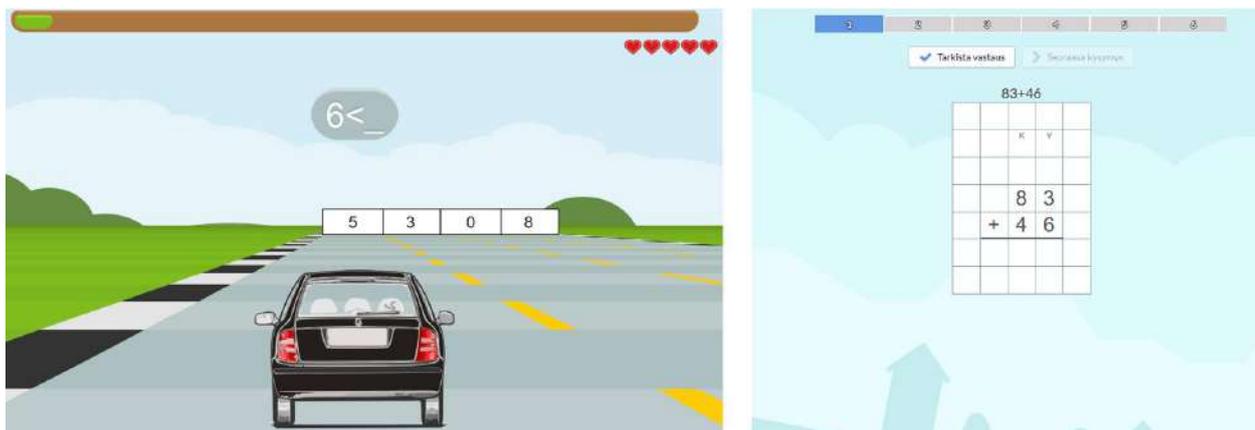


Figure 2. Examples of two different kind of exercise types used in mathematics learning path

At the moment of writing this paper, ViLLE has over 120 000 users, mostly in Finland. The number of users has more than doubled during the last year (Laakso et al., 2018a). The number of users is growing at huge pace, mostly by mouth of words marketing among teachers in Finland.

Research Setup

The research was conducted as a joint project between University of Turku, Finland and Vilnius University, Lithuania with support of UPC (Lithuanian Education Development Centre). UPC had a project with 10 schools around Lithuania. Three schools were selected to take part in this research. Schools were selected based on two criteria: the school should have enough IT-resources to participate and at least two classes on third grade.

Two classes from each of the three schools participated in the research - one class being randomly assigned as the control group and the other the treatment group. Altogether we had three control classes and three treatment classes. Independent variable is the use of ViLLE in one weekly mathematics lesson. Study was started in November during week 45 (2017) by conducting the pre-test with all classes. Treatment groups also had their first ViLLE-lessons during that week. The post-test was conducted during week 12 (2018) in March. Note that the time period includes Christmas and winter holidays, so we used 15 weekly ViLLE-lessons in total in treatment group. All pre-tests were conducted on the same week, but different schools had tests on different days. Post-test were conducted in the same manner. In all schools both groups conducted the test at the same time. A single 45 minutes lesson was used for pre-test and later for post-test.

During the 15-week time period teachers in treatment groups replaced one mathematics lesson a week with a ViLLE-lesson, meaning there was no extra time allocated for teaching in treatment groups compared to control groups. Pupils in treatment group also got homework in ViLLE. In total, pupils completed 15 lessons in ViLLE. All lessons were translated from Finnish to Lithuanian. Participating teachers, together with UPC personnel, selected suitable contents for Lithuanian third graders based on the Finnish 3rd grade mathematics content. Lithuanian pupils worked on the same exercises as Finnish pupils, except for the translation and localization of content.

Participants

All together 140 third grade pupils participated in the research. Treatment classes had altogether 69 pupils (N=69) and control classes 71 pupils (N=71). Number of pupils in each class is shown in Table 2. Before starting the research, research permission was asked from guardians. One pupil from control group did not get permission and the results were hence excluded from the study.

Table 2. Number of students in each class

School	N (Treatment group)	N (Control group)
School 1	20	24
School 2	25	26
School 3	24	21

Content in ViLLE

We selected the content used with pupils together with teachers. The content used by Finnish third graders was used as a template. Teachers were requested to choose and arrange 15 out of 48 lessons so that they would match topics they were going to teach pupils during the research period. Each lesson contains approximately 25-30 exercises and each exercise contains approximately 10-15 problems.

Table 3.2 Lessons selected by teachers

#	Lesson	Number of exercises	Number of exercise visible
1	Numbers 0-1000: Revision of addition and subtraction	44	30
2	Revision of multiplication tables 2, 3, 4, 5 and 10	44	26
3	Numbers 0-1000: Columnar addition and subtraction 1	44	25
4	Multiplication: multipliers 10 and 100	44	26
5	Multiplication tables 6 and 8	42	25
6	Multiplication tables 7 and 9	43	27
7	Multiplication: Order of operations	47	26
8	Clock: Application	42	25
9	Columnar multiplication: No carrying, carrying once	45	26
10	Division: Multiplication and division	47	27
11	Division: Order of operations	41	26
12	Measurement: Counting with units	43	26
13	Measurement: Unit conversion	43	27
14	Geometry: Perimeter, area	44	26
15	Numbers 0 - 10 000: Columnar addition and subtraction	44	26

The research was designed to measure the actual impact of ViLLE on learning performance and fluency. For the scope of this research, teachers did not create any new material, but used exercises designed by Finnish teachers and researchers.

. Participating teachers had all the tools teachers would normally have, including hiding exercises or revealing more, if needed. Teachers also had the possibility to assign easier tasks for some pupils, if they thought it was necessary.

Pre-tests and post-test

Effects of using weekly ViLLE-lessons were evaluated by using a two-part test. We used identical tests in pre-test and post-test to ensure comparability between them. All tests were conducted traditionally on pen and paper, thus the treatment group was not favored in tests.

First part of the test was an arithmetic fluency test. The test, called "a 3-minute test", contains 160 basic arithmetic facts (addition, subtraction and multiplication with operands from 0-10). Pupils had 3 minutes to solve as many calculations as they could. Papers were given to pupils upside down. When the time started, they turned the papers and started solving calculations. After three minutes, they wrote in their names as well. For the results, we counted the number of correct answers and wrong answers. Each correct answer was worth one point.

Second part of the tests was a mathematics performance test. The test used in this research is based on a standardized test that we have been developing. The test has been previously used with Finnish 3rd graders (Kurvinen et al., 2015a). The test measures topics mentioned in Finnish mathematics curriculum for third graders (Table 4.33). Some topics are more advanced than basic requirements for third graders, for example decimal numbers. Before conducting the pre-test, the math performance test was tested on seven pupils. After testing we also collected feedback on how well the test measures topics taught in Lithuania for 3rd grades. In the pre-test we found that the test should be difficult enough to use in both pre-test and post-test. Pupils completed the mathematics performance test right after the arithmetic fluency test and they had time until the end of the 45-minute lesson. Only two pupils from the treatment group would have needed more than 45 minutes to finish the test in time.

The mathematics performance test contains nine exercises. Table 4.34 describes each of the exercises and gives one example.

Table 4.3 Exercises used in pre-test and post-test.

#	Type	Example
1	Arithmetic calculations (addition, subtraction, multiplication, division). Order of operations	$(5+2)*7$; $120*3$
2	Calculations with decimal numbers and order of operations	$0.8+1.1$; $500/2+3*12$
3	Addition and subtraction in columns	$597+484$
4	Multiplication in columns	$142*9$
5	Missing numbers (find the value of symbol)	$@+@ = 6$ $@+\# = 8$ $\#+\& = 12$
6	Continue number sequence	1012, 1009, 1006, ____, ____, ____
7	Convert units (length, time, volume)	$2\text{cm} = __\text{mm}$
8	Circle the greatest (integers, decimals, fractions, currency)	1€ 9c 1,5c
9	Rounding	17 to nearest tens

Exercises 3 and 4 were open problems in the Finnish version of the test. To make assessment easier, the exercise was converted into a fill-in-the-missing-numbers exercise. Exercise 5 is a new exercise. All problems are not part of 3rd grade curriculum in Finland or in Lithuania, but the test was designed to be a difficult one to measure how much even the best pupils can improve their results.

Each separate task in the test was assessed as correct or incorrect. Each exercise was then assessed with regards to the percentage of the tasks the pupils answered correctly. For example, the first exercise has 10 calculations. If pupils got 7 correct and 3 incorrect, the total score would be 0.7 (as in 70%). The maximum score from the whole test was therefore 9 points. The internal validity of the test was measured with Cronbach's alpha ($\alpha=0.88$).

Results

Identical tests for pre- and post-tests were used. This enables us to compare groups using absolute means and the differences between each pupil's pre-test and post-test performance. The results section is divided into two sections. The first section presents the results for the second part of the test, mathematics performance test (skills), and the second section presents students' results in arithmetic fluency, which was the first part of the test.

Mathematics performance

Each exercise in the test was scored between 0 and 1 based on the correctness. The test has nine exercises, meaning the maximum score is 9 points. Table 5 lists the pre-test results from each school separately and all control groups and all treatment groups combined.

Table 5. Pre-test results. C=control group, T=treatment group

Group	Mean	p	Median	Min	Max	Std. dev.
School1 C	3.13		2.99	0.94	6.35	1.19
School1 T	2.50	0.06	2.77	1.01	4.14	0.88
School2 C	3.49		3.45	0.82	5.73	1.34
School2 T	3.66	0.65	3.41	0.63	6.57	1.34
School3 C	2.72		2.89	1.23	4.00	0.90
School3 T	3.17	0.15	3.07	0.56	6.57	1.27
Control All	3.14		3.10	0.82	6.35	1.20
Treatment All	3.17	0.88	3.07	0.56	6.57	1.27

An independent samples t-test was conducted to compare the mathematics performance in the pre-test between two groups, in each school and in total. There are no statistically significant differences between any of the classes. There is a near-significant difference in School 1. From minimum and maximum scores we can see that both control group and treatment group have high-achieving and low-achieving pupils. Best-achieving pupils got over two thirds of total score in pre-test. There was no statistically significant difference in scores between control group and treatment group ($p=0.88$, $p>0.05$). The results show that the pupils are on average equally skilled and there is no difference between treatment and control groups. Figure 3 shows this result in a more graphical way, showing the average score achieved by each group.

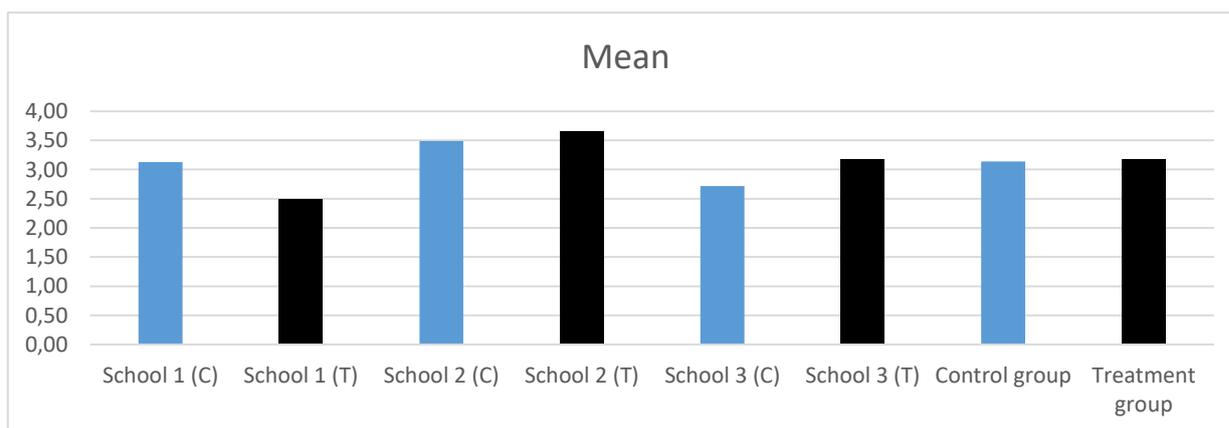


Figure 3. Average score by groups in pre-test

Figure shows clearly that School 1 and School 3 have the biggest differences between control and treatment groups. In School 1 the control group achieved higher scores on average and in School 3 the treatment group achieved higher scores. All control groups and treatment groups combined, the difference is very small.

The post-test was conducted 17 weeks after the pre-test (15 lessons). The post-test was identical compared to the pre-test. Table 6 combines results from the post-test. In addition to absolute test results, the improvement of individual pupils between the tests is included. The improvement is calculated by subtracting pre-test result from post-test result. Difference was only calculated, if the pupil

had completed both tests. In control group there were 63 pupils (N=63) with both tests and 60 pupils in treatment group (N=60).

Table 6. Post-test results. (*=statistical significance)

Group	Mean	p	Median	Min	Max	st. dev.	Mean (difference)	p (difference)
School1 C	4.49		4.54	1.63	6.76	1.34	1.30	
School1 T	4.00	0.35	4.18	1.30	6.24	1.58	1.61	0.38
School2 C	4.37		4.73	1.98	7.37	1.44	0.89	
School2 T	5.25	0.051	4.85	2.58	8.61	1.61	1.55	0.032*
School3 C	3.44		3.62	1.12	5.23	1.23	0.81	
School3 T	4.73	0.0097*	4.87	1.41	7.21	1.61	1.86	0.0019*
C	4.17		4.31	1.12	7.37	1.44	1.00	
T	4.72	0.049*	4.85	1.30	8.61	1.66	1.66	0.0006*

When the absolute post-test results from individual schools are combined, the only statistically significant difference is between the treatment and control groups in School 3 ($p=0.0097$, $p<0.05$). The difference in School 2 is statistically near-significant. When all groups are combined, there is a statistically significant difference between control group and treatment group ($p=0.049$, $p<0.05$). It is worth noting, that the treatment group from School 1 decreased the difference between the two groups, by decreasing the statistical significance from nearly significant ($p=0.06$) to not significant ($p=0.35$). These results show that the treatment group achieved higher scores from post-test with statistical significant difference. Because the pre- and post-test is the same test, more proper measure to use is to compare the difference in score (post-test score – pre-test score).

The effectiveness of ViLLE is solid when we compare the improvement of pupils between tests. Every treatment group has greater improvement than the control group. The difference is also statistically significant in schools 2 and 3. All groups combined, the improvement is statistically very significant ($p<0.001$), meaning pupils using ViLLE have greater learning in results than pupils not using ViLLE. Figure visualizes the difference between all groups.

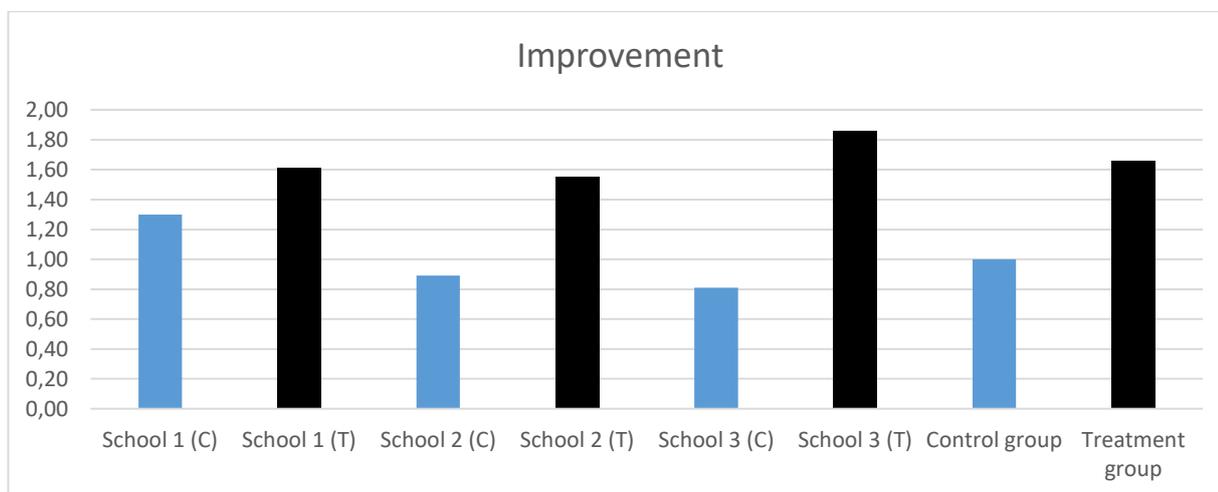


Figure 4. Improvement between tests

Figure shows clearly the greater improvement in all treatment groups. On absolute scale the overall improvement of the pupils between the groups was 11.7%. However, the improvement in the treatment group was 39.8% higher than the improvement in the control group.

Arithmetic fluency

In addition to skills in mathematics, we also tested the arithmetic fluency of the pupils. We used a 3-minute test, which has 160 basic arithmetic facts. Pupils solved as many calculations as they could in three minutes. Only correct answers were counted, but errors were also tracked. Table 7 presents the results from the pre-test.

Table 7. Pre-test, arithmetic fluency (*=statistical significance)

Group	Mean	Median	St. Dev	p	Errors on average
School1 C	60.91	59.50	12.58		0.77
School1 T	53.35	55.00	16.62	0.1	2.90
School2 C	62.00	60.00	15.77		0.80
School2 T	60.96	60.00	14.01	0.8	1.00
School3 C	49.33	51.00	10.58		0.95
School3 T	59.30	59.00	13.61	0.01*	2.22
C	57.74	57.50	14.29		0.84
T	58.16	59.00	14.82	0.86	1.97

There was a statistically significant difference between treatment and control group only in School 3 ($p=0.01$). With all treatment groups and control groups combined, there is no statistically significant difference between the two groups ($p=0.86$). On average, the pupils are on same skill level in both groups. Every treatment group made more errors and pupils from treatment groups made on average twice as many errors in calculations than pupils from control group.

In the post-test we compared the absolute results and improvement between the two tests. Post-test was identical to the pre-test. Table 8 describes the post-test results from arithmetic fluency test.

Table 8. Post-test, arithmetic fluency (*=statistical significance)

Group	Mean	Median	St. Dev.	p	Errors on average	Mean (difference)	p (difference)
School1 C	69.48	67.00	18.48		1.74	8.14	
School1 T	69.83	69.50	15.63	0.95	1.33	17.50	0.014*
School2 C	75.12	76.00	19.67		1.29	12.54	
School2 T	78.92	78.00	19.57	0.5	0.50	17.96	0.056
School3 C	61.11	62.00	13.95		0.78	13.17	
School3 T	83.37	87.00	16.06	0.00007*	0.42	27.11	0.00037*
C	69.35	67.00	18.60		1.31	11.25	
T	77.62	77.00	17.96	0.01*	0.72	20.57	0.000005*

Differences between groups were tested using independent samples t-test. In School 1, there is no statistical significance ($p=0.95$). In pre-test there was a statistically significant difference ($p=0.01$), which means that there is no more difference between groups in School 1. Using ViLLE had enabled the pupils to catch up to their peers in arithmetic fluency. In School 2, there is no statistically significant difference ($p=0.5$). However, in School 3 the difference between control group and treatment group is statistically very significant ($p<0.01$). Difference between all control groups and all treatment groups is statistically significant ($p=0.01$), which clearly shows that using ViLLE improves arithmetic fluency.

Like in mathematics performance, also in arithmetic fluency, the difference between using ViLLE and not using ViLLE is solid, when we compare the improvement of pupils between the two tests. The improvement was higher in treatment groups in all schools. Only improvement which was not statistically significant, was in School 2 ($p=0.056$). Even this result is statistically near-significant. When the groups are put together, the difference between control group and treatment group is statistically very significant ($p<0.001$). This result shows solid evidence that pupils working with ViLLE clearly improved their arithmetic fluency.

In every school, pupils in the treatment group made less errors on average than pupils in the control group. The situation is the opposite when compared to pre-test. Pupils in treatment group solved on average 20.6 calculations more in post-test than in pre-test. In control group the corresponding improvement was 11.3 calculations, making the treatment group's improvement 45.3% higher. The treatment group solved on average 8.3 calculations more in the post-test than the control group. This is 10.7% improvement on absolute scale.

Conclusion and Discussion

This study provides solid evidence that regular usage of carefully planned and designed TEL improves learning performance. Statistically significant improvement is observed both in mathematical skills and arithmetical fluency.

During the 15 weeks of the research, teachers were instructed to use ViLLE to replace one traditional mathematics lesson a week and give homework for pupils. Before the research project, we had agreed on the contents together with all the teachers participating in this research. Otherwise teachers of the treatment groups had the freedom to customize the exercise set as they saw fit. Customizing in this context means hiding some exercises or revealing others. Teachers were also able to assign easier tasks for certain pupils. Both of these features are standard tools in ViLLE to enable teachers to match the content to their pupils needs. In other words, it gives teachers the pedagogical freedom to fine tune the learning experience for their pupils. Due to the scope of this paper we did not track the customizations made by teachers or their effect, only the overall progress made by pupils.

Teachers worked independently with ViLLE during the research. They had the possibility to ask technical or pedagogical questions from the research team. The same possibility is available for all ViLLE teachers. During the 15 weeks, one researcher visited the treatment groups in total four times. First to conduct the pre-test, two times in the middle of the treatment period and last during the post-test. The idea of the visits was to ensure there are no technical issues and to observe the local school culture and how well the Finnish model fits in the Lithuanian schools.

The results presented in this paper clearly indicate that regular usage of ViLLE can improve learning performance, arithmetic fluency and even improve pupils' arithmetic fact accuracy. In the pre-test all the pupils were on average equal in skills. After the 15 weeks, the pupils in the treatment group were on average statistically significantly better than pupils in the control group. This result is in line with previous results and further strengthens the assumption that carefully designed content combined with TEL can lead to better learning results. We consider the pedagogical approach of ViLLE to be constructivist and student-centered, in context of how Li and Ma divided the usage of educational technology (Li & Ma, 2010). One of the key ideas we promote is active learning. The better learning results can be explained by increased amount of practice the pupils got, but the quality of practice is also important. As previously stated, immediate feedback can be a very encouraging and engaging element in TEL (Brosvic et al., 2006; Attard & Curry, 2010). ViLLE gives pupils immediate feedback in all automatically assessed exercise types, thus shortening the feedback loop for pupils. In traditional exercise book practicing, it might take days for pupils to receive feedback on their answers. When receiving feedback takes a long time, it increases the risk of learning wrong strategies and facts.

The verbal feedback from teachers further strengthens these results. With less need for preparation they got more time to encounter pupils individually and they got more data on the pupils' learning. According to the feedback received from teachers, the pupils were motivated and eager to work with ViLLE. According to our previous anecdotal experiences, this also shows that the teachers had proper

methods for motivating pupils and boosted their confidence and they knew how to use ViLLE in a pedagogically sound way.

Most pupils had the opportunity to complete ViLLE exercises from home. However, every school had arranged the possibility to work on the exercises also in school, either during breaks or after school hours. This was not considered to be a problem.

We selected applicable content together with the teachers for the 15 lessons covered in this study. The mathematics performance test used in pre-test and post-test is based on general math skills and it has been used in various studies. To ensure the suitability of the test, it was tested on seven pupils before the actual pre-test. The test is designed to measure general mathematics skills. The arithmetic fluency test is based on general mathematics tasks. The study clearly showed that regular usage of ViLLE will also improve fluency in basic arithmetic tasks.

To conclude our findings, the results shows that with regular usage of ViLLE, we motivate pupils to practice more and improves their learning performance statistically significantly. For teachers this does not require extra work and we can actually save teachers' time from assessment and lesson preparations.

The new Finnish national curriculum in mathematics includes programming in mathematics. At the moment ViLLE exercises integrates exercises for computational thinking and programming in mathematics. Further integration of even more computational thinking contents and research in that field could be fruitful.

Acknowledgments

The authors gratefully acknowledge the support of the Nordic Research Council through the NordPLUS programme of transverse actions. In particular the funding of the two year project "Culturally Diverse Approaches to Learning Mathematics and Computational Thinking", the project code NPHZ-2018/10063.

References

- Attard, C., & Curry, C. (2012). Exploring the Use of iPads to Engage Young Students with Mathematics. *Mathematics Education: Expanding Horizons: Proceedings of the 35th Annual Conference of The Mathematics Education Research Group of Australasia*; Dindyal, J., Cheng, L.P., Ng, S.F., Eds.; Mathematics Education Research Group of Australasia: Singapore, 2012, 75-82.
- Brasiel, S., Jeong, S., Ames, C., Lawanto, K., Yuan, M., & Martin, T. (2016). Effects of Educational Technology on Mathematics Achievement for K-12 Students in Utah. *Journal of Online Learning Research*, 2(3), 205-226.
- Brosvic, G. M., Dihoff, R. E., Epstein, M. L., & Cook, M. L. (2006). Feedback facilitates the acquisition and retention of numerical fact series by elementary school students with mathematics learning disabilities. *The Psychological Record*, 56(1), 35-54.
- Drijvers, P. H. M. (2016). Evidence for benefit? Reviewing empirical research on the use of digital tools in mathematics education. In: Ball L., Drijvers P., Ladel S., Siller HS., Tabach M., Vale C. (eds) *Uses of Technology in Primary and Secondary Mathematics Education*. ICME-13 Monographs. Springer, Cham, 151-175.
- Edwy, R., & Vodanovich, S. (2017, April). The use of 21st century technology in New Zealand primary schools: A systematic literature review. In *Computer Supported Cooperative Work in Design (CSCWD)*, 2017 IEEE 21st International Conference on, 109-114 (IEEE).
- Goodyear, P., & Retalis, S. (2010). *Technology-enhanced learning*. Rotterdam: Sense Publishers.
- Gurria, A. (2016). Pisa 2015 results in focus. *PISA in Focus*, (67), 1. Read 1.5.2018 from <https://www.oecd.org/pisa/pisa-2015-results-in-focus.pdf>

Kaila, E., Kurvinen, E., Lokkila, E., & Laakso, M. J. (2016). Redesigning an object-oriented programming course. *ACM Transactions on Computing Education (TOCE)*, 16(4), 18.

Kurvinen, E., Lokkila, E., Lindén, R., Kaila, E., Laakso, M., & Salakoski, T. (2015a). Automatic Assessment and Immediate Feedback in Third Grade Mathematics. In Proceedings of Ireland International Conference on Education ISBN 978-1-908320-67-4, 89-95.

Kurvinen, E., Lindén, R., Lokkila, E., & Laakso, M-J. (2015b). Computer-Assisted Learning: Using Automatic Assessment and Immediate Feedback in First Grade Mathematics. EDULEARN15 - 7th International Conference on Education and New Learning Technologies, 2303-2312.

Kurvinen, E., Lindén, R., Rajala, T., Kaila, E., Laakso, M. J., & Salakoski, T. (2014, November). Automatic assessment and immediate feedback in first grade mathematics. In Proceedings of the 14th Koli Calling International Conference on Computing Education Research, 15-23 (ACM).

Laakso, M-J., Kurvinen, E., Enges-Pyykönen, P., & Kaila, E. (2018a, May). Designing and Creating a Framework for Learning Analytics in Finland. Accepted to Proceedings of 41st International Convention MIPRO 2018, 771-776.

Laakso, M-J., Kaila, E., & Rajala, T. (2018b) ViLLE – collaborative education tool: Designing and utilizing an exercise based learning environment. *Educ Inf Technol*. <https://doi.org/10.1007/s10639-017-9659-1>

Li, Q., & Ma, X. (2010). A meta-analysis of the effects of computer technology on school students' mathematics learning. *Educational Psychology Review*, 22(3), 215-243.

POPS (2014). Opetushallitus, Perusopetuksen opetussuunnitelman perusteet (The national curriculum of Finland). Accessible in various formats:

http://www.oph.fi/saadokset_ja_ohjeet/opetussuunnitelmien_ja_tutkintojen_perusteet/perusopetus

Nansen, B., Chakraborty, K., Gibbs, L., Vetere, F., & MacDougall, C. (2012). 'You do the math': Mathletics and the play of online learning. *New Media & Society*, 14(7), 1216-1235.

Pilli, O., & Aksu, M. (2013). The effects of computer-assisted instruction on the achievement, attitudes and retention of fourth grade mathematics students in North Cyprus. *Computers & Education*, 62, 62-71.

Stephan, K. P. (2017). Does Mathletics, A Supplementary Digital Math Tool, Improve Student Learning and Teaching Methods at Three Private Catholic Schools in Florida?-A Mixed Methods Study (Doctoral dissertation, Creighton University).

Vogel, J. J., Vogel, D. S., Cannon-Bowers, J., Bowers, C. A., Muse, K., & Wright, M. (2006). Computer gaming and interactive simulations for learning: A meta-analysis. *Journal of Educational Computing Research*, 34(3), 229-243.

Constructionist Approaches to Computational Thinking: A Case of Game Modding with ChoiCo

Marianthi Grizioti , *mgriziot@ppp.uoa.gr*

National and Kapodistrian University of Athens, Educational Technology Lab, School of Philosophy, Faculty of Pedagogy, Greece

Chronis Kynigos , *kynigos@ppp.uoa.gr*

National and Kapodistrian University of Athens, Educational Technology Lab, School of Philosophy, Faculty of Pedagogy, Greece

Abstract

Computational Thinking (or CT) involves a wide range of mental processes like problem solving, recursive thinking, abstract thinking etc, which are considered necessary supplies for the 21st century children. However, despite the wide attention that CT has received, there is still limited research on pedagogical designs and strategies that can promote the acquisition and development of such skills. In this paper, we discuss how constructionist approaches can inform and benefit the cultivation of computational thinking by exploring the case of modifying and sharing digital games. In this context game modding is implemented as a constructionist activity that enables students' progressive engagement in computational thinking through their interaction with various affordances such as coding, data processors, graphics editors, etc. To further investigate this approach, we present the results of a design-based research in which junior-high school students modified games with the digital tool ChoiCo. ChoiCo (Choices with Consequences) is an online platform that integrates three different affordances for designing and modding digital games: a map-based (GIS) game scene, a simplified database and block-based programming editors. The research focused on a) the computational thinking skills that emerge and the meanings that are generated through students' engagement with the three affordances and b)

Keywords

Computational thinking; game modding; block-based programming; progressive engagement

Game Modding for Computational Thinking

Computational Thinking (or CT) was firstly described by Wing as a new way of thinking that derives from fundamental concepts of computer science but it is applicable to a wide spectrum of scientific fields and to everyday activities. It includes a set of thinking skills, habits and behaviours that are integral to solving complex computational problems, such as conditional logic, debugging and error detection, algorithmic thinking, process of information etc. However, despite the big attention that CT has received, there is still ongoing discussion about its definition and the set of skills that it encloses (Brennan & Resnick, 2012). While not exclusive, many researchers across the literature consider the following four skills as an important part of CT (Barr et al, 2011, Krauss & Kiki, 2017, Grover & Pea, 2013): a) decomposition which refers to the process of breaking down a problem into smaller parts that are more manageable and easier to be solved, b) pattern recognition in the terms of identifying and matching similarities (patterns) between different instances as a way of gaining extra information for them c) abstraction referring to the process of generalizing from specific instances by ignoring unnecessary details and keeping only their essentials so that a generic set is created and d) algorithm design, which refers to the ability to create a specific and effective process with step-by-step instructions, intended to be executed by a machine or even by another human.

Despite the long discussion and debate that has been done on grounding the theoretical basis of computational thinking, little attention has been given to the exploitation of known pedagogical theories for fostering its skills. In most cases, the development of CT is associated solely with closed coding activities and puzzles in which the 'making' element is limited or non-existing. However, this is a quite

strict approach if we consider that CT includes the cultivation of thinking skills and behaviors. To this end, we approach Computational Thinking through the constructionism lens, in which coding is seen as an expressive medium that enables the design, creation and sharing of complex personal artefacts (Papert 1980, Kynigos 1995). We also argue that constructionist designs that integrate coding with other affordances in a meaningful way (i.e. dynamic manipulation, robotic design, graphics creation etc), can become powerful tools for the acquisition and development of CT skills. One strategy that seems to be able to address the above challenge, but still has not been studied in the context of Computational Thinking is that of game modification or game modding.

Game Modding and Constructionism

Modding is a term coming from the world of gamers and refers to the process of modifying specific elements of a digital game to create a little or completely different version of it, which is called 'mod' (El-Nasr & Smith, 2006, Moshirinia, 2007). From a pedagogical point of view, modding shares a lot with constructionism. According to constructionism theory, students build their own knowledge structures as they are engaged in constructing a public entity (Pappert, 1991). These structures are seen as externalized expressions of ideas and thoughts which, through the sharing of the artefacts are becoming objects for discussion and change (Kynigos, 2015). Similarly, during the modding process students explore, decompose and analyse the game structure (i.e. the code, the skins, the mechanics etc) acquiring the necessary knowledge to make changes. Then they apply this knowledge to create and share a personalized artefact that depicts their own ideas and perceptions on the game. This game is then played, evaluated and discussed by others, probably leading to another cycle of modifications. We believe that this process can support students to generate and share meanings about the concepts involved in the game and also promote the development of skills like critical thinking, communication, abstract thinking etc. One difference from other constructionist activities is the process of deconstruction that precedes the construction. In game modding, students firstly break down the game structure to meaningful entities (i.e. the code, the skins, the mechanics etc) for analysis which then are modified and composed again for the creation of the mod. The process of deconstruction can put an added value to the learning process in the sense that children will use the deconstructed entities as building blocks to construct the personal knowledge which may differ from the original one (Boytshev, 2015).

A strong advantage of game modding compared to designing a game from scratch is that the modder can use the game's structures (code, database, media etc) as a starting point for understanding its development, eliminating in that way the learning curve which can be quite big in the case of game design. Moreover, the 'modder' is already familiar with the original gameplay and starts modifying it with a specific plan in mind. Depending on the type and the depth of the modifications, modding can require quite complex actions like analyzing the game structure, identifying the essential parts of the original game and those that can be modified, designing new game strategies and rules that are in line with the original gameplay etc. Several studies have explored the educational potentials of game modding. As the studies indicate, through game modding activities students acquired and implemented knowledge about several programming concepts (variables, event-handling, Boolean logic etc), but also from other fields like mathematics (3D geometry, vectors), art and physics (rotation, movement). In addition, students developed a number of skills like teamwork, iteration and refinement, debugging and error prediction etc. However, even though game modding seems a promising approach for promoting CT skills, limited related research has been done so far.

Progressive Engagement in CT skills

In order to design students' engagement in game modding activities, we adapted the three-stage progression pattern 'Use-Modify-Create' proposed by Lee et. al. (2011) for engaging youth in CT skills within rich computational environments. In this pattern, children are gradually engaged with the affordances of the environment transforming from users to creators. The approach is based on the idea that 'scaffolding increasingly deep interactions will promote the acquisition and development of CT' (Lee et. al. 2011). In this way, students tackle progressively higher design challenges as their skills and capacities increase. We argue that for a meaningful involvement in CT skills, game modding process should also follow a similar model of gradual engagement that would start from playing the original game, succeeded by small modifications or corrections and end up to the creation of the final mod which

can stand as a separate version of the game. Based on the model of Lee we propose a three stages model for progressive engagement with game modding: “Use-Fix-Mod Creation” (Figure 2).

In their model Lee et.al describe the “create” stage as an iterative process of testing, analysing and refining until the final artefact is created. We argue that a similar cycle is also followed in game modding during the stages of Fix and Mod Creation. We see the creation of a mod as an iterative process of several modifications each of which involves a cycle of the 4 steps: Test (Play the modified game) – Evaluate (evaluate the game for further modifications) – Analyse (analyse the game’s structure and components) – Modify (implement the new modification). After one modification is completed the same series of steps will be repeated until the final mod is created.

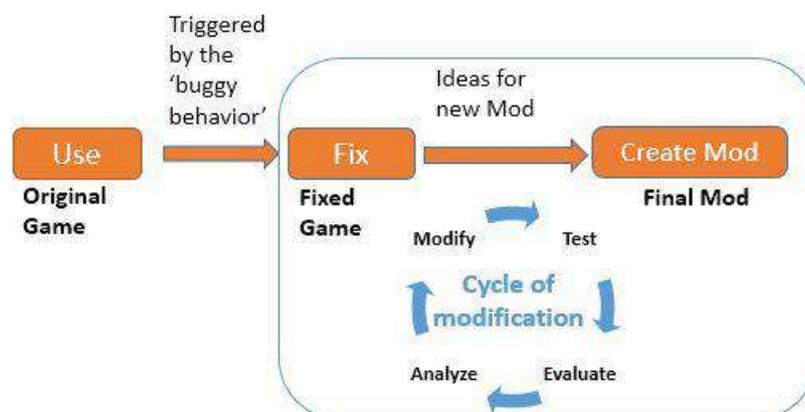


Figure 2. Three-stages of progressive engagement in game modding

In order to achieve the transition from “Play” to “Fix” and finally to “Create” stage, we are using the approach of “half-baked” microworlds which was introduced by Kynigos (2007) and refers to exploratory digital environments conceived, by design, to call for modification and change. Similarly, according to Kynigos & Yiannoutsou (2018), a “half-baked” game is designed with some characteristics that provoke students to modify it because they don’t like it. In our case, the ‘buggy behaviour’ of the game works as a trigger mechanism for engaging the students with the different game elements and urge them to make changes to them.

Related Work

Computational Thinking seems to consolidate its place in educational curricula around the world. A lot of countries, like Israel, Russia, New Zealand and Australia, recently revised their K-12 curriculum so it includes computational thinking (Grover & Pea, 2016). In addition, several groups are trying to make Computational Thinking accessible to everyone. For instance, in 2014 ISTE in collaboration with Computer Science Teachers Association developed the ‘computational thinking toolkit’, a complete collection of CT resources, from presentations to surveys and graphical animations, free to be used by anyone who wants to learn more about or to advocate for CT. Moreover, popular game design environments like Scratch and Alice, have been used lately for the development of computational thinking. For instance, the iGame after-school program aims to engage students in CT by designing their own computer games using Storytelling Alice (Lee et al. 2011). In addition, recent studies have been focused on the development of skills related to CT through students’ engagement in constructionist designs. For instance, Weintrop and Wilensky explore the emergence of programming abstractions through RoboBuilder, a constructionist video game in which the player uses block-based coding in order to play. Similarly, the tool eXpresser has been used for studying students’ engagement in generalization processes (Geraniou & Mavrikis, 2012). Finally, in some cases instead of designing a game from scratch, it is followed the approach of game editing or ‘remixing’. The term remix in game design has been widely implemented by the Scratch community referring to the process of building on an existed

Scratch project and repost it (Brennan & Resnick, 2012). Depending on the depth of remixing, it can require a high degree of sophistication and CT skills (Kafai & Bruke, 2014).

However, especially in the context of game design, there is still lack of research that studies computational thinking in the context of a) game modding and b) rich computational environments that integrate a number of affordances like graphics and media editors, data representations, coding etc.

ChoiCo: A constructionist tool for game design

Based on the above context we designed and developed ChoiCo (Choices with Consequences), an online authoring tool for designing and playing choice driven simulation games (Grizioti & Kynigos, 2018). In ChoiCo games the player revolves around different map-based settings making selections (these can be items, buildings, actions etc) each of which has specific consequences to a number of game attributes (Money, Health, Fun etc). The final target may vary, depending on how the game has been developed. For example, perfect-Ville (Yiannoutsou et. al, 2014) is a SimCity-like ChoiCo game which includes a set of city-sites such as “restaurant”, “library”, “work” etc. The player is a citizen of a city and has six attributes (Energy, Money, Hygiene, Fun, Social and Health) which should remain within specific limits. Every time that the player selects to ‘visit’ a site these attributes change accordingly until the game is over.

The modding features of ChoiCo

In ‘Design Mode’ the user can design a new game from scratch or modify an existed one. This environment is based on a carefully designed structure that aims to support computational thinking skills in multiple levels. More precisely the game design area is divided into 4 tabs each of which implements a core game element (Figure 3).



Figure 3. Modding of Game Interface in the Game Design Mode of ChoiCo

The first tab, called “Game interface” (Figure 3) contains the game scene, which is a map-based interactive tool and the game database, which contains information for all the items (rows) and how they affect the game attributes (columns). By using these two tools the user can make changes to the graphical interface of the game such as: change the background (which can be either a static representation or a map tile), add and modify items, add new areas and new player’s attributes etc. The three other tabs enclose the code for the game rules and behaviours in the form of block-based programming. Specifically, in the second tab is implemented the initial code which is executed only when the game begins i.e. the initial values of the player’s attributes. The third tab contains the code that describes the gameplay and it is executed every time the player selects an item. This code includes rules for manipulation of the items, transition between layers, feedback to the player etc (Figure 4). Finally, the fourth tab contains the ending rules of the game.

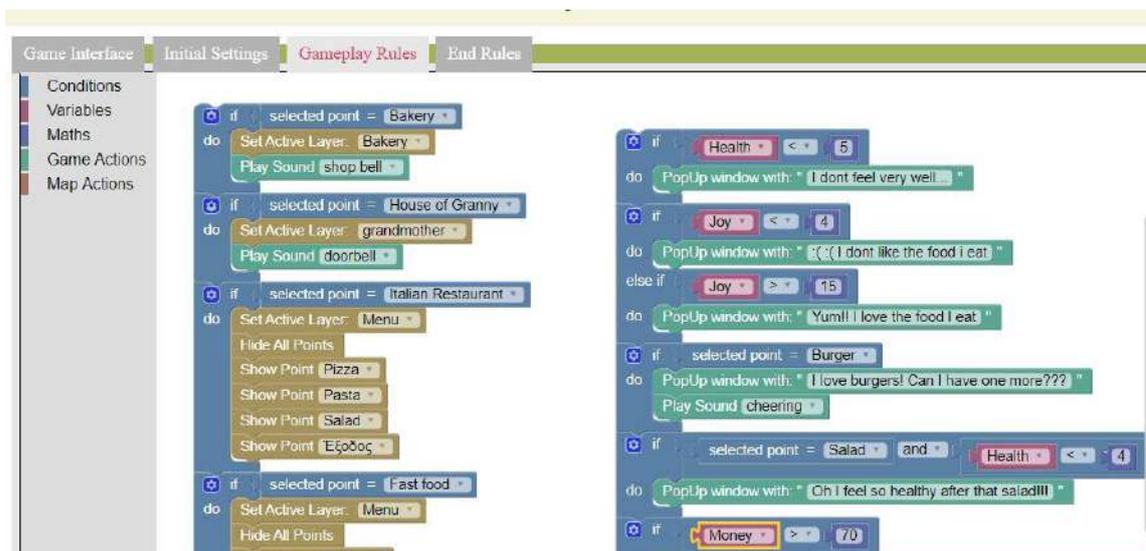


Figure 4. The "Gameplay Rules" in the Game Design mode of ChoiCo

The programming language of ChoiCo has been developed with Blockly Javascript library and apart from the basic programming structures it also contains blocks designed especially for game programming such as game over, hide/show item, set background, set active layer etc. The available blocks are divided into 6 categories ('Conditions', 'Variables', 'Maths', 'Game Actions', 'Map Actions' and 'Initialize') in order to ease the modding of different game elements. Thus, for example in category 'Map actions' there are blocks for modifying the game scene and its objects such as 'hide item', 'set active layer' etc.

Pilot Study: Context and methodology

In order to investigate how computational thinking skills emerge within game modding activities, we organized a pilot study with Junior High school students in which they modified a game created with ChoiCo. The main research questions we wanted to investigate with this study were:

- 1) Which computational thinking skills emerge and what meanings are generated during students' engagement in game modding activities with ChoiCo?
- 2) How students' computational thinking skills are being developed through the three different stages of engagement?

In the presented study we applied the method of Design-Based Research which includes the design of a pedagogical intervention and its evaluation in real classroom settings with the aim to refine the initial pedagogical design and to develop new theories (Barab & Squire, 2004). The research was carried out through three repeated cycles of design and implementation, utilizing every implementation as an opportunity for data collection, evaluation and review.

The 'half-baked' game and the modding activity

For supporting students' progressive engagement in modding activities we developed a 'half-baked' game in ChoiCo which was called "Eating out". The game represents a neighbourhood of an imaginary city with a number of different places to eat. Each place also contains a number of food choices. The aim is to keep the attributes 'Hunger', 'Health', 'Money' and 'Joy' between specific red lines that the player has to discover while playing. In order to give this game a 'half-baked' characteristic, we added on purpose the following two buggy behaviours in relation to its rules: A) An inconsistent item behaviour. The item "House of Granny" increases money by 15 without causing any damage to the other attributes. This behaviour is in contrast with the pattern of all the other items which will cause both positive and negative results to the score in order to keep a balance in the game flow. Thus the player can select this item unlimited times to get money with no consequences, giving to it a 'cheaty' role. B) A missing

ending rule. We didn't include a check condition for the attribute 'Hunger' in the code that implements the ending rules of the game. This is also contradictory with the game rules for the rest attributes.

We expected that these problematic behaviours would trigger students while playing the game to fix it and extend it. In addition, the detection and fixing of them require actions related to CT skills like recognizing patterns in items' behaviour, debugging and changing the algorithm for the ending rules etc. Regarding the modding process, it was divided into three main phases based on the three stages of progressive engagement described before. First students play the original game, then they make small changes and finally through repeated modifications they create their own mod. At the end of the second (fix) and the third (create) sessions, every team played the game of another in order to give feedback but also to discuss and exchange ideas about their artefacts.

Participants

The study was divided into three 3-hour sessions and took place in a public Junior High School as an after-school activity. The participants were 13 students, mixed boys and girls, from the third grade, aged 14-15 years old. They worked collaboratively in small groups of 2-3 in the school's computer laboratory using one computer per group. All of the students had little previous experience with block-based programming in Scratch. The ChoiCo tool had been presented to them during a math class.

Data collection and analysis

During the study, we collected qualitative data which included screen capturing files, audio recordings of each group and the files of students' modified games in different stages. Moreover, we followed the approach of artefact-based interviews for assessing CT skills and strategies (Brennan & Resnick, 2012). At the end of each session we did a semi-structured interview with every team aiming to urge discussion around three points: a) the changes that they had done so far to the game b) their involvement to computational thinking skills and c) their opinion and suggestions for the ChoiCo tool. Some representative questions are for instance: 'Did you change the number of layers and why?', 'Did you group your items in a specific pattern?', 'What difficulties did you face in programming the rules?' etc. Finally throughout the study students filled out a worksheet with their modding plans, their decisions and their final implementations.

For the analysis of the data, a qualitative analysis method was followed. More precisely, the transcripts of audio files and interviews were analyzed with the tool Atlas.ti using a set of codes divided into three main code categories a) computational thinking skills, b) modding ideas and c) interactions with the tool. The unit of analysis was the critical episode, which refers to representative moments that indicate students' engagement in one or more of the above three categories. In addition, students' games were analyzed according to the progression of the implemented modifications across the different modding stages. This data was compared with students' replies in their worksheets about their modding progress and plans.

Findings

At the end of the study, there were five different mods created, one by each team. The mods are briefly presented in table 1 with a short description and the modifications they had. It is worth mentioning that in their final mod all teams had made significant changes to the game elements with the buggy behaviour in the initial half-baked game: The Items' Values and the Ending Rules. This fact shows that the half-baked elements of a game are possible to affect the modifications that students will decide to make. Moreover, four of the mods had new attributes, while in three of them the students also added new layers and designed a new setting or story. Despite the big or small changes they made, all of the teams maintained some elements from the initial game especially in the code and in the game attributes, which helped them compare and evaluate their games both with other teams' and with the initial one.

Table 1. The 5 mods created by the students

Mod Name	Game description/Attributes	Modified Elements
Eating out 2	Choose between different restaurants in your city. Attributes: Joy, Money, Health, Hunger	Items Set, Items' Values, Ending Rules
I am hungry	Make the right food choices. Attributes: Joy, Money, Health, Hunger, Free Time	Items Set, Items' Values, Ending Rules, Game Attributes
Travelling in Europe	Visit as many European cities as possible. Attributes: Joy, Money, Tiredness	Items Set, Items' Values, Ending Rules, Game Attributes, Layers, Game Concept
School Survival	Make the right choices in order to survive a school day. Attributes: Joy, Boredom, Health, Knowledge, Danger	Items Set, Items' Values, Ending Rules, Game Attributes, Layers, Game Concept
Our School	Get in different school classes without getting bored. Attributes: Joy, Education, Health, Stress	Items Set, Items' Values, Ending Rules, Game Attributes, Layers, Game Concept

The data analysis indicated the emergence of CT skills during the three stages of engagement. Table 2 depicts the CT skills that were detected in each stage according to the results of the data analysis. Even though the skill of decomposition was mostly present during the third stage of mod creation, for the three other skills there was evidence of a progressive development from the first to the third stage. As progress, we don't consider a numeric increase in the number of related critical episodes, even though there was one, but a gradually deeper and more meaningful engagement with CT skills during the modding process. To further explain this we present this progress through characteristics example for each of the three CT skills, pattern recognition, abstraction and automation.

Table 2. Emerging Computational Thinking Skills across the three-stages of progressive engagement in game modding

Stage	Emerging CT skills
Use	Pattern Recognition, Automation
Fix	Pattern Recognition, Pattern Implementation, Automation, Abstraction
Create	Pattern Recognition, Decomposition, Pattern Implementation, Abstraction, Automation

From Pattern Recognition to Pattern Construction

Since patterns play an important role in games, and especially in ChoiCo games, students' engagement with them was present throughout the study. While they were playing the game, all groups recognized patterns related to the behaviour of the items in the game and how they affected their score. For instance, some patterns expressed by students of groups 2 and 3 are "Health is decreased inversely with Joy", "All the tasteless foods increase your health but reduce the joy, while the tasty ones do the opposite". Three of the five groups also detected the problematic behaviour of the 'house of granny' choice which they described it as an inconsistency of a related pattern. Maria from group 3 mentions as she plays the game "Look! None of the other buildings gives you money except the house of granny. That's a cheat!". She recognizes a common behaviour of all the buildings in the game (pattern) and since 'house of granny' is a building should also follow. George, a student from group 1 says to his teammates "We can go in and out from the house of granny forever and never lose! Unlike all the other items of the game, this one has only positive effects on your score!". George describes a different pattern from Maria which should also have been followed by this item as it is a pattern for all the items of the

game. Both students realize that even if a single item doesn't follow the designed game pattern, it can easily destroy the gameplay. During the Fix stage students corrected these patterns in the data table values while some of the groups experimented with the patterns of other attributes. In this stage, they progressively passed from pattern recognition to pattern implementation which in the third stage became pattern construction. In the stage of mod creation, three of the groups constructed new patterns in the database values. One example comes from group 2 who created two categories of patterns as shown in Figure 5. The first category included patterns for each attribute by item category i.e. "The city points will have zero effect on the Joy variable. Hotels will increase it a lot, but Hostels or Motels will decrease it. Tourist sites will increase it a little bit". For the design of this pattern, they modified accordingly all the values of one column (attribute) in the database and that's why we called them 'Vertical Patterns'. The other category included patterns of relations between attributes either for an item category or for all the game items. For instance 'Tiredness will be Money/10 so that they are proportional'. These are the 'Horizontal Patterns' as students applied the pattern to the related cells for each row (item) of the database.

ID	Description	Joy	Money	Tiredness
42	London	0	-100	10
51	Barcelona	0	-105	11
55	Instabul	0	-20	2
59	Hotel 1	40	-100	-5
63	Hotel 2	15	-80	1
67	Hostel	-20	-15	15
71	Motel	-45	-5	20
75	Colosseum	15	-10	5
79	Batican	20	-15	10

Figure 5.2 Patterns created in the database values (translated version of the game)

From specific items to abstract classes

In the beginning, students used the skill of abstraction combined with pattern recognition to identify categories of game elements such as items categories or code functions. Later during the stage of creation, they firstly designed and then implemented their own abstract categories (classes) of items and layers in the game. We will present as an example the mod 'School Survival' from group 1. In this game, the player is a high school student and has to make choices related to school behaviour (miss/attend a class, sleep, solve an exercise, talk with a classmate etc.) Every choice may reduce or increase the five attributes: Joy, Boredom, Health, Knowledge and Danger. When this group was planning their mod, they described on their worksheet a region category with the name "Classroom". Then they explained in the interview that "Classroom" is an abstract model that will help them for the creation of the school classroom layers because all classrooms must have some common characteristics. These were: A background image that represents students in a classroom, 3 choices of actions an 1 point 'Exit' for returning to the corridor region. Moreover, the entrance to a classroom reduces the 'Danger' attribute. Students used this model to create three different classroom instances (Figure 6).

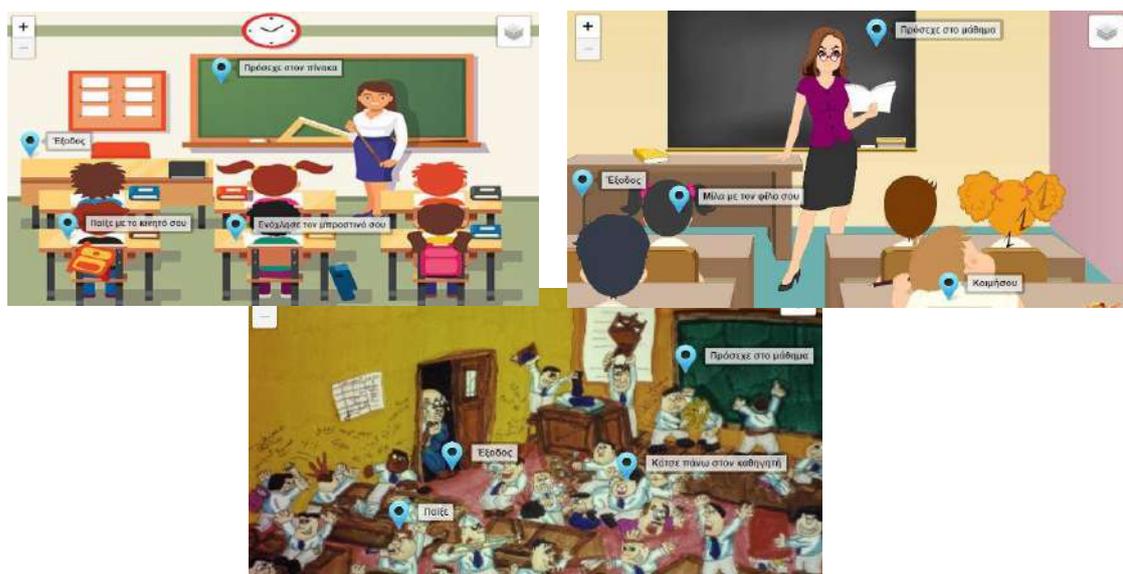


Figure 6. The 3 instances of the abstract category "Classroom" in the game 'School Survivor'

In addition, they created three abstract categories for the actions according to their consequences, before they add the actual actions in the game. These were: High Risk Actions (They increase a lot the attribute Danger, decrease the attribute Knowledge but also increase the attribute Joy), Low Risk Actions (They increase a little the attribute Danger and decrease the Boredom) and Good Student Actions (They increase knowledge but also boredom and decrease Danger but also Joy). Then, when they wanted to add a new action they created it according to these three categories. They also tried to keep a balance between these three categories in the game.

From simple descriptions to complex algorithms

From the first to the last stage, students applied algorithmic concepts in order to describe, modify and develop game rules, with the most common being conditional statements (if..else), logical expressions, the sequence of commands, and variables in the form of game attributes. An important outcome for all groups is the gradual transition from a verbal description of game rules to their implementation with programming blocks. This process led students to generate meanings about programming and algorithmic concepts, but also to develop their automation CT skill. To elaborate on this outcome we will (again) use group 2 as an example who created the mod 'Travelling in Europe'. During the first stage students detected and wrote down to their worksheet the basic rules of the game such as «If your health is lower than three you lose» «Every time you make a choice, it will reduce some of your four needs. If some of them become lower than zero then you lose». In this stage, students make a free description of the algorithmic sequence and of the conditional statement. Then in the Fix stage, they matched these rules to the relevant blocks of the game code. In this stage, they also made some small changes to the existed rules including modifications to logical conditions and the addition of an extra rule for the variable 'Hunger'. In this stage, they transit from the free algorithm description to the recognition and modification of the corresponding parts of the code. Finally, during the Create stage, they constructed new rules based on those they had already explored and modified. Some of the new rules were even more complicated than the initial ones regarding the use of the conditional statement. Figure 7 summarizes this gradual engagement in algorithm building.

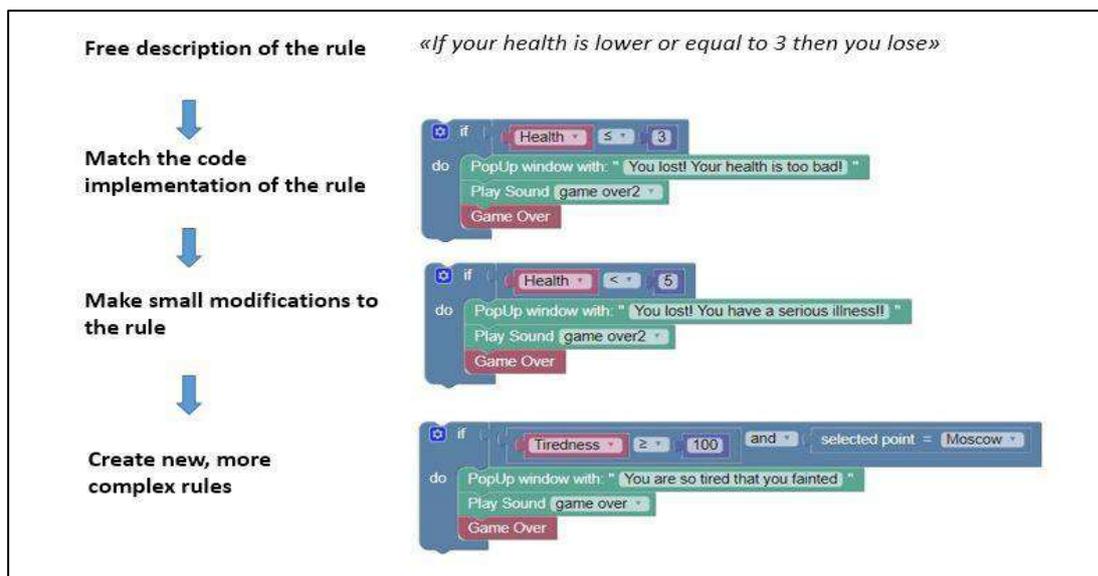


Figure 7. Progressive engagement with the conditional statement

During the create stage there was another interesting outcome for the algorithm building CT skill. While students were designing their mod they imagined and sometimes tested, possible scenarios of play to improve the game rules and flow. This led them to think the gameplay as an algorithm and to apply coding techniques such as event handling and error prediction both to the code but also to the database values. This was a complex and extensive procedure which required students to continuously switch between the three affordances (map, database and code) but also switch between the roles of designer, player and evaluator.

In the example below, students of group 1 discuss the initial and limit values of the attribute “Danger”.

S1: So, what should be the initial value of Danger (tab 2)

S2: Let's set it to 0

S1: But if you start with 0 and you choose the point 'leave school' first you will immediately lose...because...look (switch to the 1st tab)... “Leave School” gives +5 to Danger

S2: Yes ok. And why will you lose?

S1: Because here (switch to 4th tab) we have programmed it to lose if Danger is greater or equal to 5

S3: Well ok. We can just change the condition and make it greater or equal to 100

S1: Yes but not 100! It is too high. (Switch to 1st tab) All the items increase or decrease danger by 5-10 units. It is too hard for a player to reach 100. Make the limit 50.

In the above episode, students switch between the three tabs trying to predict the possible sequence of play, thus possible executions of the game algorithm. Then they discuss on algorithmic concepts which apply to their program aiming to create a meaningful game. This process indicates a progress in automation skills and algorithmic thinking. Students not only have the knowledge to use the conditional statement, but they also express their ideas for it and implement it in a meaningful form for them.

Conclusions and Discussion

In this article, we explored the emerging computational thinking skills through the process of game modding with ChoiCo tool. We presented a case study in which students were progressively engaged in the three stages of modding “Use – Fix- Create Mod”. The modding activity was developed around a half-baked game, designed with two intentional bugs to urge students to change and extend it. The results of the data analysis revealed some significant outcomes regarding CT. First of all, it seems that

this gradual engagement in construction process acted as scaffolding for the development of some important computational thinking skills. While students are transforming from players to creators so does the way they apply CT skills. In the first stage students used CT skills to understand and analyze the game structure (pattern recognition, categorization) which in the next stages utilized for meaningful constructions (pattern generation, the creation of abstractions). Moreover, the analysis showed that the half-baked approach can contribute to game modding activities in terms of motivation but also guidance for the modifications. Especially for the transition from play to fix stage, the half-baked elements of the game worked as 'trigger' mechanisms for the students. After detecting the problematic behaviour they wanted to fix it according to their own solutions and to even to improve it with new ideas. It also urged them to focus on specific game elements and guided them on avoiding the same mistakes in their mods.

As emerged from the analysis, all of the three affordances of ChoiCo environment played a significant role in the development of students' CT skills. The use of the map-based tool for the modification of the game scene, items and regional layers seemed to foster students' perception and implementation of abstractions. In addition, the representation of game items and their properties in the form of a simplified database urged students to use, process and handle a big number of data. This led them to discover, match and apply patterns, reinforcing pattern recognition and pattern generation skills. The block-based programming involved students with algorithmic thinking and automation. The categorization of blocks according to their role in game design as well as the feature of special game blocks, facilitated the creation of complex programming structures for automating the game rules. The analysis also revealed some limitations which need further investigation. Students applied the skill of decomposition only during the third stage, even though it could have emerged earlier. Even in the third stage its presence was quite restricted and not by all teams. With respect to coding, even though students made extent use of the conditional structures, the use of other programming structures like repetition or internal variables was very limited. These outcomes indicate the need for redesigns in the modding activity with more focus given on the specific aspects.

This study showed that game modding has significant potentials as an educational approach for fostering computational thinking skills. However, more research needs to be done in this field that would examine how the game genre and the game elements that are modified affect the development of such skills. Moreover, the results revealed that the integration of different affordances can provide a rich context for the development of CT skills. Thus, further research is necessary for designing and utilizing integrated designs for computational thinking, not only in the field of game modding but also in other areas that engage students in design.

Acknowledgements

This research has been financially supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 531)

References

- Barab, S., & Squire, K. (2004). Design-Based Research: Putting a Stake in the Ground. *Journal of the Learning Sciences*, 13(1), p. 1–14. https://doi.org/10.1207/s15327809jls1301_1
- Barr D., Harrison J. & Conery L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology* 38(6), p. 20–23.
- Boychev, P. (2015). Constructionism and Deconstructionism. *Constructivist Foundations*, 10(3).
- Brennan K. & Resnick M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada. p. 1-25.

- El-Nasr M.S., and Smith B. K.. (2006). Learning through Game Modding. *Computers in Entertainment (CIE)* 4(1) p. 7.
- Geraniou, E. and Mavrikis, M. (2015). Building Bridges to Algebra through a Constructionist Learning Environment. *Constructivist Foundations*, 10(3), p. 321-330.
- Grizioti M, Kynigos, C. (2018) Game Modding for Computational Thinking: An Integrated Design Approach. In: *Proceedings of the 2018 Conference on Interaction Design and Children. ACM, 2018*
- Grover S., and Pea R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42(1), p. 38-43.
- Kafai B. Y & Burke Q. (2014). *Connected code: Why children need to learn programming*. Mit Press.
- Kynigos Chronis. 1995. Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies. *Computers and Exploratory Learning*, diSessa, A, Hoyles, C. and Noss, R. (eds), Springer Verlag NATO ASI Series, 399-420.
- Kynigos Chronis. 2007. Half-baked logo microworlds as boundary objects in integrated design. *Informatics in Education* 6,2: 335–359.
- Kynigos, C., & Yiannoutsou, N. (2018). Children challenging the design of half-baked games: Expressing values through the process of game modding. *International Journal of Child-Computer Interaction*. Krauss J. & Prottzman K. 2017. *Computational Thinking and coding for every student: The Teacher's Getting-Started Guide*. Corwin Press.
- Lee I., Martin F., Denner J., Coulter B., Walter A., Erickson J., Joyce M., and Werner L.(2011). Computational Thinking for Youth in Practice. *Acm Inroads* 2(1), p. 32–37.
- Moshirnia A. (2007). The Educational Potential of Modified Video Games. *Issues in Informing Science and Information Technology* 4, p. 511–521.
- Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Yiannoutsou Nikoleta, Kynigos Chronis and Daskolia Maria. (2014). Constructionist Designs in Game Modding: The case of learning about Sustainability, in: *Proceedings of Constructionism 2014*. 19-23
- Yucel, I., Zupko, J., & El-Nasr, M. S. (2006). IT education, girls and game modding. *Interactive Technology and Smart Education*, 3(2), p. 143–156.
- Weintrop, D., & Wilensky, U. (2014). Situating programming abstractions in a constructionist video game. *Informatics in Education*, 13(2), p. 307.
- Wing J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3) p. 33-35.
- Wing J.M.(2008). Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(188), p. 3717–372

Programming Approaches to Computational Thinking: Integrating Turtle Geometry, Dynamic Manipulation and 3D Space

Marianthi Grizioti , mgriziot@ppp.uoa.gr

National and Kapodistrian University of Athens Educational Technology Lab, School of Philosophy, Faculty of Pedagogy, Greece

Chronis Kynigos , kynigos@ppp.uoa.gr

National and Kapodistrian University of Athens Educational Technology Lab, School of Philosophy, Faculty of Pedagogy, Greece

Abstract

During the last decade, coding has come to the foreground of educational trends as a strong mean for developing students' Computational Thinking (or CT). However, there is still limited research that looks at coding and Computational Thinking activities through the lens of constructionism. In this paper, we discuss how the knowledge we already have from other thinking paradigms and pedagogical theories, such as constructionism and mathematical thinking, can inform new integrated designs for the cultivation of Computational Thinking. In this context, we explore students' engagement with MaLT (Machine Lab Turtle-sphere)⁴⁸, an online environment of our design that integrates Logo textual programming with the affordances of dynamic manipulation, 3D graphics and camera navigation. We also present a study on how the integration of the above affordances can promote constructionist learning and lead to the development of CT skills along with the generation of meanings about programming concepts.

Keywords

Logo geometry; computational thinking; programming; dynamic manipulation; 3D graphics

Introduction

Programming as skill and as a learning activity is connected to technological changes (the shift from web to web 2.0, society's acquaintance with technology) and to cultural trends. Back in the 80's, many schools showed great enthusiasm in teaching their students computer programming, while by the mid-1990s schools turned away from programming, mainly because of a lack of subject-matter integration and a lack of qualified instructors (Noss & Hoyles, 1996). When software tools with dynamic manipulation and icon-driven technology appeared, programming was seen as a kind of unnecessary noise to doing interesting things with digital media (Kynigos, 2015). However, during the last decade, we are facing a big comeback of programming (Kafai, Bruke & Resnick, 2014) in a new context in which it is considered as an integral part of a new computational literacy. An advanced view about computer programming has become popular; that every child can and should learn to write code as a way to develop their computational thinking. The idea of computational thinking that Wing described in 2006 as 'a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applicable in the information society', is now considered as a necessary supply for the 21st-century students (Barr, 2011; Kafai, Bruke & Resnick, 2014; Lee et.al. , 2012). However, there is still need for exploring pedagogical approaches and strategies for the acquisition and improvement of its skills (Brennan & Resnick, 2014).

In this paper we argue that the challenge remains, both for designers and educators, to consider what kinds of learning processes and to what end students may engage with the new programmable media. It seems to be a good time, with all this enthusiasm surrounding computer programming, to utilize these

⁴⁸ <http://etl.ppp.uoa.gr/malt2/>

new opportunities for learning to code, with a constructionist approach, based on Papert's generic vision and ideas for learning (Papert, 1980). Constructionist design approaches for educational tools, aim to engage users of all ages in the construction of personally meaningful artefacts through programming (diSessa, 2001; Papert & Harel, 1991). To this end, we propose the approach of *integrated constructionist designs* in which programming is combined with other affordances forming a meaningful and enriched computational environment. The affordances are selected and implemented in a way that they engage students with scientific concepts and at the same time promote the development of wider skills, like these included in CT. In integrated designs, students use the different affordances, including coding, as means for self-expression, communication of ideas and design of personal artefacts.

Rethinking Programming Approaches to Computational Thinking

All this recent enthusiasm around Computational Thinking and 'coding for all', has led to a wide development of new educational software and applications that aim to engage students of all ages with programming. Block-based programming tools such as Scratch, Alice, ToonTalk and apps like Kodu and LightBot have become very popular and have attracted many young people, but also teachers and adults, to write simple programs with them. At the same time campaigns like the 'Hour of Code' and 'EU code week' are organized every year to introduce young students to programming.

The research on computational thinking points out the importance of active engagement, construction and exploration in programming activities (Brennan & Resnick, 2014; Resnick, 2014; Stager 2014). In many cases though (with the exceptions of Scratch and Alice), students' engagement with programming is attempted through a series of closed quizzes and puzzles with the "making" and the social elements being limited. In addition, block-based programming can also have significant limitations for older or more experienced students (Weintrop & Wilensky, 2015). On the contrary, former pedagogical designs like textual programming are left aside regarded as obsolete. This is confirmed by the fact that there is still limited research focusing on the Computational Thinking skills that can be promoted with textual programming like i.e. Logo language.

The question that arises is how these 'obsolete' but yet deeply studied approaches, can contribute to the design of new coding tools by providing a strong theoretical background. For instance, a large number of studies have shown the benefits of Logo textual programming and the effectiveness of turtle geometry in offering rich mathematical experiences and encouraging the construction of meaning (Clements et al. 2007, Kynigos 1995, Papert 1996). We argue that if coding is seen as a new kind of expression and mediation of meaning, then it may be worthwhile investing on designing ways to make formalism functional and meaningful through its connection to other representational forms in a dependent way. New educational tools for programming could for instance maintain the basic design principles of conventional Logo-based designs and be enhanced with new technologies in parts which are meaningful for the learning process such as dynamic manipulation, a variety of what is programmed (e.g. robots, devices, digital objects with properties, behaviours and interactions in diverse fields), 3D representations etc. In such tools, the development of learner's skills would be achieved through authentically creative and constructionist activities and coding will be seen a vehicle to enhance them.

Towards an integrated approach

Computational Thinking is strongly connected with other known thinking paradigms such as mathematical thinking, engineering and scientific thinking and algorithmic thinking (Wing 2008, Hu 2011). Thus, for the development of the integrated designs described above, it is quite important to investigate how grounded implementations and approaches from these paradigms may contribute.

Let's take for example the element of abstraction. Abstraction is considered by many researchers an important CT skill and the 'mental tool of computing' (Wing, 2006). However, the concept of abstraction and abstract thinking is not new at all. Abstraction and generalization are core concepts of mathematics. Despite the differences that mathematical and programming abstraction may have (Wing, 2008), the long year research and the theoretical constructs from mathematics education, can be a strong basis for the development of tools and methods that foster students' abstraction skill. An interesting method that comes from the world of the Dynamic Geometry Environments (DGE's) is the dynamic manipulation of geometrical constructions. Dynamic manipulation helps students to understand the properties of

geometrical objects and generalize their rules and the relationships between them (Goldenberg & Cuoco 1998, Psyharis & Kynigos, 2009). This is particularly interesting with respect to thinking about abstraction as a process of defining or approaching a construction as 'how it behaves when manipulated dynamically'. We argue that dynamic manipulation could also be beneficial in the context of programming abstractions and Computational Thinking, even though it has not yet been studied enough.

Another example from mathematics that requires a number of mental skills is that of creating and manipulating 3D geometrical shapes. When students interact with 3D models they have to use spatial thinking which includes complex mental processes like the capability of understanding and recognizing the location and shapes of the objects, their relations to each other and their movement in space (Hauptman, 2010; Kynigos, 2007). To achieve that they usually apply skills like pattern recognition or decomposition which are also included in CT (Lee et al. 2011). Thus the programming and manipulation of 3D geometrical objects in a digital environment could also be a technique for supporting the cultivation of such skills.

The above examples indicate that the integration of digital affordances and concepts from other fields like mathematics can possibly be beneficial for CT skills. In our research, we try to investigate and reveal the common elements between mathematical, algorithmic and computational thinking paradigms and study their possible development through such integrated designs. In the presented study we focused on four skills which are considered by many researchers across the literature, as an important part of a learners' skillset (Lee et al. 2011; Krauss & Prottzman, 2017). These are a) decomposition, b) pattern recognition, c) abstraction and d) algorithm design.

MaLT: Logo Programming of 3D dynamic representations

In order to investigate the benefits of such integrated approaches to both programming knowledge and CT skills, we organized a study with MaLT (Machine Lab Turtlesphere), an online Logo-based application of our design that allows the creation, exploration and dynamic manipulation of 3D geometrical models with textual programming. MaLT provides a learning environment that endorses the benefits and the strong theoretical basis of Logo programming but also extends it with new technological features in such ways that could enhance the development of CT skills.

More precisely it implements the classic Turtle Geometry of Papert, extending it with the integration of the following three affordances: 1) The turtle becomes a sparrow. The designed Logo language supports the coding of 3D drawings by navigating a sparrow (instead of a turtle) in a 3D spherical space. This feature broadens the range and the complexity of objects that can be programmed with the Logo language. 2) The drawings become alive. MaLT implements a dynamic manipulation system for the Logo variables, allowing the animation of any 3D model that has been created by a parametric procedure (i.e. *cube :x*). To use the dynamic manipulation the user clicks on any drawing on the scene and activates the "sliders" tool. This tool contains a number of sliders, one for every parameter of the Logo procedure that creates the clicked model, which can be varied between a max and min limit. The variation of each slider, will result in an immediate re-execution of the Logo procedure with the new input and thus to an animation of the model on the scene. The aim of dynamic manipulation in MaLT is to reinforce the process of abstraction by means of kinaesthetically causing the continuous transformation to a structure described formally as parametric to make better sense of how this may represent a generality, such as e.g. a property of a geometrical figure 3) The view becomes periscopic. The application allows navigation in the 3D space where the avatar moves with a periscopic camera. The navigation in 3D space requires a number of skills like perceptual constancy (the ability to recognize some properties of an object regardless of its size, position or color), spatial orientation (the ability to realize how an object would seem from a different point of view and recognize it even if it has been i.e. rotated or dispositioned), visual discrimination (the ability to compare multiple objects in space and recognize similarities and differences between them) and the perception of spatial relationships (Kynigos & Latsi, 2012; Lohman 1988).

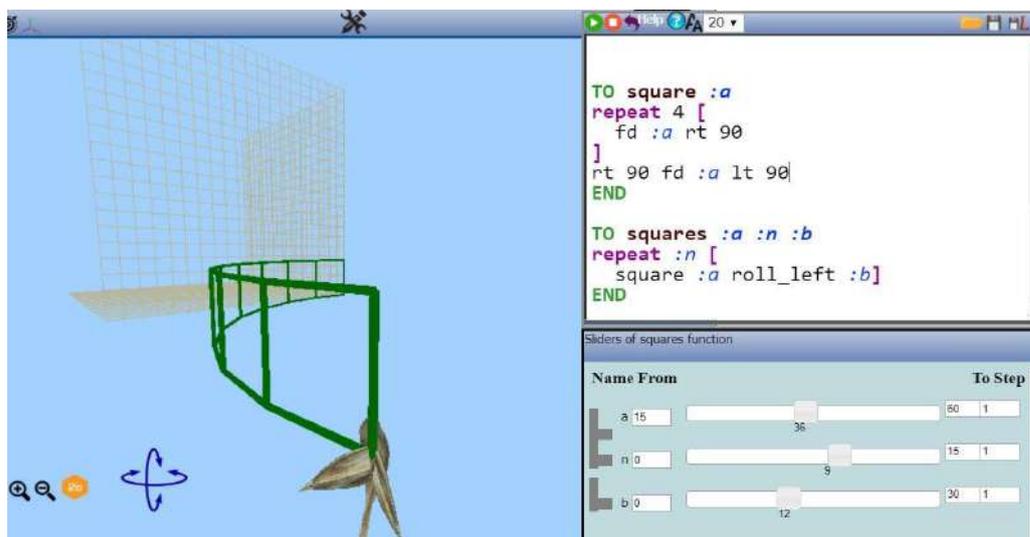


Figure 2. The different affordances of MaLT: Logo programming, 3D graphics, Dynamic Manipulation and Camera Perusal

There are three basic elements that comprise the MaLT environment are: The “command editor”, the “3D scene” and the “variation tool” (Figure 2). In the command editor, the user can write programs in order to navigate the avatar on the scene. The programming language of MaLT is based on MSW Berkeley Logo expanded with the feature of 3D graphical representations, including special commands for changing sparrow’s direction in the 3 dimensions (e.g. *up 90*), importing 3D objects while moving in space (e.g. *cube 100*), changing pen color and thickness (e.g. *setpenwidth 4*) etc. The 3D scene is where the sparrow moves as the commands are executed, with the same logic as Papert’s turtle moves and draws on a 2D canvas. The scene contains a periscopic camera, with which the learner can navigate in the 3D scene and observe the created models from different points of view. With the variation tool, the user can provoke dynamic changes to shapes created by the execution of a parametric command (e.g. *pyramid x w*). The tool provides a number of sliders, one for every parameter of the specific procedure, which can be varied between a max and min limit. The above features make MaLT a rich computational environment which extends turtle geometry beyond classic 2D designs and uniquely leads it to the modern world of online sharing, 3D graphics and animation.

The pilot study

The integration of the above affordances can provide a new constructionist context for the development of computational thinking skills through the engagement with programming and mathematical concepts. To further investigate this we organized and conducted a study in a Greek junior high school. The aim of the study was to answer the following questions:

- How the integration of the three affordances with Logo programming contributes to students’ development of computational thinking skills.
- How students use and construct meanings about programming concepts while they collaboratively create and share artefacts with the above affordances.

The total duration of the study was 9 hours divided into three 3-hour sessions and it was organized as part of the school’s mathematical club. In the study participated 9 students aged 13-15 years with small previous experience in Logo programming, high experience and interest in mathematical problems and no previous experience in 3D geometry. The students worked collaboratively in 4 groups of 2-3 in the school’s computer laboratory using one computer per group.

The research method we used is that of Design-Based Research which is evolved from design experiments (Cobb et al, 2003) and includes the design of a pedagogical intervention and its evaluation in real classroom settings with the aim to refine the initial pedagogical design and to develop new theories. The research was carried out through repeated cycles of design and implementation, utilizing

every implementation as an opportunity for data collection, evaluation and review for the next design. Design-Based Research focuses mainly on the collection and analysis of qualitative data as the objective is to identify the main characteristics and the different facets of the designed intervention when implemented with students.

Description of the activities

For the purposes of the study, we designed a set of activities with MaLT which aimed to foster students' computational thinking skills through creative construction with all of the three affordances. The activities were divided into main three phases.

The first phase is more of an introduction to the environment, aiming to get students familiar with MaLT environment and its affordances through simple Logo coding tasks. They begin by designing shapes in 2D, like a square, and create new functions. Then they pass to 3D space by using their 2D models to create 3D ones, like a cube. After that, they design 'give life' to their models by adding one or more parameters to their procedures so that they can manipulate them with the sliders of the variation tool, creating animations.

At the second phase students design and program a dynamic cube model that animates smoothly from a 2D cube net to a 3D cube. Through this activity, students investigate the transition from two to three-dimensional design and implement both mathematical and computational concepts. With respect to the mathematics, they have to use the concept of angle and rotation in space. With respect to programming, they explore the role of variables and constants in a program. They also have to think in an abstract way in order to design an algorithm that will produce the correct model for every input. The way they will follow to construct the model is open to the students and they have to decide it through experimentation with the environment and communication in their teams. There were no instructions given to them and the role of the teacher was more supportive.

Finally, the third phase is quite more open as students have become familiar with the tool. Students use parametric procedures as building blocks to design a 3D animated drawing of their choice. This is the phase where we expect students to express their personal ideas and meanings through programming. To implement their idea they would need to use more complex programming concepts and also recognize and combine properties of different 3D objects.

Data collection and analysis

During the study, a set of data was collected for analysis that included screen capturing files, audio recordings of each group, teacher and researcher observation notes and files of students' code at the middle and the end of each session. A qualitative analysis method was followed for the evaluation of all the data. More precisely, the transcribed audio files and observation notes were analyzed in Atlas.ti tool with a set of codes related to a) the computational thinking skills of abstraction, pattern recognition, decomposition and algorithm design, b) mathematical and programming concepts of the activities (variable, parameter, angle etc) and c) students' interaction with the three affordances (i.e. slider use, creation of new function, camera rotation etc). From the analysis clusters of critical episodes emerged. As a critical episode, we consider as representative moments of the student's interaction with MaLT affordances and of their engagement with CT skills. Moreover, the Logo codes of students were analyzed by comparing their progression from the first to the last session with respect to abstraction, pattern recognition and use of variables and repetition structures.

Findings

The analysis revealed that students applied the skills of abstraction, pattern recognition and decomposition especially during the second and third phases of the study. All of the teams recognized and implemented patterns related to the 3D shapes both in the scene and in the Logo code. In addition, they deconstructed 3D complex models to smaller parts or to 2D shapes (i.e. the cube to squares) which then used again to create other models (i.e. a square pyramid). Finally, they programmed and designed abstract models which, with the help of the sliders, could generate a number of successive instances creating an animation on the scene.

Moreover, the students used and discussed programming concepts, with the most common being the variable as a parameter, the procedures and sub-procedures and the error handling of different inputs. Students made an extended use of the sliders tool for debugging their code but also for experimenting with the graphical outcomes of multiple parametric procedures on the scene. In this section, we present some characteristic examples from the analysis regarding both the emerging CT skills and also the applied programming concepts. In the presenting episodes we don't use the real names of the students but randomly selected aliases.

Uses of abstraction

In the context of computational thinking, abstraction is described as the process of defining patterns, generalizing from specific instances and capturing of common characteristics or actions into one set that can be used to represent all other instances (Kafai, Bruke & Resnick, 2014, Krauss & Prottzman, 2017). In our case, abstraction was detected mainly in the following cases:

- When students used the sliders to animate an existed parameter of their model
- When students decided to add a new parameter of animation to their drawing

One characteristic example occurred during phase 2 when students designed a dynamic cube net that would be able to transform from 2D to 3D space by dragging the sliders. At this moment all groups had already programmed non-parametric 3D cubes and 2D cube nets. Thus, they had to find a more abstract solution that would combine the 3D and 2D model to one. According to the video and audio analysis, students worked in two ways.

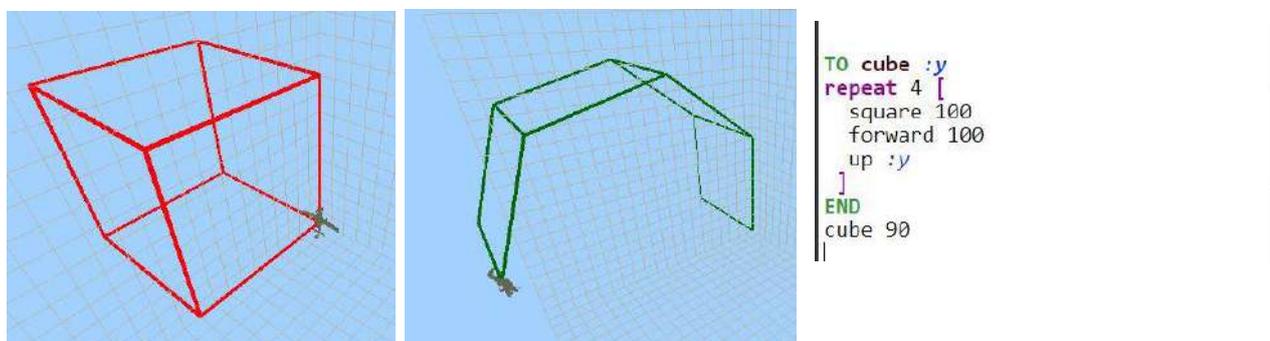


Figure 3. Transition from cube instance to cube net with the use of variation tool

Two of the groups decided to start by analysing the 3D cube shape on the scene and how it can be 'unfold', while the two others begin by the code of the 2D cube net. One example of the first case comes from group 1. In that group students first recognized two important properties of the created cube shape: its size and its dihedral angle (the angle between cube edges) which is always 90 degrees. After that, they mapped these two properties to the commands of 'cube' program mentioning that the commands *forward 100* and *square 100* are related to cube's size while the command *up 90* is related to its dihedral angle (Figure 3). Then they replaced the number 90 with the variable *:y* in all the *up 90* commands because as S1 said: "By changing the up angle we will be able to unfold the cube". However, their initial cube was created by only 4 squares resulting to a wrong cube net. Thus they added two extra squares to the procedure.

In this approach, we can see the implementation of abstraction both in the categorization of the cube's properties (size, angle) and also in the replacement of the appropriate numbers with variables. Students mapped the properties of the 3D representation to the code and decided what is important and what is not for the model they wanted to design. They also realized that the outcome of a procedure may seem right for a specific input (cube) but not for all the others (cube net).

Two of the groups followed a different method for the same problem, starting with the static 2D cube net instead of the cube. They analysed the shape of the scene and the code on the editor, in order to decide where in their code should add an extra command. As Helen from group 2 mentions "we should add an up command right before every square so that the bird would turn up some degrees before it

draws the next square". Helen and her teammate tested their idea by adding the `up 90` command in their code, then change it to `up 0`, and then to `up 45` to see if their model is correct for different angles. When they had found the correct model they generalized their algorithm by replacing the numbers of `up` command with variables.

T: Why did you write up 45?

H: Because when we wrote up 90 we couldn't see clearly what's wrong

A: And with the up 0 it was 2D and the problem was not visible.

H: Yes... And with that number, we can see better how it would fold when it will animate with the slider.

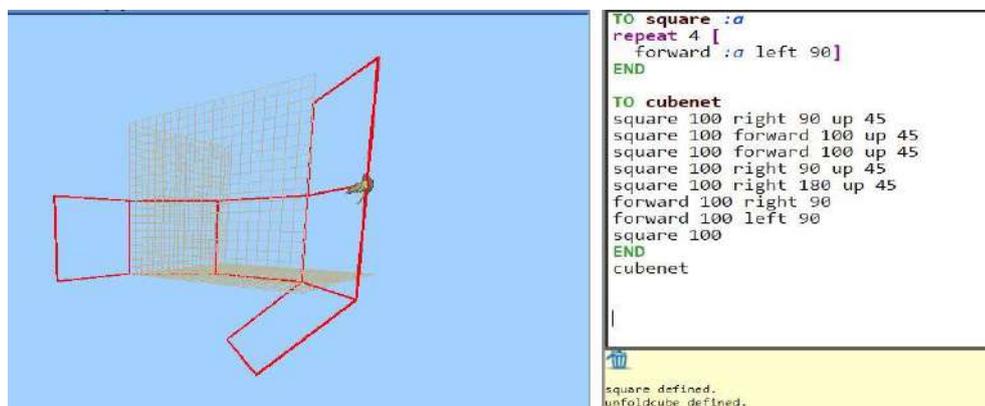


Figure 4. Creating an abstract model of the cube that would be able to animate in space

The two girls try to build an abstract cube model, by creating and comparing different instances of the same cube, like the frames of an animated picture. Then they create an abstract parametric procedure that generates all of these instances and they test it by sliding the slider of the parameter.

In both cases, the final result, which was a parametric procedure that produces any possible form of the cube model between 2 and 3 dimensions, is a product of abstract thinking as students had to create a general model from two static instances (cube, cube net). This process demands to recognize the variable properties of the graphical model, match the properties with the commands in the code and generalize the appropriate commands with the use of variables. In addition, through that process, they realize that in an abstract representation, the outcome should be correct for any possible input. During this process, students made an extended use of the variation tool in order to change the values of the parameters and to cause the animation to their 3D model.

Pattern Recognition

Pattern recognition as a skill of computational thinking is the ability to identify and match similarities (patterns) between different items as a way of gaining extra information. In our study students recognized and used patterns with respect to the objects' properties and behaviours in the 3D space.

After Helen and Alice had finished with their dynamic cube net model, they had some extra time and the teacher asked them to decide what they want to do next.

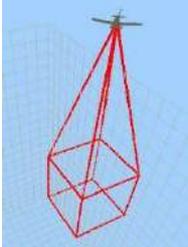
H: "I was thinking to make a pyramid... I tried to create one at home but it was difficult with the angles etc"

A: "We could try to make the net of a pyramid first which is easier and then put a variable to fold it like we did with the cube"

In the above conversation, Alice finds common patterns between the two 3D objects: their dihedral angle that transforms them from 2D to 3D space.

The presence of pattern recognition was even more visible during the third phase when students had to combine different 3D models as building blocks to program a 3D scenery. We will use as an example

the 3D house that was created by group 3 as part of their scenery. For the creation of the procedure *house*, students used as building blocks the procedures “*pyramid :x :w*”, with *:x* being the size of pyramid’s base and *:w* the pyramid’s height, and “*cube :a*”, where *:a* was the size of cube’s edge. First, they programmed a non-parametric procedure for the house in which they used the sub-procedures *pyramid* and *cube* with specific numbers as inputs. However, as they continued with their drawing they decided that a parametric house procedure would be more appropriate for implementing their ideas. Thus, they used the sliders to recognize the common patterns between the parameters of the two sub-procedures and how they could use them in their code. During this experimentation with the sliders, they realized that the cube’s parameter *:a* and the pyramid’s parameter *:w* should be the same variable in the house procedure because the base of the pyramid was attached to the top side of the cube. They also added a variable for the height of the house so that they could create a house of any size in their drawing.

<p>G: <i>Ok lets put some variables to animate the house. Let's write here... pyramid...what is the first parameter of the pyramid procedure?</i></p> <p>C: <i>Move the sliders to see what each parameter does</i></p> <p>George moves the sliders of <i>pyramid</i> procedure</p> <p>C: <i>So, the first one is the base and the second the height</i></p> <p>G <i>Ok.. so... the pyramid's base must be the same with the side of the cube</i></p> <p>C <i>It is cube 70</i></p> <p>George points on the shape with the mouse</p> <p>G <i>This must be variable <i>:a</i> for example and this must also be <i>:a</i>. Do you understand?</i></p> <p>C <i>They are both 70</i></p> <p>G <i>Yes that's what I am saying. To replace the 70 with the same variable</i></p>	<pre> TO house cube 70 fd 70 down 90 pyramid 70 70 END </pre> 
--	--

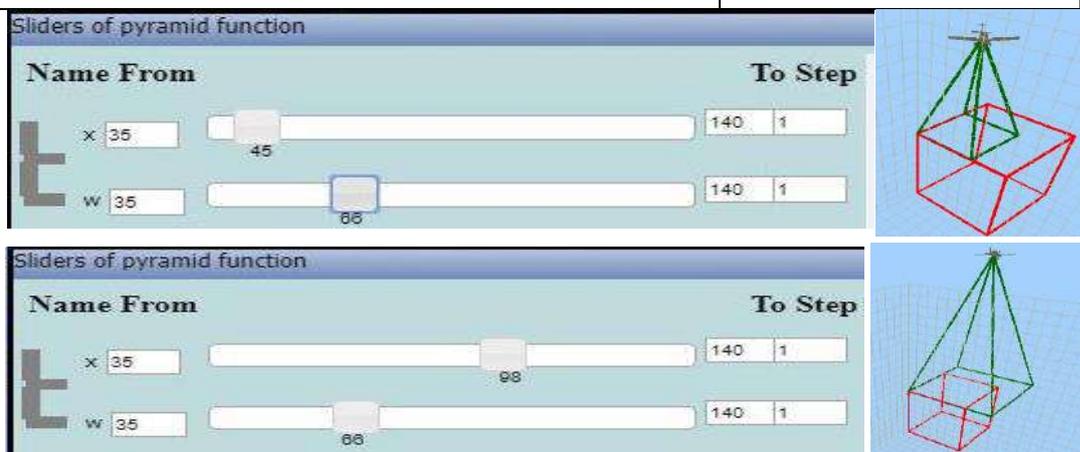


Figure 5. Experimentation with the variation tool for the pyramid function

In the above process, we can see the presence of both abstraction and pattern recognition skills. Students observe the common properties of cube and pyramid models (patterns) and then generalize them to the form of variables in order to create a relation between the two 3D models. What is important in the above episode is that George recognizes the pattern by looking at the drawings in the 3D scene (“the pyramid’s base must be the same with the side of the cube”) while Christina notices the pattern in the code (“They are both 70”). This indicates that the different affordances that MaLT offers can urge

students' abstract thinking by allowing the recognition of patterns simultaneously in the Logo code, the 3D scene and the sliders tool.

The concept of parameter

The combination of Logo programming, 3D graphics and dynamic manipulation tool seemed to urge students generate and discuss meanings about programming concepts. More precisely, students engaged extensively with the concepts of procedural programming and the use of a variable as a procedure's parameter. Two group also used a repetition structure (repeat) and one group implemented a counter by using a local variable. Below we present a characteristic episode that occurred during phase 3 with respect to the use of parameters. At that moment George and Christine from group 3 had already programmed a parametric procedure named "house :width :height" which created a dynamic house by calling the sub-procedures "pyramid :x :y" and "cube :a". At the moment they called the sub-procedures as "pyramid :width :height" and "cube :width". In the following dialogue, students express their ideas about combining the input parameters of the two sub-procedures.

- C: I was thinking to make the height of the pyramid to be proportionate with the height of the cube. Like pyramid :width :height and cube :height*2*
- G: No...I don't think it's a good idea because then we could not do much...we wouldn't be able to create a big variety of houses with the animation*
- C: Oh! What about not having two parameters in the house but only one, the width one? And then call the pyramid with both inputs as :width?*
- G: Or...Maybe the one :width and the other :width/2*
- C: Yes and then the roof would be related to the cube like the double of it width*2 or the half width/2, width/3 etc ... And it would look nice with the sliders' animation*

In this example, we can see the two students expressing ideas around the concept of the variable which is quite important in programming parametrical procedures. They seem to situate the concept of passing variables as inputs between procedures and how in that case variables maintain their numerical value. They also experimented by merging two parameters in one and creating the same procedure with a different number of parameters but with the same output. It is interesting how the dynamic manipulation affordance affects their discussion by providing an imitate animation of all the procedures. In the end, they created four alternates of the procedure "house" which had the exact same code and differ only on the number of their parameters and how they called the sub-procedures. The graphical result on the scene included four animated houses each one with different properties, as they are depicted in the table below.

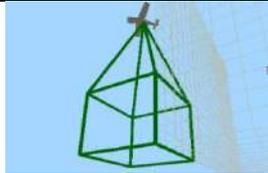
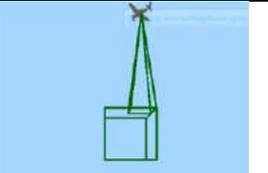
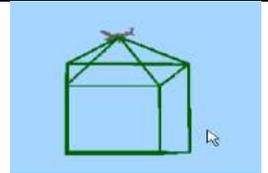
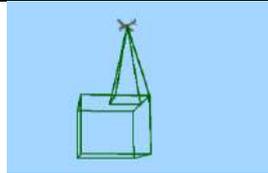
			
2 Variables To house1 :width :height cube :width pyramid :width :height end	2 Variables To house2 :width :height cube :width pyramid :width/2 :height end	1 Variable To house3 :width cube :width pyramid :width : width /2 end	1 Variable To house4 :width cube :width pyramid :width/2 :width end

Figure 6. Different uses of parameters for the creation of four different house models

Discussion and Conclusion

In this paper, we explored an integrated design in which Logo programming is combined with 3D geometry, navigation in space and dynamic manipulation of the parametric models. Our aim was to investigate how this integration of programming and mathematical concepts can be beneficial for the development of important CT skills. In this context, we conducted a 9-hour pilot study with 7 junior high school students who got engaged with programming activities in MaLT digital environment. During the study students designed from simple to more complex dynamic 3D models and drawings using the different affordances of the environment. The analysis of the data showed that the combination of the three affordances of MaLT urged students to apply CT skills in an authentic context and to connect them with different representations. For instance, during the design of the animated cube net, students engaged with the concept of abstraction not only in their code (replacement of constants with variables) but also in the animation of the 3D model with the sliders (by moving any slider the model should animate smoothly between the instances of a cube). Furthermore, the dynamic manipulation feature of the sliders offered a strong tool for code debugging, as it allowed students to experiment with an unlimited number of inputs and immediately connect the algorithm result to the real-time animated model.

However, apart from the use of variables as parameters, there were limited results related to other programming concepts such as iteration or conditionals. In addition, even though students used programming concepts in all of the three activities, the majority of critical episodes related to CT skills emerged during the third activity in which students freely designed their personal 3D drawing. This fact indicated the need for including more open and creative activities to our next study. This is also an interesting outcome for other designs and activities that aim to reinforce students CT skills and reinforces the limitations of closed tasks and quizzes. Our future plans include a large-scale study which will focus on the emerging CT skills from students' engagement in a variety of activities with MaLT. We also plan to include block-based programming in the same environment so that we can make a comparison between the two coding approaches.

We believe that the integration of different affordances can be a strong vehicle for the development of CT but also a new way to combine programming and mathematics in a meaningful context. The results of this study may inform other integrated design, contributing to the ongoing seek of strategies for promoting Computational Thinking for all students.

Acknowledgements

This research has been financially supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 531)

References

- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?. *Acm Inroads*, 2(1), 48-54.
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1–25).
- Clements, D. H., Battista, M. T., Sarama, J., Swaminathan, S. 1997. Development of students' spatial thinking in a unit on geometric motions and area. *The Elementary School Journal*, 98(2), 171-186.
- Cobb, P., Confrey, J., diSessa, A., Lehrer, P., Schauble, L. Design. (2003). experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- diSessa, A. 2001. *Changing Minds: Computers, Learning and Literacy*. USA: MIT Press.
- Goldenberg, E. P., & Cuoco, A. 1998. What is Dynamic Geometry? In R. Lehrer, & D. Chazan (Eds.), *Designing Learning Environments for Developing Understanding of Geometry and Space* (351-368). Mahwah, NJ: Lawrence Erlbaum Associates.

- Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 223–227). ACM.
- Kafai, Y. B., Burke, Q., & Resnick, M. (2014). *Connected code: Why children need to learn programming*. Mit Press.
- Krauss J.& Prottzman K. (2017). *Computational Thinking and coding for every student: The Teacher's Getting-Started Guide*. Corwin Press.
- Kynigos, C. (1995). Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies. In diSessa, A, Hoyles, C. and Noss, R. (Eds) *Computers and Exploratory Learning*, Springer Verlag NATO ASI Series, 399-420.
- Kynigos, C. (2015). Constructionism: Theory of Learning or Theory of Design? In *Selected Regular Lectures from the 12th International Congress on Mathematical Education* Sung Je Cho (Eds) 417- 438, Springer International Publishing Cham Heidelberg New York Dordrecht London, Switzerland 2015.
- Kynigos, C., & Latsi, M. (2007). Turtle's Navigation and Manipulation of Geometrical Figures Constructed by Variable Processes in a 3d Simulated Space, *Informatics in Education*, 6(2), p.1–14 Institute of Mathematics and Informatics, Vilnius.
- Lee I., Martin F., Denner J. et.al. (2011). Computational thinking for youth in practice *Acm Inroads*, 2(1), 32–37..
- Lohman, D.(1988), Spatial abilities as traits, processes and knowledge , in R. J. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol. 4). Hillsdale, NJ: LEA.
- Noss, R., Hoyles, C. (1996). *Windows on Mathematical Meanings*. Netherlands: Kluwer academic Publishers.
- Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Papert, S & Harel, I. (1991). Situating Constructionism. In: Papert, S, Harel, I. (Eds) *Constructionism* Norwood, NJ: Ablex Publishing.
- Psycharis, G. , Kynigos, C. (2009). Normalising geometrical figures: Dynamic manipulation and construction of meanings for ratio and proportion. *Research in Mathematics Education*, 11(2). 149-166
- Resnick, M. Give P's a chance: Projects, peers, passion, play. In: *Constructionism and creativity: Proceedings of the 3rd International Constructionism Conference*. Austrian Computer Society, Vienna. 2014. 13-20
- Stager, G. (2014). This is Our Moment In: *Constructionism and creativity: Proceedings of the 3rd International Constructionism Conference*. Austrian Computer Society, Vienna. 2014.
- Weintrop, D., Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* 199-208. ACM.
- Wing J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3) p. 33-35.
- Wing,J.M. (2008) Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366 (1881) 3717–372

Modeling Time

Kit Martin, kitmartin@u.northwestern.edu

Northwestern University, USA

Gabriella Anton, gabriellaanton3.2020@u.northwestern.edu

Northwestern University, USA

Abstract

We held a two-day workshop where eight 5th graders constructed historical simulations in Unity. We situate the work in the theoretical positions of constructionist history, figured worlds, and narratives. We find that students could use Unity to make figured worlds, discussing their construction and the origins of the sources they used to make the environments. This constructionist learning environment highlighted aesthetics and collaboration.

Keywords

History; constructionism; unity; learning environment

Introduction

Many people are interested in history, it just may not be the history presented in classrooms (Kelly, 2013; Matthews, 2016). Educational researchers argue that what individuals believe about the past is informed by experiences outside the class (Wineburg, 2001). Regardless of their interests, research has shown students can benefit from an investigative inquiry-based approach to studying history (VanSledright, 2002). This approach differs from Lee and Dickinson's (1996) depiction of teaching history in primary school, where the primary focus was on the veracity of facts. Their research brought timely attention to the purpose and quality of history teaching, what students should attain, and in what order, along with realistic expectations of children studying the past. They concluded that the focus on facts can distract from the more important work of validating historical arguments. Common Core Standards in History/Social Studies has advanced history education through integration of standards regarding key ideas, craft and structure, and integration of knowledge and ideas (Common Core, 2010). These advocate that students should progressively analyze, evaluate, and critique the perspective, claims, and reasoning of historical authors. While these standards situate learners as critical consumers of history, they do not encourage learners to produce their own historical texts or arguments (Herrenkohl, and Cornelius, 2013). In short, historical pedagogy traditionally focused on the veracity of facts, but with the move to online fan-based history, and the idea that history is made, not known, pedagogy should shift towards validating arguments and be led by investigative inquiry.

Current trends in New Media studies seek to capture the everyday practices of people engaging with historical content, often focusing on the ways that historical media is understood, critiqued, deconstructed, and reimaged (Matthews, 2016). These studies showcase the ways in which people actively construct personal narratives using historical media and texts, and then share these narratives in online communities. Other scholars studying everyday practices focus on the ways in which people engage in historical content through historically themed video games, like Civilization, Assassin's Creed, or Call of Duty (Squire, 2004). Researchers have shown that players can meaningfully engage with historical content through playing historically situated games (Squire, 2004; Bogost, 2007; Chapman, 2013). Ultimately, both threads suggest that new media content that are historically themed provide a means of creating shared artifacts that allow people to build their own understanding that they can share with others and thereby people together and can craft historical understanding.

In this paper, we extend the argument for engaging with history through new media, suggesting that construction of historical artifacts with new media can be conceptualized as a historical practice situated in the constructionist history tradition we describe below. We present a constructionist design and study of an informal learning environment, in which learners were encouraged to construct historical simulations using the Unity game engine. We argue that a constructionist design facilitates

constructionist history practices, including historical sourcing, the development of mental models or figured worlds, and the creation of personal historical narratives, through the construction of artifacts that externalize mental models and necessitate learners to concretize their thinking.

Constructionist History

In this paper, we present historiography from the constructionist history lens as studied by Wineburg (1998), White (1982), Lowenthal (1992), Davis (1975), and Foucault (1970). Historians, like children working in a constructionist learning environment (Papert, 1980), use mental models to produce intuitions about their subject matter (Schnotz & Bannert, 2003; Schnotz and Kuschner, 2008). White argued that putting yourself in the shoes of agents lets you see the world from their perspective:

The imagination, however, operates on a different level of the historian's consciousness. It is present above all in the effort, peculiar to the modern concept of the historian's task, to enter sympathetically into the minds or consciousnesses of human agents long dead, to empathize with the intentions and motivations of actors impelled by beliefs and values which may differ totally from anything the historian might himself honor in his own life, and to understand, even when he cannot condone, the most bizarre social and cultural practices. (White, 1982, p. 123)

Long dead agents seem to be better understood when one takes their perspective. The process of putting oneself into agents' perspectives motivates constructionist thought. For example, students practice geometry from the perspective of the agent: drawing triangles by directing an agent to take a series of turns (Papert, 1980). In history, this is called "putting oneself in the place of past agents" (White, 1982, p. 123). Natalie Zemon Davis (1975) said in order to defend taking historical agents' perspectives, "We, current historians of popular culture in preindustrial Europe, have a strong streak of interest in the people. But I am not sure we really respect their ways very much; and this makes it hard for us to understand their lives, just as it was for our learned forebears" (p. 266). Cronon posited that "to recover the narratives people tell themselves about the meanings of their lives is to learn a great deal about their past actions and about the way they understand those actions (p. 1369). Baron (2016) put the use of contextualization pragmatically when he argued that through the use of prior knowledge stimulated by historical imagery in concert with new information, college students — and other historians, we would contend — build mental models of historical eras (Baron, 2016, p. 17). Foucault (1970) argued that these narratives are what give order to the past: "History gives place to analogical organic structures, just as Order opened the way to successive identities and differences" (p. 219). That historians and constructionist tinkerers are both engaged in this same perspective-taking is a powerful motivator to build an immersive constructionist historical learning environment.

Frames and Discourse

When children make games, as in Vygotsky's (1978) depiction, they assign new meanings to common objects. For example, behind the couch becomes the robbers' den. The broomstick becomes the cowboy's horse. Through these sign manipulations, meanings of one social setting are mapped onto another imagined place. This remapping is the heart of the notion of figured worlds (Holland et al, 1998). The process also has a dissociative element. People learn to "detach themselves" (Holland et al, 1998, p. 50) from their experienced physical surroundings and enter an imagined play world. In this world, people use collectively developed signs and symbols (Vygotsky, 1978). For example, a prop as simple as a stick might launch a child into a world as a cow wrangler riding horseback. For us, these imagined worlds are similar to the historical practice of stepping into the historical agents' shoes. In our implementation, we can see these symbolic remappings (figure 1) as children remixed digital artifacts to make imaginative use of visual primitives, such as a Malian town, an ancient Chinese palace available for free online, or a simple golf ball model becoming boulders.

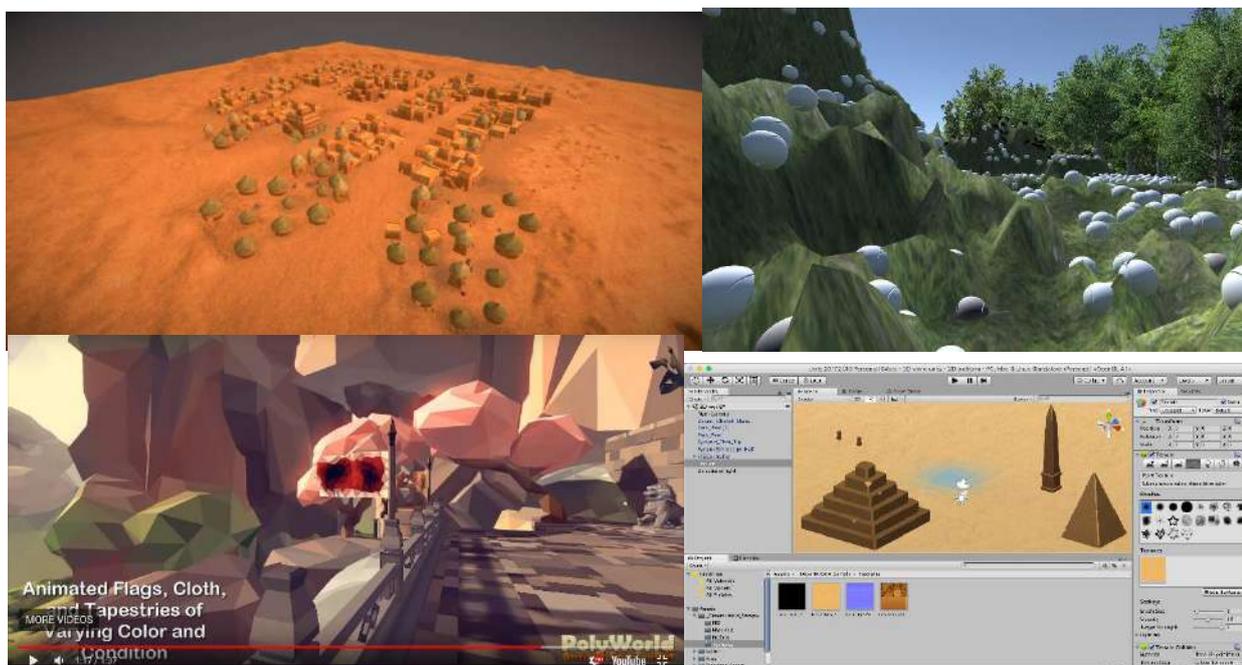


Figure 1. Mali, Ancient China, rolling hills of an imagined landscape decorated with tennis balls as boulders, and Ancient Egyptian monuments were some of the historical primitives students used in their constructs.

Narratives

These figured worlds can be contested. Cronon (1992) shows that narrating the primary sources affects those documents' meaning. Narrative choice matters because the same events can have different meanings. Cronon uses the example of different perspectives on the Dust Bowl. He shows how the way "facts" are presented legitimized and discredits. Historical narrative "sanctions some voices and silences others" (Cronon, 1992, p. 1350). Moreover, the narrative modes people use to order stories have a subtle effect. Particular archetypes of story affect the presentation of history. Cronon presents the two archetypes of improvement and tragedy that were noticeable in depicting the Dust Bowl. Improvement narratives argue that the march of civilization, though ever upward, has some setbacks, but will improve. The tragic type of story argues that society needs to act differently, otherwise calamities like the ones presented will happen again. From the same set of historical sources, historians of the time narrated both types of stories.

These types of stories order genres of depicting the past, such as the long struggle to settle the west that erases the Native American experience and demands continuous improvement. These are ascending and descending narrative voices, and they order not just individual authors, but generations of writers. He points out that the progressive narrative was opposed in the Dust Bowl by New Deal thinkers, who used tragedy in their tale to "construct their stories so as to place themselves on the center stage" (Cronon, 1992 p. 1361). This pitted notions of linear progress against statist constructs of centralized control. Cronon uses these examples to argue for the power of history to "reframe the past so as to include certain events and people, exclude others, and redefine the meaning of the landscape accordingly" (p. 1364).

Like sculptors cutting stone, historians choose which parts to leave in, how to shape them, and how best to present the subject. Though history may not be intrinsic to the universe, it is fundamental to how we humans organize our experience (Cronon, 1992 p. 1368). Cronon gets to the fundamental challenge of postmodernist thought on viewing the past. If our choice of narrative reflects just our power to choose a proffered version of reality on a past event, an event that has no ability to prevent our interpretation, then what is left of history? From this question he poses three axioms that good history must follow — and we see these applying regardless of modality — (1) history does not lie intentionally, (2) history exists within the natural world, and the rules of that system form a natural check on what we

can tell (without obvious fabrication), and, (3) good history exists in a community that others with more knowledge than ourselves can critique our artifacts and therefore there is a community check. As Cronon (1992) said, “We tell stories with each other and *against* each other in order to speak to each other” (p. 1374). Lived worlds then are organized around figured worlds, and figured worlds grant influence to players that they can use in either one (Holland et al, 1998).

Constructionist Digital History Learning Environment

Throughout history, humans have consulted artifacts, such as manuscripts and statues side-by-side with the orally passed concepts to construct history. From Aristotle to Howard Zinn, we have looked to sources of information about the past to develop new ways of viewing our humanity, our present, and our future. This practice’s importance to history is clear from the focus of common core standards on its accurate and proficient implementation.

Ever since we started discovering our ancestors’ cave paintings, we have questioned our past: Who drew it? What does it mean? These questions lead us to even more questions about what happened before us, and we use this information to rhetorically justify our future actions. The battle over who owns history is endless; because we spend so much time with rhetoric, we sometimes forget its powerful sway over our lives. In the rest of this paper, we present a pilot implementation of a constructionist learning environment of history where students can practice the skills of considering, comparing, and validating historical sources. This implementation rests on the idea that Historiography is a contest over what history is, why it matters, and for and through who. It asks, on whose authority? In other words, in this curriculum we posit a design that leverages a fact of history. It is constructed (White, 1982; Cronon 1992; Davis, 1975; Lowenthal, 1998; Foucault, 1970). We argue utility in highlighting the constructed nature of history and in providing venues for learners to construct and share their own narratives of historical events. This classroom design builds on earlier attempts to make history in digital environments (Squire, 2004; Bogust, 2007; De Freitas, 2006; Chapman, 2013; Spring, 2015; Uribe-Jongbloed, Scholz, Espinosa-Medina, 2015). To support this intervention, we will explore how students engaged with the past and how they collaboratively built history. This work acknowledges historians’ traditional apathy for digital history (Chapman, 2013).

Design of the History Simulation Course

From a computer or a textbook, to a pencil and a compass, or a curriculum and a teacher, people use tools to learn. Constructionism flips the traditional way of learning, by looking at how creating the tool, or artifact, helps the builder form a deeper understanding than simply engaging with the learning tool. Papert (1987) coined the name *constructionism* as mnemonic to describe a species of constructivist thought. It focuses on the benefits of learning from the external construction of an artifact beside the internal construction of a mental model or framework. Papert explored its deep connection to computer programming especially how taking a perspective improves learning and facilitates deeper thinking. The field advanced and extended the approach (Papert and Harel, 1991; Wilensky, 1999; Wilensky 2001; Peppler and Kafai, 2007; Blikstein and Wilensky, 2009; Ares, Stroup and Schadmen, 2008; Eisenberg 2007; Worsley and Blikstein, 2013).

The ideas came into the public eye after Papert published his seminal work *Mind Storms: Children, Computers and Powerful Ideas* (1980). Since this time, constructionist principles contributed to many powerful tools for education (i.e., Wilensky, 1999; Resnick et al. 2009). For instance, designers deploy these environments to make mathematics and scientific exploration tools that afford learners the ability to act in sequence or simultaneously on multiple representations of a phenomena (Schwarz and Hershkowitz, 2001; Kaput, 1992). This approach allows for the comparison of different analogies or meanings, which allows students to derive their own order to their knowledge, which is deeper than isolated facts. A motivation for this work is that “Research has shown that many of children’s best learning experiences occur when they are engaged in designing and creating things, especially things that are meaningful to themselves or others around us” (Papert, 1993). In other words, open-ended play, around complicated ideas, restructures the means learners employ to acquire knowledge. Therefore, the flipped approach can aid in knowledge acquisition.

We approached the design of the environment by leveraging the power within constructionist learning environments. We use the pedagogical approach to reimagine history education. More so, we argue that the combination of the two may allow learners to more meaningfully engage with history, as they can virtually construct the figured worlds of history through computer modeling/game design.

In designing the course, we created design principles taken from these two fields. First, we sought to provide learners the opportunity to construct models and games in the context of history with granularity and structure of their choosing. Second, we sought to constrain the experience through the “primitives” that students could access during their exploration. Third, we sought to support flexibility in how learners used primitives (Figure 1). Fourth, we aimed to provide learners an authentic tool that would allow them to create high-fidelity representations of their figured worlds of history.

We selected the Unity Game Development Engine as the software for the course. While there are other game development engines or software available for the construction of new media, Unity provides the opportunity for learners to use community developed 2D and 3D assets, or visual primitives, in their constructions. David Helgason, Nicholas Francis, and Joachim Ante created Unity. Now it is one of the world’s largest creative communities and the number one game development platform as measured by installations worldwide. The environment is free and has all the tools for students and educators to produce high quality simulations fast and move on. Though intended for production, the authoring environment itself is a highly constructionist learning space in which builders can see their vision quickly using prefabricated assets. The ones we used were mostly free and coupled with easy-to-use code.

To construct a historical time period, participants in our intervention chose periods of time they found interesting, sought out images of their period of time using Google Image search, and searched Unity 3D’s online asset store for free 3D assets that aligned with their understanding of the period. Though students’ understanding of the time period is subject to misunderstanding, this plethora of historiographical views is exactly what the Common Core standards advocate are required in a competent historian. Though the intervention was short, in this constructionist project students participated in historical practice: evaluating and incorporating sources to present their understanding of a historical period. We present our immersive historical simulation as one way to create a constructionist history learning environment.

Methodology

The historical modeling course took place over a weekend enrichment program. We collected pre-, mid-, and post-interviews. Additionally, we captured video of the entire two-day intervention. We had students screen capture pictures and videos of times during their construction they found interesting. We collected the games they created, and screen shots of their process. We conducted interviews with students to understand their motivations as they built in paper, in the video game engine, and with each other. Additionally, we coded the data to find moments where students engaged in the core themes including constructionism, collaboration, and use of historical sources.

Day 1

- Intro to History, Games, Modeling
- 2D platformer game
 - Phase 1: building level
 - Phase 2: adding historical images

Day 2

- Introduction to 3D games
- Physical modeling of game environments
- 3D design
- Showcasing Constructions

Results

We will look first at the results our constructionist design approach afforded. Students created playable games that allowed flexible exploration of the concept. They were active in requesting additional information and this allowed students to build on each other's work, but each created with an individual twist. Then we will look specifically at the way students used sources in their design and implementation. The results will show that some students engage historical practices, including evaluating sources, to construct their history simulation in pairs using free resources online, such as Google Images for references of pyramids and free 3D assets.

Constructionism Environment

Learners were successful in creating playable games of varying aesthetic, mechanic, and dynamic feel. The experience with Unity required more scaffolds or constraints than a software explicitly designed for use with novice designers but also allowed for flexible design and provided dynamic feedback. Students could amend their work directly after visual inspection of their creations if they discovered a bug or if something in their design did not look as expected.

The scaffolding of the course provided learners with shared basics and the flexibility to explore components of the game. Seen in Figure 2, L created a platformer with a focus on narrative. She sent a robot back to prehistoric time to thwart evil, time-traveling scientists' attempt to kill the dinosaurs.

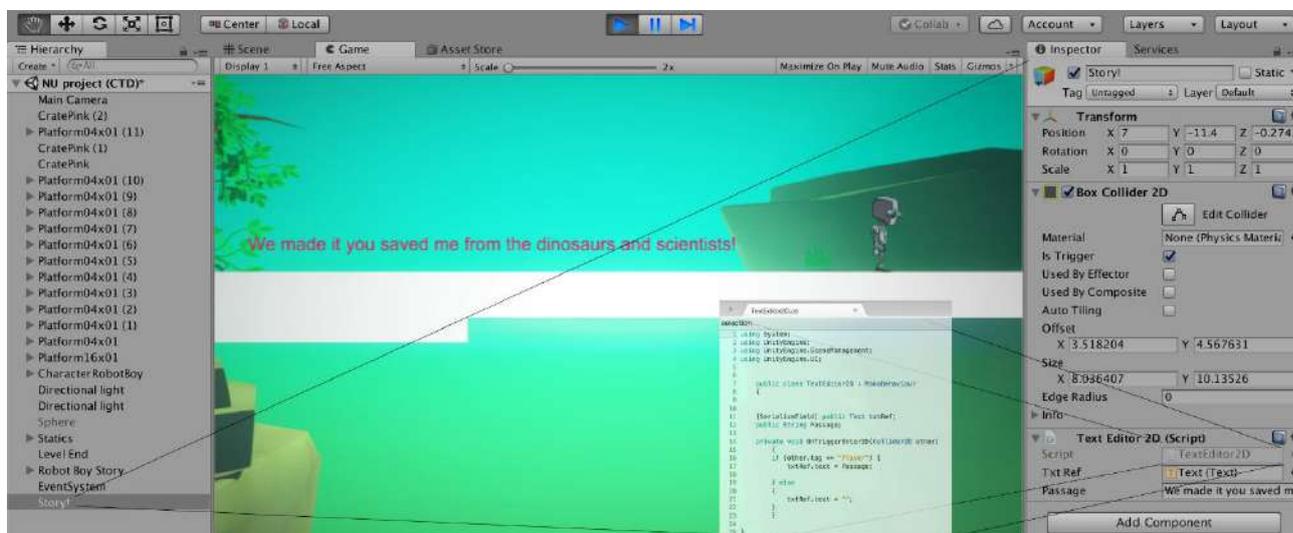


Figure 2. L coded a narrative into her game: a time traveling robot saves the dinosaurs from evil scientists.

This narrative touches on the death of the dinosaurs and added the element of a time-traveling villain. In order to create her time-traveling robot, L sought out support with coding text when the character got to particular key frames in her platform. This is one example where a student wanted to introduce a narrative into the simulation world through code.



Figure 3. Stylized prehistoric era where the appeal of jumps was paramount in design.

Meanwhile, M designed a platformer with a natural aesthetic. In her platformer high-resolution trees rustle in the wind as the time-traveling robot climbs a prehistoric hill (Figure 3). She enjoyed the natural elements in the world she created and spent a lot of time timing jumps so that when the robot moved it was the most visually appealing. Both of these creators largely worked independently after the teacher's initial introduction to the world of Unity game creation. Innovatively, M's use of 3D elements such as the rustling trees into the 2D design created a spectacular bridging between the primitives she could download from the Unity asset store and her own sense of aesthetics.

S focused on the mechanics of the gameplay, working out how to use a 3D character controller in his games to change his character's appearance. He frequently sought out information about how best to approach his tasks, and then diligently worked independently towards his goals. He was a continuous source of inspiration to his classmates as he shared ideas, techniques, and exciting breakthroughs he made in his own Unity world through micro-sharing.

P created a tragic narrative: a world in which humans have exhausted all the natural resources and left behind only now-defunct robots. She used little pink tree primitives from the ancient China package (Figure 1) to build the very last spot of nature in the world (Figure 4).

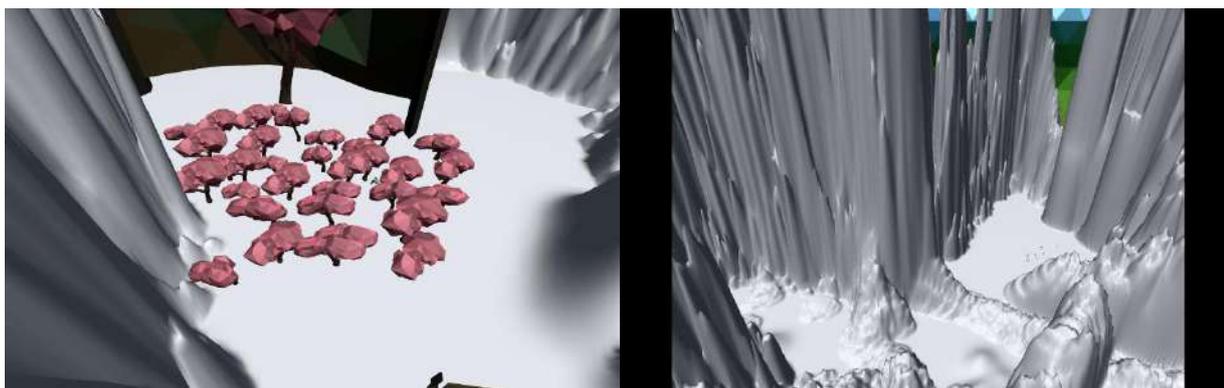


Figure 4. The haunting construction of a post environmental collapse world.

The haunting construction, like the Dust Bowl era historians (Cronon, 1992) evokes a sense that we must take action to save the planet. P shared knowledge with peers who were less confident or didn't know how to approach the problems. This activity allowed her to present herself as an expert that others in the class seemed to consult when they got stuck. This role both made her feel good when she was asked questions, but also made her feel uncomfortable to raise her own questions when she got stuck.

Use of Historical Sources

In constructionism, we are rarely concerned with where the materials come from. We do not ask the origin of straws for building or code for writing. We are interested in the ways the items are used to construct a physical manifestation, an artifact, that felicitously builds a mental model that a builder uses.

Whereas in history, while we are interested in the construct that emerges through historiographic reasoning — traditionally the manuscript — we also care very much about the origins of the materials drawn on to make that manuscript. Historians argue about the historical validity of constructs and the source materials. As a result, we chose to look at the way students engaged with sources separately from whether students engaged with the constructionist design principles of the intervention.

Halfway through the intervention we requested participants to construct their own 3D historical model. During this process some students consulted sources to determine if what they were building appeared like ancient Egypt. In this process we first demonstrated how to make a 3D prototype of a historical period. We gathered around a table with a piece of poster pad paper. To demonstrate the process, the students imagined what existed in the Ice Age: “Huts,” one called. “Ice,” another said. “Saber-toothed cats.” We wrote the ideas in diagrammatic form on the large poster paper without interrogating their notions. Afterward, we asked the students to pick their own time period that they wanted to model and follow this same process to get their ideas down. A1 and J1 chose ancient Egypt.

A1 and J1 constructed 3D versions of ancient Egypt. J1 started the process by placing Post-it notes on his large sheet of paper indicating where to construct 3D temples and pyramids (Figure 5a).



Figure 5. Designing historical simulation on paper referring to Google Images for sources.

Working from his memory of the referenced Google Images sources, he placed them in an arrangement on what he called the desert. Meanwhile, his partner A1, engaged in a high fidelity rendering of one pyramid; he drew each block in a multi-story 2D representation.

The design process took place mostly in silence. Fifteen minutes into the activity, the two of them pulled up images of ancient Egypt on Google Image search (Figure 5). Scrolling through the images, they took conceptual inspiration deciding what to include in their world and in their simulation. Taking turns, they pointed at images on the screen they found relevant to their task and kept scrolling. A1 found the image of the Sphinx particularly interesting. When we asked, “What are you looking at?” A1 responded simply, “Pictures.” When we asked them, “What for?” A1 responded, “Trying to show what ancient Egypt looked like.” When I asked, “How do you know which are real?” J1 flippantly pointed at a cartoon, “That one’s real.” A1 more cautiously responded, “I guess (which one is real).” Continuing “You have to decide” (somehow). This moment is an example of considering what counts as history.

At this point, we asked the participants to start building their prototype in Unity 3D. When J1 asked, “How will I build this in 3D?” the first author responded, “First, build a terrain and then place objects on it.” Oh yeah,“ he responded, as if it was obvious. He then logged into the Unity asset store and started searching for an ancient Egypt model set. When he found one, he celebrated. “Freeeeeeee!” he shouted. He said, I need to search “free”, noting that many assets on the store cost money, but quickly noted he can use keyword search to discover free resources. J1 started his first download. A1 came over, having finished his high fidelity drawing of a pyramid. Appearing to speak to no one in particular J1 said “We are going to make a pyramid.” Then bragging loud enough the whole room could hear “I have such a big pyramid!” demonstrating that he is beginning to feel pride and ownership in his construction.

Right before lunch, with our guidance, J shared the 3D pyramid resource he had found with J1, and then J1 started chanting “cobra” as he added large 3D stone cobras to his simulation. A1 turned his computer toward J1 and started viewing what J1 was making. After consulting the screen, he turned his screen a bit more toward J1 so as to indicate, ‘Look what I’ve made.’ A1 did not lift his eyes from his

screen. J1 said to no one in particular, “Look what this thing gave me,” and rotated his computer toward A1. They now seemed to begin a partnership to build their ancient Egypt (Figure 6).

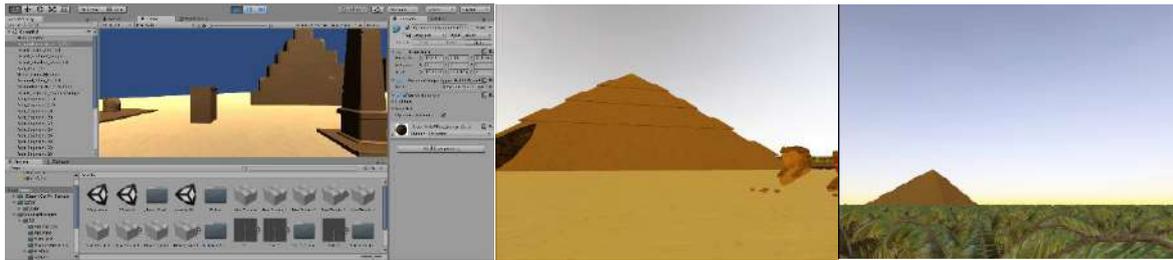


Figure 6. Ancient Egypt as built by two collaborative 5th graders.

Conclusion and Discussion

Within a two-day course, eight students constructed playable video games, engaged in discourse around what to include, and generated two genres and four contexts. While we admit that the constructions in this short course were not accurate representations of the past, we argue that the process students employed to include sources into their environment were accurate historical practices (Matthews, 2016). Students created a historical agent in each of these worlds, and through the view they generated, began to enter sympathetically into the minds of humans long dead. We do not think they began the second part of White’s (1982) activity of beginning to understand, even honor, beliefs and values that differ completely from their own. That being said, the students pragmatically engaged — primed by prior knowledge and historical imagery — with contexts they created which formed mental models of how the past may have worked. We found this work ordered their analogues, which opened the way to the possibility of new successive identities. This process accords with how Foucault (1970) argued history organizes identity.

Through the process of playfully imagining the past, students engaged within this class in a practical approach that could be implemented in another classroom. Though the two authors of this paper learned to use Unity over the past two years, the features we used and had students implement are teachable in a teachers’ training in a few hours. We did see more question-asking by students at the beginning of the course than in a traditional classroom, but by the end of the second day students were mostly independent as they disseminated their technical skills through the classroom through micro-showcasing. This distributed skills sharing, as individual students took on the role of competence, was prevalent.

In the open-ended learning environment our classroom provided, students formed partnerships to build playable games. These partnerships helped some of the students to work from sources to construct their understanding of the past. Often times, these designs were engaged with diligently and mostly silently. This resulted from the way we organized the design process. This process allowed students to engage in laying out their historical understanding, reference sources, and integrate and evaluate multiple sources of information presented in a variety of formats and media to address a question or solve the problem of constructing a past Egyptian pyramid and surrounding environment (as required by Common Core State Standards, English Language Arts Literacy RH.11-12.7).

This work was engaged in both individually and solitarily. The process allowed each student to add an individual twist to their constructs while bringing together the various sources. The process supported friendship formation. For instance, that S sat close to J1 and A1 allowed for easy micro-showcasing. These results came from the open-ended nature of play. Likewise, the flexibility of the design facilitated students to create playable games. This happened in a dynamic feedback environment provided by Unity. We looked at both the historiographic and constructionist implications of this short intervention. We found the environment a useful means of teaching historiographic principles through a constructionist learning environment for three reasons: the level of production, the students’ engagement, and the vitality of the learning environment. We look forward to scaling the approach to a longer course.

The design of the learning environment provided the opportunities to engage in these activities. The class allowed students to make history simulations through the provision of scaffolding constraints to the otherwise very open Unity authoring environment. This was done through an authentic tool, which affords high-end aesthetics and functionality so students could foreground aesthetics to express themselves.

For Vygotsky, fields of human action were games. As a result, social worlds are organized around figured worlds which influence the participants in dialogically contextual constrained world. Different voices, from Google Images, to online assets, to students' own prior understanding of Egypt merge together in these constructs of children in our short course. From the ice-swept slopes of E and L, to the pyramid-spotted deserts of A and J, 5th grade students constructed histories in a professional game design software.

P's games mobilized arguments for a particular past. As Cronon (1992) pointed out, narrating sources affects those sources' meanings. The historical narrative is a sanctioning of some sources' and voices' meanings. Discourse in these games legitimized views of the past by the way the place was laid out. We showed sanction's subtle effect, emerging through people talking and deciding what to include in their past. We noted in P's dystopia a particular ordering of the tools representing Cronon's archetype of tragedy. This narrative structure was similar to the way Cronon (1992) argued political interests marshalled documents to describe the environmental disaster of the Dust Bowl to motivate action. This employment of a tragic archetype motivates viewers to action to save the planet before we environmentally destroy it. We also saw how genres of discourse swept through the authors in our course: two made the Ice Age, two made ancient Egypt, and two made the Stone Age. They came together to agree on what was cool. Like Cronon suggests historians do, our students took the license to include some voices and exclude other voices. This practice is crucial in the study. We find that the main gain of this approach to history-learning arises through discussing constructing the past in a new mode. Though history may not be intrinsic to the universe, it is fundamental to how we humans organize our experience (Cronon, 1992 p. 1368). We saw students organize their understandings of the past through the process, drawing on the sources available to them.

When children played the game of "what if I was traveling to the past," they constructed representations of the past. They assigned new meanings to the digital authoring space available in Unity to turn it into ancient Egypt, the Stone Age, ancient Mali, the future, and Revolutionary War America. They chose to go there in their play. This reassignment is very much like how Vygotsky saw students turn the area behind the couch into the robbers' den. They used collectively developed signs to detach themselves (Holland, 1998) and enter an imagined world of play.

We organized this experience in a constructionist learning environment. We found many affordances to opening up the play. We created an opportunity for students to engage in a process of history-making that aligns with history class standards such as CCSS.ELA-LITERACY.RH.11-12.7. The design afford the opportunity by mixing this environment with the modern understanding of history as a socially constructed artifact. This design built on the promise that video game design may soon change how history is taught and advanced, while keeping an eye on the traditional wariness of history teachers to embrace digital methods.

Acknowledgements

This work emerged from working with Jolie Matthews in her New Media class in 2015 and without her help it could not have developed. Additionally, we would like to thank the Uri Wilensky and the Center for Computer Based Modeling, and the School of Social Policy at Northwestern who provided us many resources in this work. We would also like to applaud Chastity West, and her copious feedback. Finally, we would like to thank US Department of Education, Institute of Education Sciences, Multidisciplinary Program in Education Sciences, Grant Award # R305B140042 for supporting this work.

References

- Ares, N., Stroup, W. M., & Schademan, A. R. (2009). The power of mediating artifacts in group-level development of mathematical discourses. *Cognition and Instruction*, 27(1), 1-24.
- Baron, C. (2016). Using Embedded Visual Coding to Support Contextualization of Historical Texts. *American Educational Research Journal*, 53(3), 516-540.
- Bogost, I. (2007). *Persuasive games: The expressive power of videogames*. Mit Press.
- Chapman, A. (2013). Is Sid Meier's Civilization history?. *Rethinking History*, 17(3), 312-332.
- Common Core State Standards Initiative. (2010). Common Core State Standards for English Language arts & literacy in history/social studies, science, and technical subjects. Retrieved from http://www.corestandards.org/assets/CCSSI_ELA%20Standards.pdf
- Cronon, W. (1992). A place for stories: Nature, history, and narrative. *The Journal of American History*, 78(4), 1347-1376.
- Davis, N. Z. (1975). *Society and culture in early modern France: eight essays*. Stanford University Press.
- Eisenberg, M. (2007, March). Pervasive fabrication: Making construction ubiquitous in education. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on* (pp. 193-198). IEEE.
- Foucault, M. (1970). *The Order of Things... Transl.*
- Harel, I., & Papert, S. (Eds.). (1991). *Constructionism*. Norwood, N.J.: Ablex Publishing.
- Herrenkohl, L. R., & Cornelius, L. (2013). Investigating elementary students' scientific and historical argumentation. *Journal of the Learning Sciences*, 22(3), 413-461.
- Holland, D., Lachicotte, W., Skinner, D., & Cain, C. (1998). Figured worlds. *Identity and agency in cultural worlds*. Cambridge: Harvard University Press.
- Lowenthal, D. (1998). *The heritage crusade and the spoils of history*. Cambridge University Press.
- Kafai, Y. B., Peppler, K. A., & Chiu, G. M. (2007). High tech programmers in low-income communities: Creating a computer culture in a community technology center. In *Communities and Technologies 2007* (pp. 545-563). Springer London.
- Kaput, J. J. (1992). Technology and mathematics education. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 515-556). Reston, VA: National Council of Teachers of Mathematics.
- Kelly, T. M. (2013). *Teaching history in the digital age*. University of Michigan Press.
- Lee, P., Ashby, R., & Dickinson, A. (1996). Progression in children's ideas about history. *BERA DIALOGUES*, 11, 50-81.
- Matthews, J. C. (2016). Historical inquiry in an informal fan community: Online source usage and the TV show The Tudors. *Journal of the Learning Sciences*, 25(1), 4-50.
- Nilsen, A. P. (2016). Navigating windows into past human minds: A framework of shifting selves in historical perspective taking. *Journal of the Learning Sciences*, 25(3), 372-410.
- Schnotz, W., & Bannert, M. (2003). Construction and interference in learning from multiple representation. *Learning and Instruction*, 13(2), 141-156.
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36, 1-11.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S. (1987). *Constructionism: A New Opportunity for Elementary Science Education*.

- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67.
- Schnotz, W., & Kurschner, C. (2008). External and internal representations in the acquisition and use of knowledge: Visualization effects on mental model construction. *Instructional Science*, 36, 175–190.
- Schwarz, B. B., & Hershkowitz, R. (2001). Production and transformation of computer artifacts toward construction of meaning in mathematics. *Mind, Culture, and Activity*, 8(3), 250–267.
- Spring, D. (2015). Gaming history: Computer and video games as historical scholarship. *Rethinking History*, 19(2), 207-221.
- Squire, K. (2004). Replaying history: Learning world history through playing Civilization III. *Indiana University, Indianapolis, IN*.
- Wineburg, S. S. (1991). Historical problem solving: A study of the cognitive processes used in the evaluation of documentary and pictorial evidence. *Journal of Educational Psychology*, 83(1), 73.
- Wineburg, S., Mosborg, S., & Porat, D. (2001). What can Forrest Gump tell us about students' historical understanding?. *Social Education*, 65(1), 55-55.
- White, H. (1982). The politics of historical interpretation: discipline and de-sublimation. *Critical Inquiry*, 9(1), 113-137.
- Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- Wilensky, U. (2001). Modeling nature's emergent patterns with multi-agent languages. In Proceedings of EuroLogo Linz, Austria (pp. 1–6).
- Worsley, M., & Blikstein, P. (2013, April). Towards the development of multimodal action based assessment. In *Proceedings of the third international conference on learning analytics and knowledge* (pp. 94-101). ACM.
- VanSledright, B. (2002). *In search of America's past: Learning to read history in elementary school*. Teachers College Press.
- Vygotsky, L. S. (1978). *Mind in society: The development of higher mental process*.

Ant Adaptation: A Complex Interactive Multitouch Game about Ants Designed for Museums

Kit Martin, *kitmartin@u.northwestern.edu*

Northwestern University, USA

Michael Horn, *michael-horn@northwestern.edu*

Northwestern University, USA

Uri Wilensky, *uri@northwestern.edu*

Northwestern University, USA

Abstract

This paper describes visitor interaction with an interactive tabletop game on the topic of evolutionary adaptations of social insects that we designed in collaboration with a large Midwestern museum. Our exhibit, called *Ant Adaptations*, takes the form of an agent-based modeling game (ABMG) that integrates complex system learning with gameplay. We video recorded 38 groups (114 participants) playing the game and conducted pre- and post-interviews. We propose a methodology, constructivist dialogue mapping (CDM), to monitor the group-level knowledge as it emerged through interaction with the exhibit. CDM is useful for analyzing the short session times typical of museums. Our results show that visitor groups collaborated effectively to discuss and elaborate on their understanding of emergent ant behavior.

Keywords

Ants; complex systems; informal learning; constructionist epistemology; methodologies; tools and technologies; innovative computing education

Introduction

97% of children play video games at some point in their lives (Lenhart et al., 2008). By encouraging open-ended engagement and exploration, games can support learning across a wide variety of topics and contexts, providing a powerful way for learners to construct new knowledge and understanding (Harel & Papert, 1991). Constructionist learning games try to strike a balance between open-ended play and targeted treatment of learning content (Holbert, Weintrop & Wilensky, 2014). Constructionist video games employ traditional game structures infused with constructionist ideals to create a game experience that both encourage exploration and engages desired content (Egenfeldt-Nielsen, 2006). For this research, we develop a means to research this informal education, constructivist dialogue mapping (CDM). In this paper we describe the design and evaluation of a constructionist learning game called *Ant Adaptations* that we developed in collaboration with The Field Museum. The exhibit presents visitors with the opportunity to create an interactive digital ant ecosystem that allows learners to better understand the world of ants by exploring a microworld (Papert, 1980; Edwards, 1995). We then present findings to answer two questions. First, how can interactive, agent-based, complex-system-based tabletop games best be designed for museum settings? Second, how can effect and efficacy of such interactions in museums best be analyzed?

Research within the museum space could provide a fecund location for future educational game interventions that employ a model-based inquiry of complex systems. Our past work with agent-based models of social insects and microworlds has been effectively used for classroom teaching, but less so in informal museum settings. Examples of this classroom use including: *Ant Food Grab*, a yearlong block-based programming curriculum with ants (Martin, Sengupta & Pierson, 2018; Sengupta et al, 2015), and *Beesmart*, a curriculum about the hive-finding behavior of honey bees (Guo & Wilensky, 2014a; Guo & Wilensky, 2014b). These insect Microworlds allowed students to construct their own understandings of science through exploring and adapting models in a classroom setting. Like other complex systems interventions, the students build their understanding of complex systems and

recursion in natural systems (Danish, Peppler, Phelps and Washington, 2011; Blikstein & Wilensky, 2009; Klopfer, 2003; Levy & Wilensky, 2009; Wilensky & Reisman, 2006; Sengupta & Wilensky, 2009; Wagh & Wilensky, 2012; Wilensky, 1997b; Wilensky & Resnick, 1999; Yoon et al., 2013; Wilensky & Novak, 2010). However, there is much less work that leverages the pedagogical value we discovered in these earlier microworld based interventions in museums (Horn et al., 2014; Strohecker, 1995a, 1995b), a dearth our present work addresses.

Although microworlds can be useful for learners to explore about complex systems, developing robust understandings of complex systems can be challenging. Wilensky & Resnick (1999) describe the difficulties people have in “thinking in levels”, exhibiting “levels confusion” and difficulties with distributed control and stochastic processes. Not only do learners have a hard time thinking across multiple levels such as disease of the whole body resulting from microscopic pathogens, but they also tend not to think about phenomena such as the flow of ink dropped into water as the processes of collectives of agents interacting (Chi, Roscoe, Roy & Chase 2012). If the glass of water changing color is explained by the individual parts of ink interacting with H₂O than the process becomes more intuitive. Most people, however, are not familiar with ink particles. Building off earlier work in classrooms (Martin, Pierson and Sengupta 2018), we posit that thinking about ants is a powerful way to address both of these challenges in museum learning. The game scaffolds thinking in levels such as between the colony and the individual ant, and seeing success and failure of the colony as a result of ants interacting through processes. To research this claim, this paper explores how a user made sense of the self-organization of ant colonies in competition.

Theory: studying learning in informal learning through constructivist dialogue mapping

We developed a way to study learning in an informal environment based in on constructivist theory. In constructivist theory, a learner’s mental model drives his or her construction of understanding and internal cognitive structures. This process includes accommodation and assimilation (Piaget, 1952) and maintaining a balance between stability and change, continuity and diversity, and closure and openness (Ackerman, 2001) when exploring the world. For Piaget, children are not just incomplete adults. Their ideas function very well for their current context and as a result, their mind changes through experience. As Ackerman (2001) said, children’s’ conceptual changes are like those of scientists: they happen through “action-in-the-world” (p. 3) to accommodate for experiences, and most likely through a host of internal cognitive infrastructures. “Knowledge is not merely a commodity to be transmitted, encoded, retained, and re-applied, but a personal experience to be constructed” (Ackerman, 2001, p. 7), like sitting with safari ants.

Mental models are built out of theories of how the world works, or in other words, the sum of lessons learned from thinking passed obstacles. D’Mello and Graeser (2012) showed how people learn when they encounter system breakdowns, which cause a cognitive disequilibrium or obstacle. These moments allow for reflection and reordering of their dynamic memory (Schank, 1999). DiSessa and Cobb (2004) argue that from Newton, to Einstein, and Darwin, theories embody generalizations to organize overly abundant data that is subsequently viewed as part of a new theory. In this way diSessa and Cobb (2004) posit theory as a lens, “teaching us how to see” (p. 4). How we see the world, is the crucial bit about these theories since the lenses constructed through experience take actual form. Just as “[t]he world is not just sitting out there waiting to be to be uncovered, but gets progressively shaped and transformed through the child’s, or the scientist’s, personal experience” (Ackerman, 2001), constructionist thought highlights transformation and molding as the work of mental models.

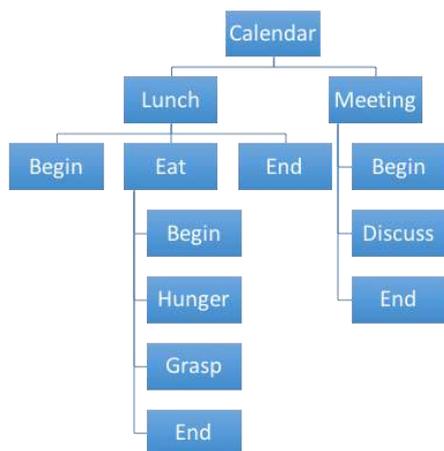


Figure 1. A calendar is an order of processes. Each process has subprocess that are learned through time. So that a meeting has the "begin", "discuss", and "end" subprocesses.

This theory led us to construct a methodological innovation that is highly useful for the short, interactions typical of museums. We develop concept maps (Chi and Koesek, 1983) as they emerge through transcript data. At one level, these maps represent what players said. At this level, we captured ideas about the understanding of science players demonstrate during play in a constructivist map inspired by Minsky's society of mind (1986). For Minsky (figure 1), the human mind works through a cognitive map of agents. In adapting Minsky's frame, we present simplified concept maps of players elaborating their ideas about agents through observation and interaction we call constructivist dialogue maps (CDM). Thus, maps will visually depict the ideas players share through

what they say and how they interact with the game. We posit that we can research what people conserve (that is, learn through accommodation) by filling in a map with what people say and do during play. Constructivist dialogue mapping allows us to track how people's words and actions indicate learning through short play periods.

Ant-Based Modelling

Early work on agent-based modeling was inspired by the behavior of social insects (Resnick & Wilensky, 1991; Wilensky & Resnick, 1993, Langton, 1997). Their behavior has inspired games; SimAnt (McCormick & Wright, 1991), which is based on Hölldobler and Wilson's (1990) *The Ants*. The collective behavior of ants has been simulated using agent-based models (ABM) many times. StarLogo was used to model the collective behavior of social insects (Resnick & Wilensky, 1992, 1993). Wilensky (1997a; 1990) modeled food source preferences resulting from pheromones, as well as the formation of ant trails (Wilensky, 1997b). Bonabeau investigated the role of ABM in pattern formation (1997), and, more broadly, looked at swarm intelligence (Bonabeau, Dorigo & Theraulaz 1999). Prat et al. (2005) modeled collective nest selection of *Temnothorax albipennis* also using an ABM. Sumpter and Pratt's joint work explored the importance of group decision making with quorums (2009). Their work showed that when choosing a destination together, cooperation reduces the probability that an individual will suffer predation. Robinson, Ratnieks, and Holcombe (2008) used an ABM to explore attractive and repellent pheromones in pharaoh ants. Likewise, frameworks, such as Anthill, have been used to support the design, implementation and evaluation of technical systems, such as peer-to-peer networks (Babaoglu, Meling, and Montresor, 2002). Their work drew on examples of complex adaptive systems to justify engineering and user applications because complex adaptive systems exhibit resilience, adaptation, and self-organization that are seen as valuable in social applications. While these earlier models provided insights and enjoyment, none of them delivered their lessons in the short interaction times typical of museums. Taking this previous use of ants as a means to understand, research, and teach complex systems, we designed a complex system model to deliver complexity learning in the short, opened interactions normal for museum educational experiences without the mess of installing 20 million safari ants in the Midwest.

The Game: Ant Adaptation, agent-based modeling in museums

In order to provide context for our analysis below we review the game. In the game we created, *Ant Adaptation*, ants go out to collect food and return to the nest. As they return to the nest ants lay down a pink pheromone that attracts others nearby. Other ants walk toward the strongest chemical smell, which in most cases is where the first ant just arrived. When ants find a flower, their food source, they return,

lay down more pheromones, and thus reinforce the pink trail. This creates an emergent feedback loop that routes more and more ants to successful sites of forage. As the ants exhaust a food source they must find new locations, and thus repeat a cycle. When two or more ants of opposing colonies encounter each other, they fight or scare each other away, also leaving chemicals that attract more ants. Ants either fight and kill each other, or they scare each other away. For the winner, this works to protect the food source from competing colonies. The ant queen reproduces when the ants in her colony collect enough food. The player interacts with this complex system by adding pheromone trails that the ants follow, as well as adding sources of food (i.e., flowers) to the system, thus changing amount of food in the game. Through interacting with the system, students form a functional understanding of the entities and their mechanisms of action (i.e. agents and their rules) in the model.

This setup scaffolds experimentation. Players must simultaneously make choices. Players can touch the screen to add pheromone the ants will follow. Alternatively, at the flick of a switch, they can add more flowers anywhere they like in the game, acting like an ant or a seed, but with a bird's eye view. Lastly, they can choose to apply vinegar, which erases trails. Erasing trails was used by some game players (like Thomas discussed below) to get ants out of a feedback loop that was leading them nowhere. For players with a bird's eye view to achieve their goals in the competitive environment requires they understand the emergent consequences of simple ant behavior. Their finger is both a single agent, and systemic agent.

Furthermore, each player can decide how big its ants are and how aggressive they are. When the size of ants increases, they become slightly faster and stronger in a fight. Each level of increase adds up. At the highest levels they are thirteen times stronger. When players make their ants more aggressive, it increases the radius in which ants detect opposing ants and thus the probability that they will attack. Increasing either the size or the aggressiveness also increases how much food is required to raise an ant, so the largest ant requires thirteen times as much food to feed to adulthood. This gamification impacts how much food ants must collect to make a new baby ant. Increases in either of these parameters reduces the expected population of the colony, though it increases their likelihood of fighting and winning through emergent interactions of parameters and agent actions. This sets up the main action of the game as a series of strategic choices: to decide whether to pacifically collect food, thereby increasing the population, or, to go on the warpath where big, aggressive ants conquer their opponents. Either method of play could lead to high populations or the elimination of the opponent through better controlled food resources. For example, after learning about the consequences of strategic choices through gameplay, players strategize by increasing ants' size, aggressiveness, or both. This might lead them to win the game by annihilating the other group's ant colony. However, bigger and/or more aggressive ants consume more food to reproduce, and potentially reduce the colony's population size. Thus, a player might strategize by adding more flowers and pheromone tracks around the colony to help the larger ants survive. This learning and strategy cycle interweaves the learning into the gameplay.

The game has four affordances that support three learning objectives. In *Ant Adaptation*, playing with parameters allows the player to (1) construct their colony in competition with an opponent, (2) share strategies through comparison, (3) discuss what is happening through observer scaffolding such as parents' intervention, or interaction between players, including taunts, and (4) learn about the emergent impacts of colony behavior arising from individual ant behavior in a complex systems game. This approach allows visitors to learn (1) the impacts of adaptation on ant colony life, (2) how attractants such as pheromones work in ants' organization.

Designing digital multi-touch tables for museums

Prior research on building interactives in museums informed our design. Current research on multi-touch tables for museums suggests several key design elements (Horn et al., 2016; Davis et al., 2015) such as enjoyment, comparability, and productive conflict. Enjoyment, expressed through affect words such as "whoa", "wow", "cool", and "hah," is significantly correlated with learning measures considered by Horn et al. (2016). Facile comparability aided learning in the case of a tree of life game where players who drew comparisons between lineages learned more easily and were more likely to use terms of interest in open-ended questions on post-tests. Block et al. (2015) and Horn et al. (2015) found that dyads spend more time at an exhibit and engage more with scientific content than groups of three or

more. Finally, conflict can be productive. Davis et al. (2015) and Falcão and Price (2011) argue that interference between users on, and across, a multi-touch interface can be productive for learning when it triggers argumentation and collective knowledge construction.

To answer our first research question—how to design for the museum context—from our review of this literature, we hypothesized that to design interactive, agent-based, complex system tabletop games for museum settings that expedite learning in these short interaction times, designers should encourage discussion and comparison in a competitive game mode where the biology and complexity science weave into the experience. As a result, (1) we implemented turns into the play, as Block et al. (2015) found that groups who take turns spent longer times, and engaged more with biological content, in the tree of life exhibit. Taking turns both increased the use of learning terms and comparisons with the biological content. (2) Our design included two teams which allowed players to explore the game's possibilities and compare between strategies. These two design elements facilitated comparison and discussion between teams. Exploration involves moving their bodies and hands across a digital tabletop to engage in a game. This process engages the group at play more than mousing at a keyboard; we guessed that a body in motion, talking out ideas would create a rich discussion and a problem-solving mindset. The game includes hampering the competing colony through players' dexterity, or at times, physically blocking other players' hands to develop another colony's strength following on earlier work on productive conflict (Falcão and Price, 2011). This competition is mediated through the luck of the stochastic system. Trade-offs of adaptations and complex systems thinking are woven into the game, which allows users to explore and learn about complex systems and ants by making strategic choices both in the digital microworld and while standing in the museum. The design encourages talk and comparison to maximize learning about ant behavior, a complex system, in a short (expected 2 minute) interaction at the museum. In this paper, we demonstrate this design's learning gain through a novel methodology. The argument is that highly engaged play with a compelling complex biological systems model taught a person about ants in a very short period of open ended play.

Agent-based modeling game design framework

By combining the classroom work and general ant-based-models, refined through rigorous testing, we designed an ant-based modeling game that effectively builds on prior successful implementations of complex systems models for the context of museums. An ABGM is a game environment that adopts principles of microworlds to 1) actively engage students in an authentic challenge, 2) help them understand a model of agents, and 3) to let them explore through the construction of their own colony and ultimately understand ant colonies. The ABGM fuses a long research line (Egenfeldt-Nielsen, 2006). ABMG's allow players to explore a representation of the life of an ant colony and learn by interacting with the microworld, thus getting to know its limits through playful exploration and experimentation.

Method: constructionist dialogue mappings

In this study, we applied a mixed methods approach to the observations we made during this pilot (Clampet-Lundquist, Edin, Kling & Duncan, 2011). In addition to observing visitors' interactions with the display and taking ethnographic field notes, we conducted a pre-post semi-structured interview protocol with participant groups. The protocol asked about groups' background knowledge about ants, complex system notions, and feelings about the role of adaptations in ant colony life. This protocol provided a wider view of *Ant Adaptation* players in order to make between-player comparisons (Small, 2011) as well as better understand what they learned about the complex system through playing with the

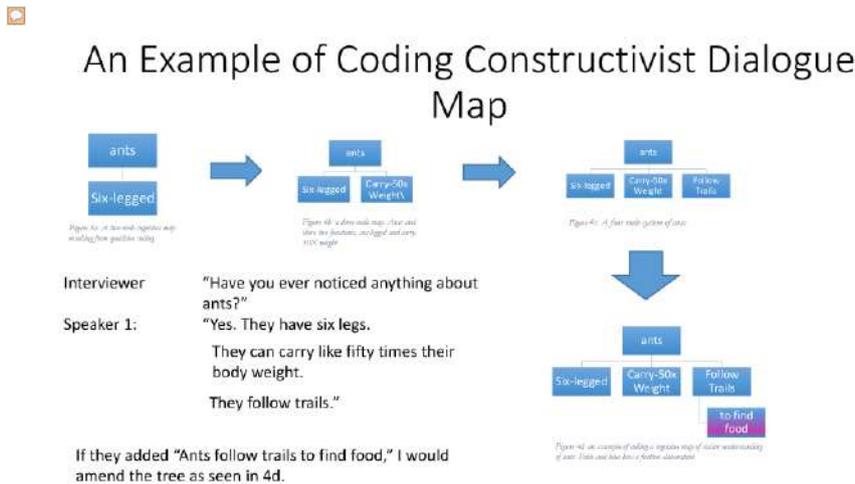


Figure 2. Constructivist dialogue mapping provides a simple interactive way of mapping ontological entities, to their functions as demonstrated by players in short interaction.

maps were constructed by building a hierarchical map of players' utterances (Figure 2) before, during, and after play. Change in ontological maps over time can be compared, and learning shown as the change. The coders proceeded through the transcript utterance by utterance. When an entity was named, such as an ant, the coder added a box. When other entities were named—such as six-legged or carries a lot of weight—they determined what that concept modified. If, as in figure 2a, they modified ant unambiguously, then the coders added subordinated boxes below ant. In our coding, we broke the transcript into a pre-interview, a game play portion, and a post interview. We analyzed the change in how the identified entities receive subordinate boxes. For example, in Figure 2, we would look at how the person changed from describing ants as six-legged (Figure 2a) to six-legged ants that carry 50x their weight and follow trails to find food (Figure 2d). This change over time is our measure of learning during the intervention.

We performed this mapping instead of measuring change through responses to a more classroom-style question about biological facts because people learning in a microworld learn what is allowed in a system, rather than memorizing. "In this picture, the participants are active theorizers. They gather new evidence and devise methods to test their theories. Instead of accepting classifications as given, they see these classifications as provisional theories that are constantly reassessed and reconstructed in light of the dialogue between theory and evidence" (Wilensky & Reisman, 2006, pp 172). We sought to capture how talk changes based on interactions with the learning environment of *Ant Adaptation*. We propose that learning is demonstrated by what students added to their discussions while playing. This elaboration is demonstrated by coding their interactions with a CDM.

Results

We tested the game in a major natural history museum outside of a large, popular, temporary exhibit. The game was used over a six-day period by 114 museum visitors (87% White, 4% Black, 5% Asian, 3% Latino). Of the players, 60 were male (53.57%) and 54 (48.21%) were female. This contrasts with the museum-wide attendance demographics of 70% White (difference of +16.61% points), 5%

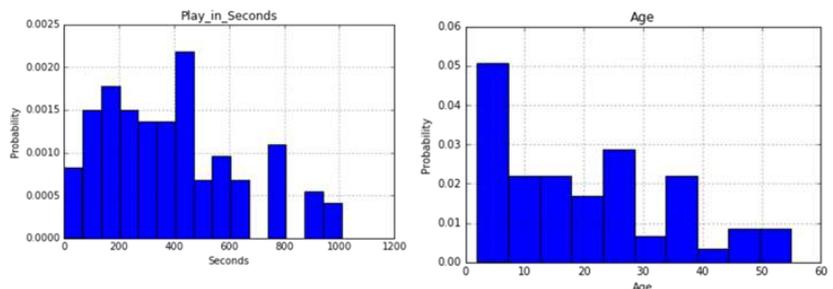


Figure 3. Histograms of play time show the most players engaged for 400 seconds. The average age was 20 with a sizable number of players under 10 and some as old as 55.

interactive tabletop ABMG. From these two sources, we qualitatively coded the types of their interactions (described below), their frequency, and how they elaborated on their understanding of the phenomena (the entities they recognized and their actions) presented during gameplay.

To answer our second research question—how to analyze the effectiveness and efficacy of interactions in learning more about ants in museums—we developed a novel map of ontological entities in the game (i.e., agents). The

Black (difference of -0.54% points) and 14% Latino (difference of -11.32% points). Figure 3 shows players ranged in age between 5 and 55, with the age distribution skewed to lower ages. The average length of time people played was 6:27 minutes as opposed to a museum-wide average interaction time with digital interactives of 1:45 minutes (as reported by internal museum evaluations). The result indicates that the design was a better than average interactive experience in the context of the museum. Unlike Block et al. (2015) and Horn et al. (2015) we found groups of three or more spent more time at an exhibit than dyads. Worryingly, the design seemed to appeal to certain demographics. When we bifurcated playtime by race and gender, however, we find that non-white users who engaged with the game were some of the most engaged users. As shown in figure 4, fourteen of the seventeen non-white users engaged with the game for longer than 4 minutes. In other words, though most players in our sample were white museum patrons, non-white users engaged with the game longer. Because players could stop playing whenever they want, unlike a standardized test, this longer engagement is an interesting proxy for interest. More study is required to understand the implications of the design on audience.

Dialogue elaboration through play

For brevity, we present one demonstrative case of the 38 groups interviewed. Our interrater reliability with two codes was 85%, coding 20% of the transcript data following the CDM method. Looking at how one group engaged with this game for seven and a half minutes, we examine four White youth in the remainder of this section: Thomas, age 7; Ed age 12; Mary, age 9; and Sam, age 6 (names changed for anonymity). The four players were slightly more engaged than the average player of *Ant Adaption*, along several measures: 100% of the group members touched the screen, smiled during play, and worked to maximize their ants' colony population count. This as in contrast to the wider sample where 81% (92 people) touched the screen, 43% (49 people) smiled during play, and 41% (46 people) tried to maximize their colonies' population during play. As there was no guidance on the goal of the interaction, it was surprising to us that so many of the groups chose maximizing their population of ants as their primary goal. In the open ended environment, they could just have easily drawn smily faces with their fingers, or planted a flower garden. We think this goal was so popular because of the competitive arrangement of the exhibit. More study is required to investigate this outcome, which we will do in the analysis of the remaining groups in future work.

Pre-Interview

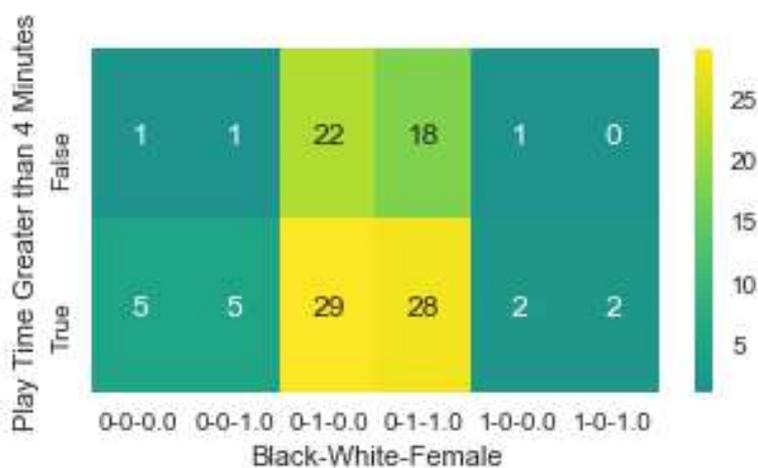


Figure 4. Heat Map of players by race and gender. We see the majority of players, in the middle were white males and females. Notably, of the 17 nonwhite players (0-0-0.0, 0-0-1, 1-0-0.0 and 1-0-1), 14 played for the twice the average engagement time in museum interactives. 1-1-1.0 means non-black, non-white, male.

During the pre-interview, we established a player's prior understanding of ants through a semi-structured interview. As shown in figure 5, during the questions he said ants carry 50x their weight. In response to probes about how ants control traffic, he offered the explanation that ants make physical paths, to control traffic, which he later rescinded after playing the game. Protocol questions on aggressive roles prompted the seven-year-old to guess that being aggressive simply increases an ant's likelihood of getting hurt or even killed. A claim he later amended.

Interviewer: Okay. And then imagine you're an ant. How do you think being aggressive affects your life?

Thomas: Uh, if you're aggressive, like, you- you're ... Uh, how do we explain this? If you're aggressive, like, you- since you go for more things you have a greater risk of getting hurt or killed, I guess.

In response to questions about how ants know what to do, Thomas offered animal instinct, or, a command and control understanding when ants get orders through antenna and/or from their queen.

Interviewer: Okay. That's fair. And then, how do ants know what to do? So you said they- they pick up leaves, or they scavenge, but how do they know to do that?

Thomas: Um, animal instincts.

Interviewer: Animal instincts.

Thomas: Or the queen tells them to, if there's a queen ant. we don't know.

Interviewer: So what's the diff- How does the queen tell them to?

Thomas: It's something with their antennas.

Interviewer: Okay.

Thomas: I think.

Additionally, he said that ants scavenge food from the ground or maybe eat leaves.

Thomas: Like, they can like go hunt for food. They can like, um, try like, get to some, like, maybe some food on the ground like in the city or like in a park, or they can just eat a leaf.

After we explained how the game worked, players broke into two teams, Thomas and Ed versus Mary and Sam. At the beginning, the younger Sam and Mary chose to have maximum-sized, not very aggressive ants (10% aggressive). Thomas and Ed chose to have medium sized though not very aggressive ants (2% aggressive).

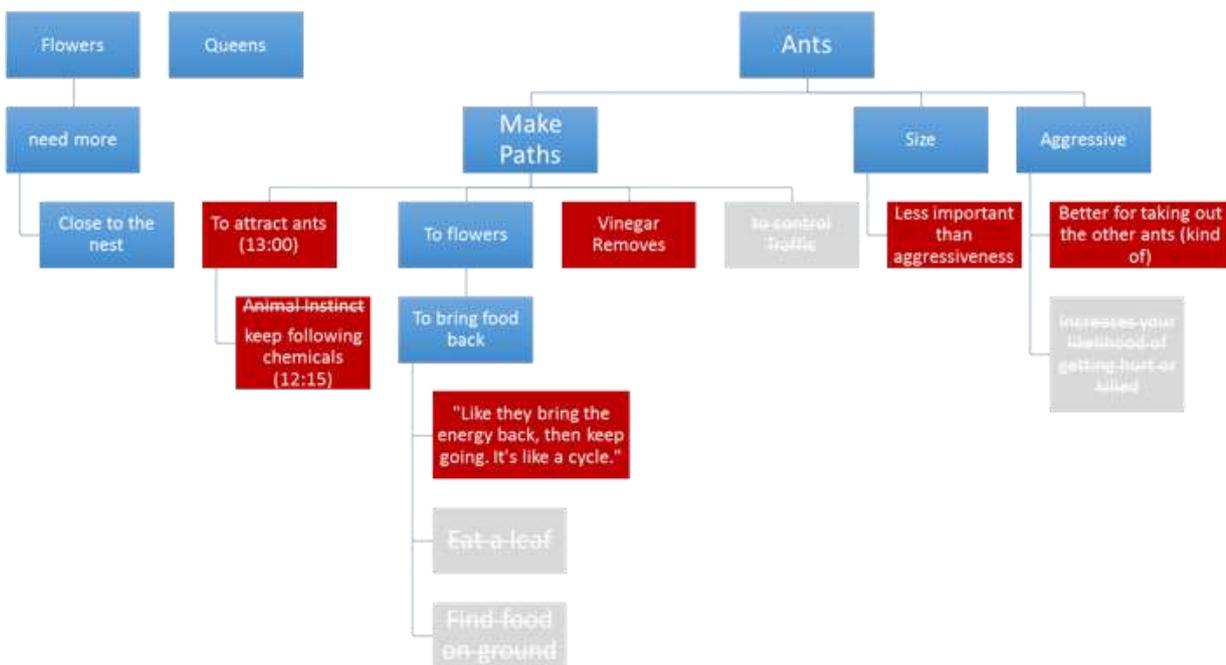


Figure 5. Pre-play cognitive dialogue map of four players' understanding of ants. For instance, they think ants (entity) can carry 50 times their own weight (mechanism).

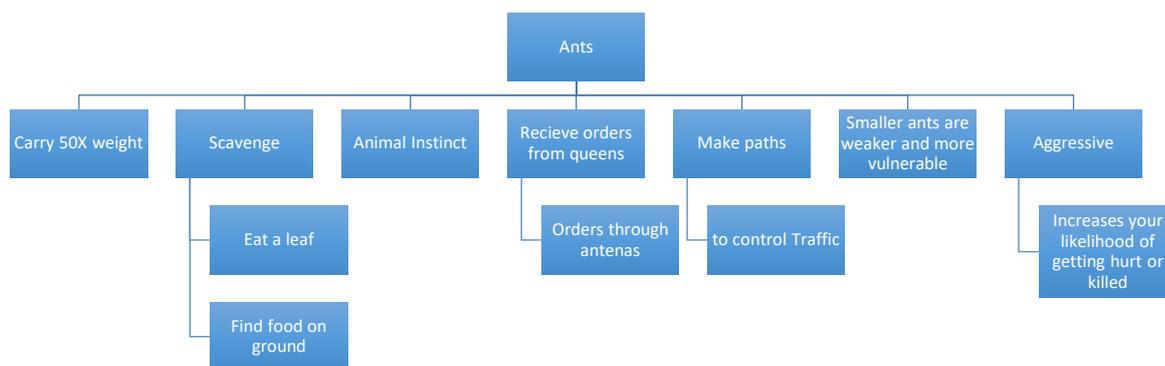


Figure 6. post-play cognitive map of four players. Came to the cyclical understanding of pheromones in food foraging and a more contextual understanding of the role of adaptions utility. Red Boxes indicate elaboration. Grey ideas that were not mentioned again.

Near the beginning, players agreed through conversation on what strategies matter. Soon, Thomas informed the older Ed how to play: “add flowers close to the nest.” From this we add a new ontological entity, flowers, into the group’s constructivist dialogue map and put a mechanism ‘close to the nest’ under it because the players started planting flowers in close proximity. The players then established the connection between collecting flowers and increasing ant population. They offered this as the way to win.

Post- Interview

From the game, Thomas seemed to develop an understanding of the feedback cycles inherent in *Ant Adaptation*:

Thomas: Yeah, you had to figure it out and the- you have to have some flowers, see, and then you put the chemicals and lead it to there, then they’ll bring it back, and like, if you want to get rid of the chemicals you use the vinegar. So, um, you put some sunflowers down, then you get the chemicals and lead it to the sunflowers and if- if there’s too much then the ants aren’t getting the sunflowers and you- then they’ll just like, then you use the vinegar and erase it. But if- if you just do one path that leads to the sunflowers it’ll just get the energy and just keep going back and forth and back and forth. And that’s how we got 21 [ants].

In the post-test players’ concept maps became more elaborate (highlighted in the red boxes in Figure 6). More importantly, he takes on a more cyclical understanding of the role of ants’ paths to attract each other to flowers. Thomas argued that ants follow chemicals to bring food to “get to 21” ants. He also saw that sometimes ants can get trapped in their own chemicals or “white spots.” He used vinegar to clear excess chemicals. Predictions of population level effects emerging from the ants’ simple rules indicated an understanding of intra agent action. The action’s multiscale (Wilensky and Reisman, 2006) effect was learned through play with the game. Thomas leveraged this new understanding to use the macro-level effects of population size to lead his ants to victory by making predictions based in the agent based model’s simple rules. Thomas employed complex systems thinking learned in the short interact to reach his goal of maximizing population. He set this goal in communication with his teammate, in the open-ended constructionist learning environment afforded by our design of *Ant Adaptation*.

Through play the player learned that (entities) ants’ (have a mechanism) lay trails to attract other ants to flowers in a cycle by recursively following the chemicals. The player also realized that sometimes this process can lead ants astray as shown by his use of vinegar to redirect them. He also found that the food source’s (flower entities) proximity to the colony (mechanism) increased the ant population by increasing food intake. Interestingly, after he stated as much, the other side started placing flowers close to their nest.

Conclusion and Discussion

Through the presentation of a single case, from our 38-group sample, we show that a player added to his understanding of ant colonies through the elaboration of discussion about a complex system. He started using vinegar to eliminate local optima where the ants got trapped. Simultaneously, he used the feedback of pheromone trails to organize his colony's foraging and adjust adaptations to changing circumstances.

When working in museums, the short informal interactions with digital interactives changes the relationship between mediating object and the learner. Users can engage with these learning tools how they like. We developed CDM to measure learning in these less structured environments, and from our first test of the method here, it is an efficient and reliable way of tracking what a person says, and how the content of their statements changes during group interactions with a digital interactive. We think this method has potential for measuring informal learning in other spaces, such as maker spaces, and museum exhibits.

The design of the experience drew users in, with people playing for up to a quarter of an hour with average play times over twice the normal interaction times of exhibits in the natural history museum. In light of the success of this pilot, we conclude it reasonable to extend the design principals of *Ant Adaptation* and create complex systems arcade for natural science learning in informal settings. The approach developed here can allow us to identify and describe learning by examining how players reconstruct provisional theories considering dialogue between theory and evidence (Wilensky & Reisman, 2006). Through this approach to teaching natural history in short interactions, we can bring theory building to players, who can, like Newton, Einstein or Darwin, organize abundant data as part of the theory they are building (DiSessa and Cobb, 2004).

The approach to game-play in *Ant Adaptation* scaffolded player theories through the learning objectives. (1) Construction of their own colonies in competition with an opponent afforded comparisons which allowed for dynamic theory validation: Thomas proposed proximity of flowers to the nested aided population growth, the other player also started planting flowers close to the nest. (2) Sharing strategies allowed players to update their operating theory. (3) The social aspect of the multi-touch interactive allowed discussion to guide the theory exploration. (4) Taken together, these scaffolds facilitated players exploration and learning about the complex system. People have argued that learning about complex systems is hard (Chi, Roscoe, Roy & Chase 2012), and for an individual it is. When groups of people each engage part of a complex system, and attempt their best theories in real time, and received dynamic feedback from the computer and each other, the overly abundant data gets fit into a theory that the people test in mediation with the machine and themselves and finds regularities and break in to the system. These learning moments may happen most when they notice breaks where to get out of there confusion, learners must engage in effortful problem-solving activities (D'Mello and Graeser, 2012). That effortful solving activity is the process of science, and that is the process players in *Ant Adaptation* took. While Thomas vocalized most of the learning, the other players tested approaches, and in future work, we hope to better share the joint sense making mediated through technology and each other.

The decision built into the main action of *Ant Adaptation* — whether to peacefully collect food or go to war to eliminate their opponent — sets up a crucial engagement where the uncertainties make the testing immediate and productive (D'Mello and Graeser, 2012). The theory building exercise at the heart of the game was engaged with in the process of this motivated play. The Game afforded that type of play to happen. This work in short expedites the activities found in earlier agent-based modeling and theory building exercises that have been used in schools and with swarm intelligence researchers. The design allowed us to work in a novel context: super-fast interaction times typical in informal learning environments.

The constructivist dialog map (CDM) approach introduced in this paper was able to capture changes in a player's understanding of agents' actions (entities' mechanisms), learning complex ant systems through playful interaction with our agent-based modelling game (ABMG). The use of this design was associated with comparisons and elicited talk, that was associated with learning like our first research question hypothesized. CDM captured the changes as utterances occurred during a short interaction.

By analysing changes in talk pre and post, we found that a player learned about feedback, and employed that learning at multiple levels to maximize an ant population. The elaboration took place by forming and testing theories with the ABMG, *Ant Adaption*. Thus, an agent-based modeling game of ants can be used to learn about social insects in a short intervention, in an informal learning setting. This method of developing a learning environment has the potential to change how we develop natural history museum interventions. The ABGM design approach is a scalable means to bring a microworld's affordances to informal settings and teach targeted content through games that 97% of children play. If future designs in this space are attached to powerful ideas (Papert, 1980) that help children understand their world, agent-based modeling games can create a powerful means for people to learn about the complexity in the world.

Acknowledgements

We would like to thank Robert Grider who wrote NetLogo multitouch for this project. Also, Corrie Moreau and the Field Museum's Ant Lab, that gave amazing feedback and support throughout the design of the exhibit as well as their welcome to and assistance at the Field Museum. We would also like to thank the IEF for their generous support of this work, Multidisciplinary Program in Education Sciences for funding the project (IES: Award # R305B090009).

References

- Ackermann, E. (2001). Piaget's Constructivism, Papert's Constructionism: What's the difference?
- Babaoglu, O., Meling, H., & Montresor, A. (2002). Anthill: A framework for the development of agent-based peer-to-peer systems. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on* (pp. 15-22). IEEE.
- Blikstein, P., & Wilensky (2009). An atom is known by the company it keeps: A constructionist learning environment for materials science using multi-agent simulation. *International Journal of Computers for Mathematical Learning*, 14(1), 81-119.
- Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems* (No. 1). Oxford university press.
- Bonabeau, E. (1997). From classical models of morphogenesis to agent-based models of pattern formation. *Artificial life*, 3(3), 191-211.
- Chi, M. T., Roscoe, R. D., Slotta, J. D., Roy, M., & Chase, C. C. (2012). Misconceived causal explanations for emergent processes. *Cognitive science*, 36(1), 1-61.
- Chi, M. T., & Koeske, R. D. (1983). Network representation of a child's dinosaur knowledge. *Developmental psychology*, 19(1), 29.
- Clampet-Lundquist, S., Edin, K., Kling, J. R., & Duncan, G. J. (2011). Moving teenagers out of high-risk neighborhoods: How girls fare better than boys. *American Journal of Sociology*
- D'Mello, S., & Graesser, A. (2012). Dynamics of affective states during complex learning. *Learning and Instruction*, 22(2), 145-157.
- diSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *Journal of the Learning Sciences*, 13(1), 77-103.
- Danish, J. A., Peppler, K., Phelps, D., & Washington, D. (2011). Life in the hive: Supporting inquiry into complexity within the zone of proximal development. *Journal of science education and technology*, 20(5), 454-467.
- Edwards, L. D. (1995). Microworlds as representations. *Computers and exploratory learning*
- Egenfeldt-Nielsen, S. (2006). Overview of research on the educational use of video games. *Nordic Journal of Digital Literacy*, 1(03), 184-214.

- Guo, Y., & Wilensky, U. (2014a). Beesmart: a microworld for swarming behavior and for learning complex systems concepts. Paper presented at the Constructionism 2014 Conference, Vienna, Austria.
- Guo, Y. and Wilensky, U. (2014b). NetLogo BeeSmart – Hive Finding model. <http://ccl.northwestern.edu/netlogo/models/BeeSmart-HiveFinding>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Guo, Y. & Wilensky, U. (2016). Small bugs, big ideas: Teaching complex systems principles through agent-based models of social insects. In C. Gershenson, T. Froese, J. M. Siqueiros, W. Aguilar, E. J. Izquierdo & H. Sayama (Eds.), *Proceedings of the Artificial Life Conference 2016* (pp. 664-665). Cambridge, MA: The MIT Press
- Horn, M. S., Brady, C., Hjorth, A., Wagh, A., & Wilensky, U. (2014). Frog pond: a codefirst learning environment on evolution and natural selection. In *Proceedings of the 2014 conference on Interaction design and children* (pp. 357-360). ACM.
- Klopfer, E. (2003). Technologies to support the creation of complex systems models—using StarLogo software with students. *Biosystems*, 71(1), 111-122.
- Langton, C. G. (Ed.). (1997). *Artificial life: An overview*. Mit Press.
- Lenhart, A., Kahne, J., Middaugh, E., Macgill, A. R., Evans, C., & Vitak, J. (2008). Teens, Video Games, And Civics: Teens' Gaming Experiences Are Diverse and Include Significant Social Interaction and Civic Engagement. Pew internet & American life project.
- Martin, K., Pierson, A., & Sengupta, P. (2018). *Ant Food Grab: Integrating Block Based Programming and Biology in 8th Grade Science*. NARST 2018 Conference, Atlanta, United States. <http://www.vimapk12.net/-8>
- McCormick, J., & Wright, W. (1991). SimAnt. Orinda, CA: Maxis Inc
- Minsky, M. (1986). The society of mind. *Simon and Shusier, NY*.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Piaget, J. (1952). *The origins of intelligence in children*. International Universities Press, Inc.
- Resnick, M., & Wilensky, U. (1992, June). *Starlogo*. Workshop presented at the Artificial Life III.
- Resnick, M., & Wilensky, U. (1993, April). Beyond the deterministic, centralized mindsets: New thinking for *new sciences*. Paper presented at the annual conference of the American Educational Research Association, Atlanta, GA.
- Robinson, E. J., Ratnieks, F. L., & Holcombe, M. (2008). An agent-based model to investigate the roles of attractive and repellent pheromones in ant decision making during foraging. *Journal of Theoretical Biology*, 255(2), 250-258.
- Schank, R. C. (1999). *Dynamic memory revisited*. Cambridge University Press.
- Small, Mario. 2011. "How to Conduct a Mixed Methods Study: Recent Trends in a Rapidly Growing Literature." *Annual Review of Sociology*, 37: 57-86.
- Sengupta, P. & Wilensky, U. (2009). Learning Electricity with NIELS: Thinking with Electrons and Thinking in Levels. *International Journal of Computing and Mathematical Learning*, 14, 21-50.
- Sengupta, P., Dickes, A., Farris, A. V., Karan, A., Martin, K., & Wright, M. (2015). Programming in K-12 science classrooms. *Communications of the ACM*, 58(11), 33-35.
- Strohecker, C. (1995a). A Model for Museum Outreach Based on Shared Interactive Spaces. In *ICHIM, Multimedia Computing and Museums* (pp. 57-66).
- Strohecker, C. (1995b). Embedded microworlds for a multiuser environment. *Archives & Museum Informatics*, 57-66.

Sumpter, D. J., & Pratt, S. C. (2009). Quorum responses and consensus decision making. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1518), 743-753.

Wagh, A. & Wilensky, U. (2012). Breeding birds to learn about artificial selection: Two birds with one stone? In Proceedings of ICLS 2012, Sydney, Australia.

Weintrop, D., Holbert, N., Wilensky, U., & Horn, M. (2012). Redefining constructionist video games: Marrying constructionism and video game design. In *Proceedings of the Constructionism 2012 Conference*. Athens, Greece.

Wilensky, U. (1997). NetLogo Ants model. <http://ccl.northwestern.edu/netlogo/models/Ants>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach. *Cognition and Instruction*, 24(2), 171–209.

Wilensky, U., & Resnick, M. (1995, April). New thinking for new sciences: Constructionist approaches for *exploring complexity*. Paper presented at the meeting of the American Educational Research Association, San Francisco.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and technology*, 8(1), 3-19.

Wilensky, U. & Novak, M. (2010). Understanding evolution as an emergent process: learning with agent-based models of evolutionary dynamics. In R.S. Taylor & M. Ferrari (Eds.), *Epistemology and Science Education: Understanding the Evolution vs. Intelligent Design Controversy*. New York: Routledge.

Enabling Collaboration and Tinkering: A Version Control System for Block-based Languages

Tilman Michaeli*, tilman.michaeli@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Stefan Seegerer*, stefan.seegerer@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Ralf Romeike, ralf.romeike@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Version control systems are essential for coordinating teamwork when working in projects. They support computational thinking approaches such as collaboration and tinkering. Yet, when using block-based languages, which are an excellent choice for novice programmers, there is no adequate solution that allows this form of collaboration. This paper presents a concept for a simple and easy to use web-based version control system as well as an exemplary implementation for the popular language *Snap!*. This concept is based on an analysis of existing version control systems and their use in Computer Science Education. Furthermore, possible use cases for such a version control system in classroom environments will be outlined.

Keywords

version control system; block-based languages; Snap!; computational thinking; collaboration; tinkering

Introduction

Collaborative learning based on the work of Vygotsky integrates social aspects into constructionistic learning. Much has been written about the advantages of working in groups early on (e.g. Chase & Okie, 2000). Using Projects and project-based-learning (PBL) is one way to enable collaborative learning and a typical method of CSE. PBL is suitable for novices as well as for more advanced learners (Kastl & Romeike, 2015). When carrying out programming projects, one of the recurring challenges is that often arises that different versions of code have to be managed, project groups need to coordinate and merge their code. To work together efficiently, professionals use version control systems. Such systems enable collaboration by allowing teams to work together on the same project by sharing corresponding files. Furthermore, they keep track of revisions and, therefore, make it possible to go back to old versions, to track changes, to fix bugs, or to work in branches, which enables experimenting and tinkering in a sandbox. In a PBL context, such a version control system therefore enables collaboration and tinkering, which are approaches to Computational Thinking (CT) (Barr & Stephenson, 2011). Collaboration is an important aspect of working as a computer scientist. It includes factors such as decomposition of tasks or communication among each other and promotes motivation and commitment. When sharing or discussing their actions, learners can learn from, reflect and build on the work of others (Laurillard, 2009). In CSE, collaboration is considered to be important early on as well as throughout the whole curriculum. For example, the new ACM/CSTA standards for K12 education (Computer Science Teachers Association, 2017) require collaborative work already in Level 1b (Grades 3-5). Tinkering needs a risk-free environment that supports trial and improvement and fosters confidence, creativity and independent learning (Resnick & Rosenbaum, 2013).

The use of a professional version control system in the classroom is generally possible, but it is suitable only for text-based programming languages and comes with a lot of overhead. Even graduate students are often overwhelmed by the sheer complexity of professional tools (Haaranen & Lehtinen, 2015). For a lot of purposes, only a few functionalities like version history, merging and committing are needed.

* These authors contributed equally to this work.

For novice programmers in Java, the integration of SVN and Git in BlueJ aims to reduce this overhead (Fisker, McCall, Kölling, & Quig, 2008).

Block-based languages like *Scratch* or *Snap!* are very popular in lessons with novice programmers for multiple reasons. As an example, they enable students to build creative programs without needing to worry about syntax (Maloney et al., 2004). Collaboration can take place in two dimensions. Currently, block-based languages only support collaboration in a sequential sense, by supporting and emphasizing remixing (Monroy-Hernandez, 2012). However, there are only limited solutions to enable parallel collaboration in the sense of working on one project at the same time. In a day-long workshop on agile project management with 9th to 12th graders, we recently had a team consisting of three programming pairs working in *Snap!*: Every time they wanted to put together their program pieces (e.g. for a prototype), every pair had to export their project and download it. Then they needed to transfer the XML files to one PC, e.g. via memory stick, cloud drive or e-mail. Afterwards, they had to import each project in *Snap!*, manually assemble their scripts, sprites, etc., test and fix bugs on this PC. Hence, the whole team sat in front of one computer to finish the prototype. This is not efficient, the same counts for redistributing the code to all team members. Afterwards, they needed to export the new project status, download it and share it to the other two computers so that everyone was up to date. They also had to manage the versions properly in order to be able to use an old version if necessary. Experience has shown that this often causes problems. As one might expect, the students named these as major downsides to the workflow in the concluding reflection phase.

In order to address this problem, we decided to design a version control system for block-based languages. Therefore, a review of existing professional and didactically adapted version control systems and their use in CSE was carried out with the goal to identify important findings and adapt those for block-based languages. Using an exemplary implementation for the block-based language *Snap!*, the proposed concept is demonstrated and its benefits are highlighted.

Context and Background

Versions control systems

Version Control systems offer a variety of functions important to collaborative practices. First of all, they document changes and their reasons by providing a history for each file under version control. Each change can be described and summarized by the user through comments. Furthermore, version control systems offer the possibility to restore older versions. This way, unwanted or problematic changes can be reverted. Besides that, they enable coordination by offering features to resolve conflicts with multiple users working on the same file simultaneously. These features include locks to prevent multiple users from editing the same file at the same time, the automatic merge of concurrent edits or support for manual merges, if needed.

All files under version control are located in a *repository*. If a user adds new files to the repository and/or changes old ones, they *commit* their changes. Each commit contains *awareness information*, which describe the commit and can be divided into two categories. Internal awareness information includes changes made, time and date of changes, revision numbers, or names of committing users. This information is generated automatically. On the other hand, there is explicit awareness information, which is stated explicitly by the user. It requires explicit action. One example is a *commit message*, in which a user describes the changes he has made (Fisker, McCall, Kölling, & Quig, 2008). If no one else changed any files in between the last commit and the new one, this results in a new version, also known as a *revision*, of the project without any additional action. If someone else has made changes to files in between, but they are not in *conflict* to each other, these changes are automatically *merged*. If there is a conflict, when e.g. the same line of source code was changed by more than one person, it must be solved manually by choosing for each conflict the version which should be in the new revision. Changes between revisions (added through commits) can be viewed via *diffs*. Old revisions can be viewed or *reverted* to at any time. Furthermore, modern version control systems offer the possibility to *branch*. A branch is an alternative path starting from a certain revision, so that changes can be made in parallel. Branches are used for development of new features or experimenting, without impacting the current

product and state of the project. A branch can be merged into the master/production branch again later on, e.g. if the new feature is fully implemented and tested.

Version control systems can be divided into *centralized* systems (like CSV or SVN) and *distributed* systems (like Git or Mercurial). In centralized systems, the repository is kept on a remote server everyone has access to. Whenever a user wants to introduce changes, they retrieve the latest version from the server first. As commits are transmitted to the remote server immediately, any recent changes must be merged, and conflicts have to be resolved before the commit. In contrary, distributed systems store the whole project history locally on every computer but also on a remote server. Therefore, each commit will initially be registered locally. To make changes by a commit available for other people as well, they have to be *pushed* to a server. From there, they can be *pulled* by each collaborator to be available locally. This way, merging and conflict resolution are not necessary for commits, but when interacting with the server (pushing and pulling). Distributed version control systems are dominant nowadays. Their main advantages are the local “sandboxes” which enable local changes, reverts etc. for every user offline, the easy branching and merging, and the independence from just one location where everything is stored (Somasundaram, 2013).

Version control in the classroom

Version control systems are used both at school and university level. At universities, control version systems are used frequently. Typical use cases include the provision of course materials or the submission of homework. Throughout literature, advantages of the use of control version systems can be divided in organizational and pedagogical ones. Organizational ones are the easy way to post assignments and give feedback, the possibility to start with skeletons, revert changes and work remotely, as well as having timestamps for submissions (Lawrance, Jung, & Wiseman, 2013). The pedagogical advantages include easier collaboration, the possibility to assess individual contribution, making the development process visible for the teacher and data security in the sense of a backup (cf. Reid & Wilson, 2005, Lawrance, Jung & Wiseman, 2013, Glassy, 2006). Overall, it is reported that version control systems are considered useful by students and teachers alike (e.g. (Isomöttönen & Cochez, 2014)). Just like the advantages, further experience and especially problems in the use of control version systems are reported. These hint at obstacles that must be addressed in a pedagogical version control system. One reported problem is a non-iterative workflow with long periods without a commit. This is an obstacle especially at the beginning and takes a lot of the advantages away (Glassy, 2006). Overall, professional version control systems are reported as hard to learn (cf. Isomöttönen & Cochez, 2014, Haaranen & Lehtinen, 2015). Students sometimes damage repositories so that tutors need to repair them, or misuse features, e.g. repeated checkouts instead of updates in SVN (Reid & Wilson, 2005). In some cases students even accessed third-party repositories (Reid & Wilson, 2005). From a student's point of view, conflicts and their resolution are the most complex and difficult tasks (Isomöttönen & Cochez, 2014). If the students always work in the centralized repository they have more problems than when they work in their own branches and merge when finishing a subtask (Lawrance, Jung, & Wiseman, 2013). In addition to students, teachers also need significant competencies to use version control systems successfully in the classroom. They need to set them up and configure them, and also support the students, e.g. when fixing broken repositories.

Brichzin and Rau (2015) give an overview of typical problems that can be addressed by the use of a version control system in a school context that matches our experiences. One problem is the pupils name convention and versioning practice – e.g. filenames like *game*, *game_2*, *game (copy)*, *game (working)*. This is an obstacle for collaboration as well as identifying the current state of the project to work on after holidays or a longer break. If the students make no backups of the current or former status of the project, there is always the danger of deleting the work of up to several weeks by accident. The next problem they mention is merging partial programs in PBL together regularly, no matter whether it is an agile project with iterations or a traditional waterfall project. Using a version control system facilitates regular merging and therefore helps to identify interface problems at an early stage of the project and address them accordingly. Another typical problem within the school context is that an entire team gets blocked because a student has forgotten the current code at home or they lack access to his account while the student is sick. Furthermore, enthusiastic students can't continue to work on the project at home, because the code is stored on the schools' machines.

However, the introduction of a professional version control system is associated with a large overhead. Pupils must develop an understanding of the functionality of version control systems, familiarize themselves with the respective commands and working procedures. The complexity of this task poses problems even for entry-level professionals and graduate students. Therefore, Fisker et al. (2008) enabled group work support in the form of a simplified SVN and Git integration for the IDE BlueJ. One design principle for this was making awareness information available. This is important for group work and to keep track of others' progress. Another explicit focus was simplicity by reducing overhead: files no longer need to be added to version control manually. Furthermore, many of the powerful but not essential features (such as branching, tagging, revert, single file functions) of SVN resp. Git are not available via the BlueJ IDE to ensure easy access. The same goes for the graphical user interface, which is kept basic and simple. Functionalities such as *commit* or *update* are made clearer but still contain the standard terminology (e.g. "Update from Repository")

Block based programming

Traditional text-based programming languages have been used for introductory programming or computer science courses but are considered to be major entry barriers. Block based languages take away users' responsibility to take care of precise syntax compliance. They allow for easier realization of creative projects and give direct feedback by visualizing the current program's state. Known examples of block-based languages are *Scratch*, *Snap!*, or *GP*. Most block-based languages are capable of running in the browser. Hence, they do not require an installation on student devices, making them a reasonable choice for educators. Those languages are used in schools and university courses, especially for novices. Using block-based languages with a traditional version control system is unsuitable: Traditional version control systems are built to manage multiple source text files, whereas in block-based programming environments, students interact with their project in a graphical way. In *Snap!*, objects are represented as so called "sprites". Due to this, each object is represented graphically. Sprites have scripts, which are a sequence of several connected blocks (see Figure 1).

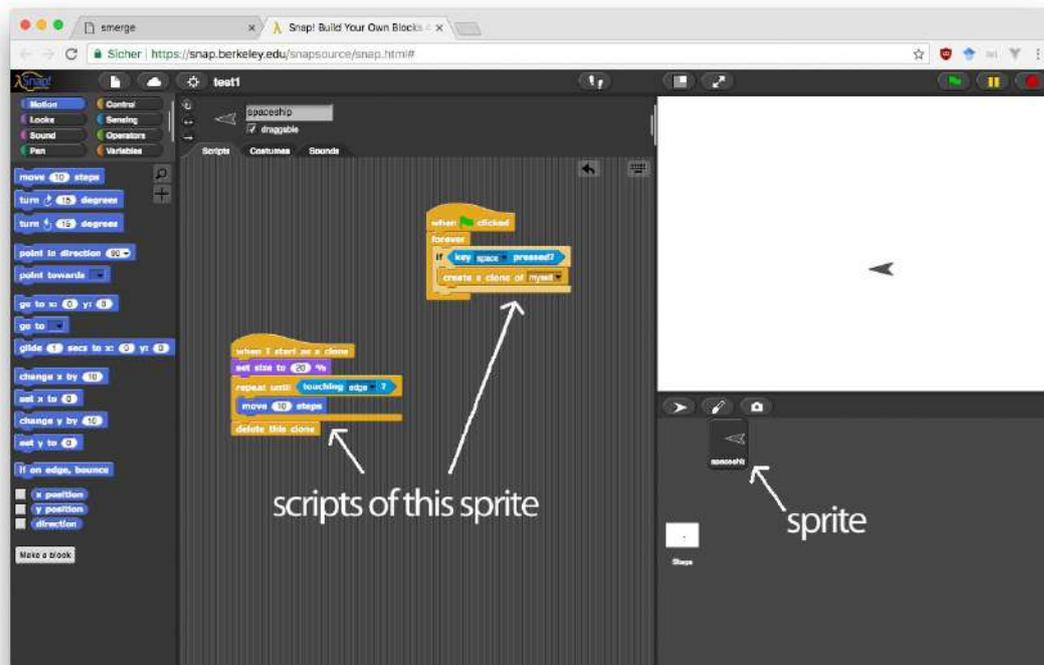


Figure 1. Snap! user interface

Certain block-based applications such as *Kanto*, *Blockly*, or *Netsblox* already allow multiple students to remotely collaborate on a project (Broll et al, 2017, Ohshima, Freudenberg & Amelang, 2017). However, they lack essential features that version control systems offer, such as version history, branches, or commits, which are considered essential in PBL settings.

A version control system for block-based languages

Since existing tools of version control systems cannot be used for block-based languages, we have developed a solution based on research regarding the use of version control systems in the classroom. Therefore, we conducted a didactic transposition of professional version control systems with explicit attention to specific characteristics of block-based languages and needs of programming novices. It follows two guiding ideas:

- *Visualization*. The project status should always be visible at a glance. For this purpose, it should be displayed in a graphical way.
- *Easy to use*. The number of functionalities should be reduced, and unnecessary settings removed. The usage should be simple for both students and teachers. The latter usually having little experience with professional software development tools necessitate this guideline even more.

The second objective is consistent with the goals for version control realization in BlueJ (Fisker, McCall, Kölling, & Quig, 2008). In contrast to their solution, however, the operation is further reduced, and more simplifications are offered even for teachers. In the following, we describe the concept of the version control system and use images of our concrete implementation for *Snap!* to illustrate the concept.

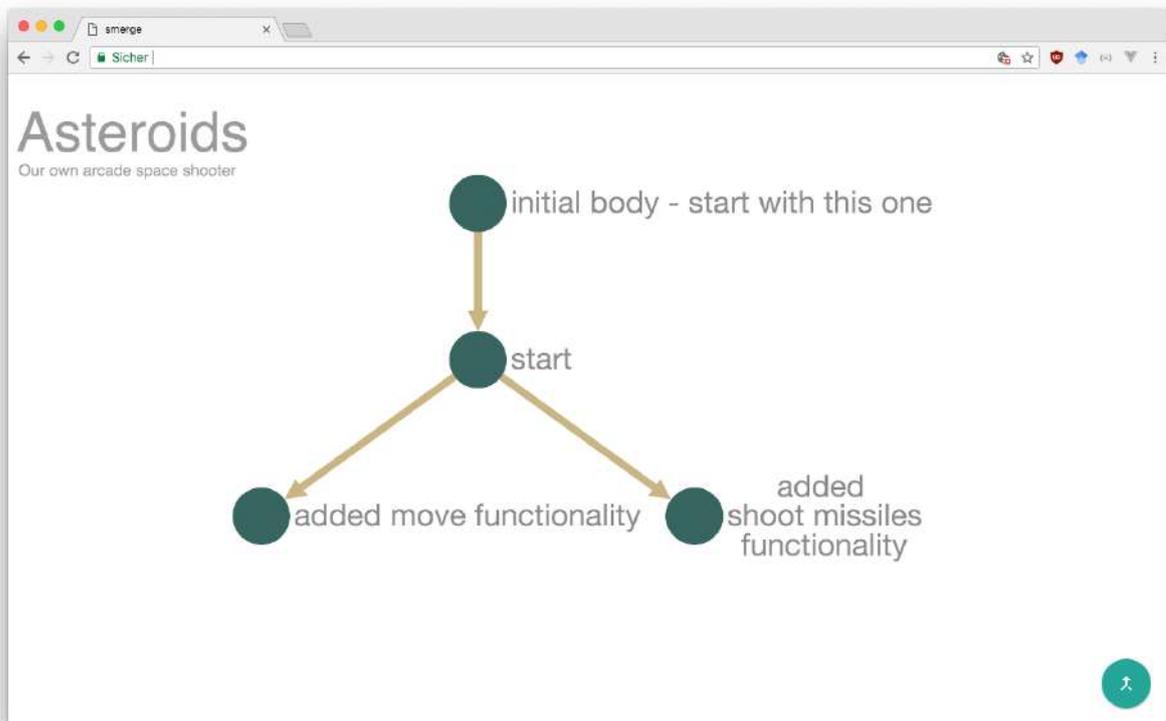


Figure 2. Graphical visualization of a project

Conception

A version control system for block-based languages needs to be web-based. A project is represented by a graph similar to a *Git tree*. A node corresponds to a revision of version control systems (see Figure

2). It can be classified as a special case of a centralized version control system: there is only one project status stored centrally on a server. However, each user always works in their own branch, which is the norm for distributed version control systems. That means, if a user starts editing a revision, they implicitly start their own branch. If two or more users start working on the same revision, one individual branch per user is automatically created. Therefore, changes to the same revision cannot lead directly to a conflict. This also means that there is no explicit master branch. This addresses the experiences described by Lawrance that students had fewer problems when using their own branch for each sub-task (Lawrance, Jung, & Wiseman, 2013).

To create a project, users can either use an empty project or upload their own templates (e.g. with predefined blocks or sprites). It is also possible to upload additional files later on. As the version control system runs in the browser, there is no need to set up an individual server to use the version control system or configure existing services. This allows the teacher to use version control systems without specific knowledge in a very flexible way.

Double-clicking a node in the graph opens the respective revision directly in the corresponding programming system (e.g. *Snap!*). In doing so, an additional button is inserted into the menu of the programming system. Clicking this button commits every change made by the user directly. By enabling a commit with only one command directly from the user interface, the described problem of few commits during long work periods is counteracted. The user is prompted to enter a commit message. In this way, students are motivated to briefly summarize their changes. Additional implicit awareness information such as timestamps, number of sprites or scripts added and removed, and total number of sprites or scripts are provided for each node. These make it easier for other group members to track changes in the project. Reverting to an old revision is done simply by opening the specific node and beginning to work from there.

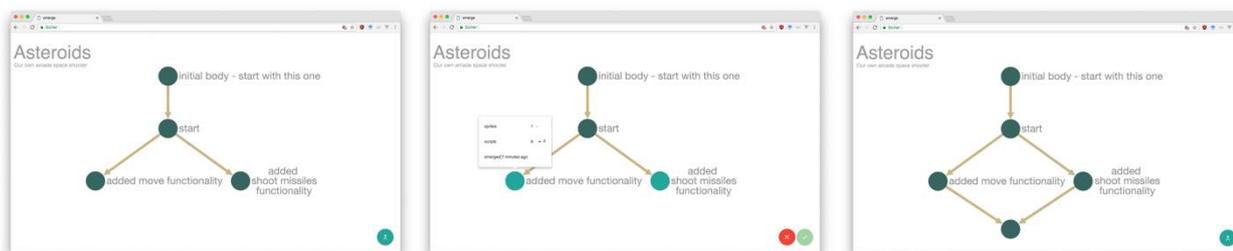


Figure 3. Merge process

The process of merging multiple revisions is initiated by selecting several nodes. If the selection is confirmed, the merge takes place. As long as there is no conflict, the system will merge automatically, similar to a professional version control system. This is the case if neither sprites nor scripts have been changed or only one user has made changes. The more recent version is identified by the ancestor relationship in the graph. Therefore, no interaction on the student side is necessary, unless several students have edited the same script. If this is the case, there will be a conflict. To resolve this conflict, we use a merge view providing all alternatives of the conflicting scripts side by side with comments attached. The students can then select the appropriate version or compose a suitable solution. For an example, see Figure 4: Bob and Susi edited the same script. In addition, Bob added another script. The new script will be merged automatically, while the existing script they both edited will raise a merge conflict. Therefore, both scripts are added to the merged version providing commentary details.

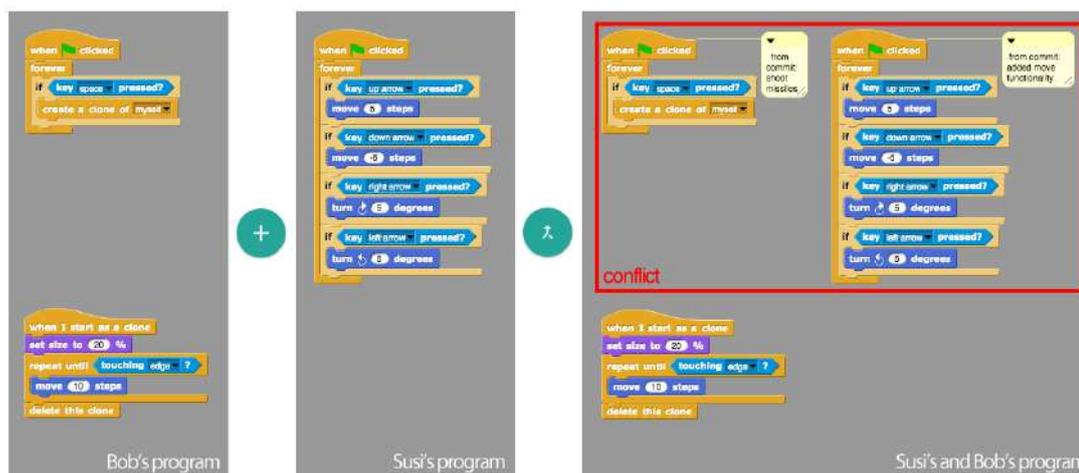


Figure 4. Merge view

Features such as add, push or status, which are known from professional VCS, are not necessary, because the structure of the version control system and block-based languages make these features obsolete. The only activities that students must actively do and learn are commit and merge.

The user guidance and interface are deliberately kept simple. The same applies to terminology, which must be tailored to the target group. Only two essential features need to be named. In discussions with computer science teachers, the use of the term *commit* was rated as difficult. The term merge, on the other hand, was considered suitable for students of all grades. Accordingly, the original term was used in this case, whereas the term *post to <<project_name>>* was introduced for *commit*. This term provides a suitable analogy for commit, comes from students' daily life and is appropriate for all ages.

In summary, key features provided are:

- visualization of the project and its history in a graph
- automatic branching for each editor of a revision
- opening each revision directly in the respective programming system in the browser
- easy commit from within the programming system used
- merging by selecting the respective nodes
- visualizing conflicts in a merge view
- providing implicit and explicit awareness information for each revision
- support for multiple templates and starting nodes

Exemplary Implementation: smerge

With “smerge” (derived from the terms “Snap!” and “merge”), we provide an exemplary implementation of the described concept. The tool is implemented in Python 3 and JavaScript using the Django framework⁴⁹ and cytoscape.js⁵⁰. For running a separate instance, only a server running Apache, Nginx, or similar is needed⁵¹. Instead of handling plain source code as with traditional version control systems, block based languages require a different approach due to their structure. Therefore, we utilize the XML file structure of *Snap!* projects, which differs from languages like *Scratch* or *GP*. On opening a certain revision, the associated XML project file is passed to a *Snap!* instance. In this step, a custom block containing the commit functionality (written in JavaScript) is injected. On commit, the current state of

⁴⁹ <https://www.djangoproject.com/>

⁵⁰ <http://js.cytoscape.org/>

⁵¹ A running instance can be found at smerge.org, the source at github.com/manzanillo/smerge.

the project is exported in XML format and sent to our servers. As soon as the user triggers a merge, the corresponding project files are analyzed and compared on XML level. While conflict detection is easy when comparing source code in text form (usually line by line), once more a new solution for block-based languages is needed. Our solution regarding this conflict detection problem is based on sprite names and script coordinates. For conflict resolution, revisions and their ancestors are compared on XML level according to the auto-merge concept described above.

(How to) smerge in the classroom

In the following, we will describe a possible workflow in smerge when using it in PBL. One way to implement PBL in the classroom is agile projects, which have already been used for PBL successfully (Kastl & Romeike, 2015). In doing so, agile practices such as user stories, standup meetings, pair programming, sprints or prototypes are adapted for the use in schools (Romeike & Göttel, 2012). We will use this framework to describe an exemplary workflow for smerge in PBL (see Figure 5).

For constructionist learning in school projects, students first create their own initial draft. Therefore, each group creates their own smerge project. In this way, they have created a place where all changes to the code are stored centrally. Each programming pair continuously works on one user story at a time. With the auto branch feature of smerge, every user story or feature is realized in its own branch. Pupils are therefore encouraged to work on tasks in parallel and can focus on a single feature each. Each pair has its own sandbox in which they can experiment and tinker.

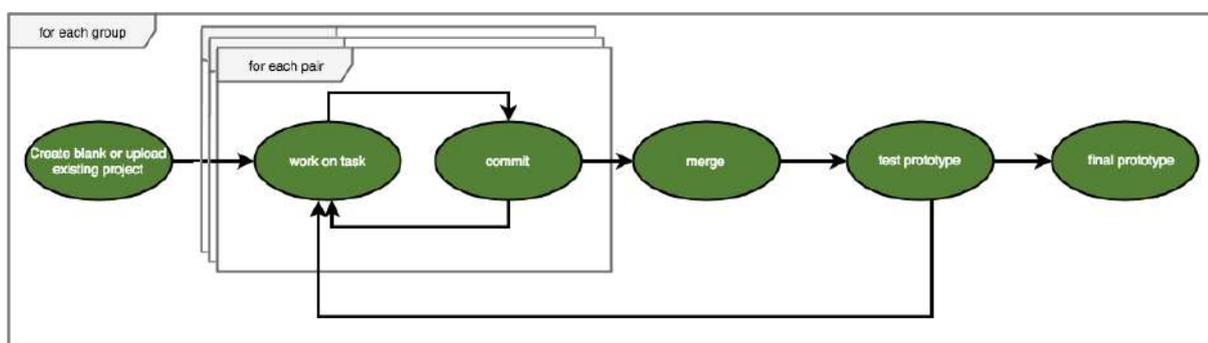


Figure 5. Smerge workflow

This type of workflow also promotes a more realistic form of collaboration in projects. At the end of a sprint, at the latest, the students will assemble their subprograms into a new prototype by using smerge to merge their branches. With smerge, every programming pair can then test the resulting prototype on its own as opposed to the described problem of all group members in front of a single pc. The merge is a ritualized team activity. A project also involves constantly improving the inner structure of the program through continuous refactoring. Smerge supports this with its merge view. By contrasting the individual parts from different sources with each other in the merge view, smerge motivates students to think about possible refactoring. For example, in order to make their own code easier to read, students tend to outsource redundant parts of the code into custom blocks. During the process, the visual representation of smerge, in addition to a possible project board, helps teachers and students keep track of the progress of the project.

The version control system can also be used highly flexibly in teaching outside of PBL. In the following, we will describe a lesson to introduce broadcasting in lower secondary education. In this scenario, every student will create his/her own individual instance of a prototype given by the teacher. Let's assume we want to make a group of penguins dance to a given beat. The beat is determined in a sprite created by the teacher and is delivered to the students via broadcasts. The students' task is to create their own penguin and let it react to the different bars of the music. For this purpose, the teacher provides a template with a simple penguin sprite. The students then rename their own sprite, design its looks and implement an individual behavior on the rhythm. After the students have finished their task, all individual solutions are to be combined into a complete work to emphasize the concept of broadcasts with more

receivers. Without such a tool, it would require teachers to collect all students' solutions and combine them manually. In smerge, this combination is reduced to nothing more than the click of a button.

In addition, smerge can be used for a longer teaching sequence. In doing so, students develop multiple small programs to learn specific CT concepts. The class, or each student, can collect all these sub-projects within one smerge project. Every lesson, the students receive a new template in which they complete a specific task. After several units, the subprojects are combined to form an overall project, and a greater coherence becomes apparent. One example is the game Breakout, where the paddle, the ball and the bricks can be considered three sub-projects. So, handling user input, movement and bouncing of walls as well as list for placing bricks are main topics for individual lessons.

It would also be conceivable to use several templates for differentiation. Teachers can provide different templates for different types of learners. By providing weaker students with other tasks or more support, e. g. through given blocks, within the framework of a project, they can be supported individually. In addition, the process of tracking students' progress and assisting them accordingly is enhanced using smerge.

Conclusion

In conclusion, this concept addresses all the initially described school-specific use-cases such as managing files and problems like blocked teams with code forgotten at home or sick students. It offers organizational advantages such as the possibility to share templates and skeletons in an easy manner, to start with multiple skeletons for different groups, to revert changes or work remotely from home. Teachers can concentrate on the pedagogical aspects of their lesson concept, as they are no longer involved in organizational activities such as setting up a server for a version control system. Regarding pedagogical advantages, both the current status of the project as well as its development process and progress become visible to both teachers and students. Because of the graphical visualization and the possibility to directly open, execute and test each node directly, this goes much further than professional version control systems. It allows for great flexibility and a constructionist way of teaching and learning: it supports PBL, differentiation, decomposing a greater whole in small learning lessons, or class-wide collaboration.

The outlined concept for a version control system enables collaboration in block-based languages. The version history provides a risk-free environment that invites users to experiment and tinker. Features, design and interface are reduced and adapted to the target group of novice programmers and based on existing research and experience regarding the use of version control systems. Smerge as an exemplary implementation of the concept offers all these features and is ready to be used in CSE.

References

- Barr, V., & Stephenson, C. (2011, March Volume 2 Issue 1). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads*, pp. 48-54.
- Brichzin, P., & Rau, T. (2015). Repositories zur Unterstützung von kollaborativen Arbeiten in Softwareprojekten [GERMAN]. *INFOS 2015 - Informatik allgemeinbildend begreifen* (pp. 73-82). Bonn, Germany: Lecture Notes in Informatics (LNI), Gesellschaft für Informatik.
- Broll, B., Lédeczi, A., Volgyesi, P., Sallai, J., Maroti, M., Carrillo, A., Weeden-Wright, S., Vanags, C., Swartz, J., Lu, M. (2017). A Visual Programming Environment for Learning Distributed Programming. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 81-86). New York, NY, USA: ACM.
- Chase, J. D., & Okie, E. G. (2000). Combining cooperative learning and peer instruction in introductory computer science. *SIGCSE '00 Proceedings of the thirty-first SIGCSE technical symposium on Computer science education* (pp. 372-376). New York: ACM.
- Computer Science Teachers Association. (2017). *CSTA K-12 Computer Science Standards, Revised 2017*. Retrieved from <http://www.csteachers.org/standards>

Fisker, K., McCall, D., Kölling, M., & Quig, B. (2008). Group Work Support for the BlueJ IDE. Proceedings of the 13th annual conference on Innovation and technology in computer science education (pp. 163-168). New York, NY, USA: ACM.

Glassy, L. (2006, Volume 21 Issue 3). Using version control to observe student software development processes. *Journal of Computing Sciences in Colleges*, pp. 99-106.

Haaranen, L., & Lehtinen, T. (2015). Teaching Git on the Side: Version Control System as a Course Platform. Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (pp. 87-92). New York, NY, USA: ACM.

Isomöttönen, V., & Cochez, M. (2014). Challenges and Confusions in Learning Version Control with Git. *Information and Communication Technologies in Education, Research, and Industrial Applications Communications in Computer and Information Science : 10th International Conference, ICTERI 2014* (pp. 178-193). Kherson, Ukraine: Springer International Publishing.

Kastl, P., & Romeike, R. (2015). "Now they just start working, and organize themselves" First Results of Introducing Agile Practices in Lessons. Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15) (pp. 25-28). New York, NY, USA: ACM.

Laurillard, D. (2009). The pedagogical challenges to collaborative technologies. *International Journal of Computer-Supported Collaborative Learning*. 4(1), pp. 5-20.

Lawrance, J., Jung, S., & Wiseman, C. (2013). Git on the cloud in the classroom. SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education (pp. 639-644). New York, NY, USA: ACM.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: A Sneak Preview. Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing (C5 '04). IEEE Computer Society (pp. 104-109). New York, NY, USA: ACM.

Monroy-Hernandez, A. (2012). Designing for remixing: Supporting an online community of amateur creators. Cambridge, MA, USA: Doctoral dissertation, Massachusetts Institute of Technology.

Ohshima, Y., Freudenberg, B., & Amelang, D. (2017). Kanto: a multi-participant screen-sharing system for Etoys, Snap!, and GP . Proceedings of the 3rd ACM SIGPLAN International Workshop on Programming Experience (pp. 7-10). New York, NY, USA: ACM.

Reid, K. L., & Wilson, G. V. (2005). Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments. SIGCSE '05 Proceedings of the 36th SIGCSE technical symposium on Computer science education (pp. 272-276). New York, NY, USA: ACM.

Resnick, M., & Rosenbaum, E. (2013). Designing for tinkerability. In M. Honey, & D. E. Kanter, *Design, make, play: Growing the next generation of STEM innovators* (pp. 163-181). New York, NY, USA: Routledge.

Romeike, R., & Göttel, T. (2012). Agile projects in high school computing education: emphasizing a learners' perspective . WiPSCE '12 Proceedings of the 7th Workshop in Primary and Secondary Computing Education (pp. 48-57). New York, NY, USA: ACM.

Somasundaram, R. (2013). *Git: Version control for everyone*. Birmingham, UK: Packt Publishing Ltd.

Racket Programming Material for Finnish Elementary Math Education

Tiina Partanen, *tiina.s.partanen@tampere.fi*
City of Tampere, Finland

Pia Niemelä, *pia.niemela@tut.fi*
Tampere University of Technology, Finland

Timo Poranen, *timo.t.poranen@uta.fi*
University of Tampere, Finland

Abstract

Programming is becoming a part of basic education in number of countries. 2014 Finnish National Curriculum (FNC-2014) for K-9 education embeds programming into the subjects of mathematics and crafts. The change in the curriculum challenges teacher training and causes a demand for appropriate teaching materials. This paper introduces a free Racket programming material for grades 5 to 9 fully integrable with mathematics lessons. Racket is a general purpose multi-paradigm programming language especially fit for education. The material consists of theory sections, corresponding exercises and answers. Covered math topics comprise natural and rational numbers, arithmetic operations, the order of operations, rounding, percentage calculations, scaling, polynomials, truth values, coordinate systems, geometric shapes, angles, etc. We evaluate the material by comparing the math concepts with the ones in FNC-2014.

Keywords

Racket programming material; computational thinking; computer science education; spiral curriculum; grades 5-9; mathematics

Introduction

In Finnish schools, programming has been mandatory since August 2016 (The Finnish National Curriculum, 2014). This retrospective change in teachers' job descriptions challenges the education system and creates an urgent demand for in-service training and training material. Unlike a number of other countries, Finland decided to add programming into the existing mathematics syllabus without granting time to cover the new material. This approach mandates tailoring the material to be as math-supportive and mutually beneficial as possible: while mathematical concepts ought not to be compromised, the learned programming concept subset should still be as expressive and exploitable as possible. As a further complication, the FNC-2014 requirements for programming are on a high abstraction level and defined only for certain grades (i.e., grades 2, 6, and 9; others were left to be specified locally), which impedes application.

In 2016, to support mathematics teachers in the elaboration of FNC-2014 requirements, the Finnish Association for Teachers of Mathematics, Physics, Chemistry and Informatics (MAOL) initiated a project to develop programming material for mathematics lessons. In 2017, the same material was used in a two-day hands-on in-service teacher training in fifteen cities all-around Finland. This paper presents the contents of the created Racket material concentrating on grades 5-9. We present the learning trajectory of the programming concepts and how they were integrated into the mathematics curriculum. The underlying research questions are:

1. Which kind of programming theory, material and exercises are foreseen beneficial for learning mathematics?
2. How do the introduced exercises comply with FNC-2014 and its underlying learning theory?

This study follows the principles of educational design research, where the feedback is used for incremental improvements (Akker et al., 2006). The rest of this paper is organized as follows. First, we

introduce the learning theory behind FNC-2014. Second, we review the programming requirements for Finnish schools. Third, the Racket programming material is introduced and followed by the comparison and evaluation with FNC-2014. Then we evaluate the material based on feedback received from teacher training. Finally, we give some directions for future research.

Constructivism

Learning theory of FNC-2014

The FNC-2014 is based on the constructivist theory of learning underpinning a large family of related theories. All emphasize an individual's responsibility of his own learning in structuring knowledge as consistent constructions, i.e., as schemas and other elaborations. As the most prominent schema building theory, Piaget and Duckworth (1970) launched the theory referred to as cognitive constructivism. Piaget studied a child's cognitive development and described learning as the formation of mental data structures as cognitive models, i.e., schemas. When a child faces a new concept, he will place it in the schema: 'assimilation' strengthens the schema a new concept is a snap to the sketch. Inconsistency in schema forces to reconstruct the data structures into a new one, i.e., the schema needs 'accommodation'. In particular, Piaget's genetic epistemology addresses the meaning of knowledge and forming schemes by employing reflective abstraction, the whole process being supported by self-regulation skills (Piaget and Duckworth, 1970). In summary, learning can be characterized as changes in one's schemata.

As an extension of student-centred learning, the new winds of phenomenon-based learning are blowing in FNC-2014. Even if as a concept of phenomenal learning is quite new, it complies with the long tradition of experimental and active learning traditions, such as 'learning by doing' (Dewey, 1902) and 'learning by making' (Papert, 1980). Both of these educational luminaries, Dewey and Papert, emphasize the affective side of learning and aim first at motivating students with engaging exercises. According to Dewey, the gained experiences must be relevant to a student in order to meaningful learning to take place. This view is seconded by Papert, who increments the idea with the cultural resonance, i.e., 'the topic must make sense in terms of a larger social context.'

Papert spent four years working under Piaget at the International Centre of Genetic Epistemology. Thus, Papert counts among a number of later development psychologists that own to Piaget. Papert's constructionism is founded on Piaget's cognitive constructivism, which Papert felt being too tightly personified to Piaget's constructivism often reads 'Piagetism'. However, constructionism shares constructivism's connotation of learning as 'building knowledge structures', but increments it with experiential learning, in other words, 'learning by making'. According to Kolb (2014), gaining experience is yet alone not enough, but the experiences must be elaborated with reflection and analysis to deliberately built deeper knowledge, the process of which Piaget refers to as reflective abstraction. Experimental and phenomenon-based learning has now-a-days strong advocates in particular in the discipline of science, its development being based on well-planned experiments.

Constructionism in math

Piaget's influence has diffused widely into different disciplines, in particular math. More systematically than other academic subjects, the syllabus of math relies on the consistent learning progressions and cumulative learning; topics proceed in consecutive steps by constructing on what has been learned previously. In a comprehensive school, the math syllabus can be divided into such main syllabus areas as arithmetic, algebra, and geometry, each of which can be examined separately by applying constructivist theories. First, the gradual possession of algebra fundamentals exemplifies the process of reflective abstraction. Next, geometry demonstrates evolving spatial perception. Last, Euclidean geometry is expanded as computational turtle geometry by Papert, who regards computers as highly valuable learning tools, in compliance with his theory about constructionism incrementing constructivism with computer-based exercises.

In the context of algebra, Piaget's genetic epistemology has suited as the action-process-object-schema (APOS) theory (Dubinsky and McDonald, 2001). The theory hypothesizes the subsequent steps of reflective abstraction started by simple actions of, e.g., counting or ordering, which are transformed as

more complex processes when they are recognized to share common patterns. Learning continues with the encapsulation of objects once APOS processes are abstracted as functions that contain mutable numbers as variables. The ultimate target is a constitution of flexible schemas based on the processes and objects, and fitting these schemas to the larger schemata of a learner.

In geometry, Van Hiele (1999) observed the deepening levels of spatial understanding. Analogously to APOS, Van Hiele levels of geometry consist of similar learning progressions: a student starts by recognizing the shape, then the patterns, i.e., analogies in shape properties, and comparison of the properties triggers an informal deduction phase, where students produce definitions in their own words; later these initial deductions are formalized. 'Rigor', the last van Hiele level, anticipates the maturity of transferring and applying the knowledge in different systems, which is often exemplified by the switch from ordinary Euclidean to non-Euclidean geometry. Van Hiele criticizes attempting too much too early in reference to the deteriorated learning results of the 'New Math' movement. Instead, the best learning outcomes are reached by adjusting the material appropriate to the level of children's thinking.

While proofing the theory of van Hiele levels of geometric thinking, Clements and Battista (1992) suggest a new zero level without any conditions referred to as 'pre-recognizing', where children perceive geometric shapes, but attend to only a subset of a shape's visual characteristics. In addition, their study implicates that the levels are not as discrete as described, rather *'there was instability and oscillation between the levels in several cases ... Continuity rather than jumps in learning was frequently observed.'* For an early exposure to spatial thinking, a computer is an excellent instrument. In an attempt to combine the affective and cognitive sides of learning, Papert (1996) launched turtles to stimulate both learning geometry and simultaneously exercise computing. To get a turtle to move in a desired direction and to draw the aimed geometric shapes, children are subtly exposed to the classifications and properties of shapes. Intendedly, turtle exercises are visually pleasing and capable of engaging students.

A number of Papert's principles overlap with Hungarian math referred to as the Varga-Nemenyi method that, e.g., Tikkanen (2008) introduces in her dissertation. For instance, to use learning tools, a lot of hands-on experiences and multisensory exercises that are refined as logic-mathematical conceptions when students progress along the 'way of abstraction'. In the same tune, Papert promotes a more playful way of nurturing computing basics and practice math in sync. According to his interpretation, computing is applied mathematics, and playing with turtles provides a gentle way of practicing it and strengthening a child's self-efficacy.

In essence, motivation and self-efficacy are accurate predictors of successful learning (Bandura, 2006). Thus, the instructional setup should focus on the affective side of learning instead of concentrating on mechanical routine work. Regrettably, in Finland, math attitudes deteriorate throughout the whole comprehensive school (Tikkanen, 2008), and the gap between poor and good students' skills to further widens, which is called Matthew effect, i.e., stronger students become stronger compared with poorer students (Lampinen and Korhonen, 2010). In contrast, the development in mother tongue is opposite so that the differences over time are reduced. Likewise, Papert is worried about the increasing number of math-phobic students that label themselves as too stupid to learn. He hypothesizes that this trend is a consequence of a too stiff way of solving problems and the pressure of getting it right at the first attempt, or in his words, the 'technology of grading'. Because of this phobia, self-efficacy-impaired students are mathematical under-performers. Programming experiments can be interpreted as a counteract of previous stiff problem-solving procedures criticized by Papert.

Programming in Finnish Schools

Programming syllabus emphasizes problem solving, algorithmic and logical thinking in particular. In addition, identifying the importance of math as the foundation for information and communication technology is a learning goal: future work and economy are dependent on the technological development and digital skills. FNC-2014 emphasizes digital competence as an interdisciplinary skill throughout all grades. The curriculum excerpts below mention programming explicitly in the objectives of two subjects, mathematics and crafts:

Grades 3-6

- Digital competence: "Students learn to program and become aware of how technology depends on decisions made by humans."
- Mathematics: "Students plan and implement programs using a visual programming language."
- Crafts: "Students practice programming robots and/or automation."

Grades 7-9

- Digital competence: "Programming is practiced as part of various other subjects."
- Mathematics: "Students should develop their algorithmic thinking and learn to solve problems using math and programming. In programming, students should practice good coding conventions."
- Crafts: "Students use embedded systems, plan, and apply programming skills in order to create products."

The mathematics syllabus has been rather stable over the recent years. The syllabus relies on the consistent and cumulative learning: topics proceed systematically by constructing on what has been learned previously. Traditionally, mathematics follows a spiral curriculum (Bruner, 2009), e.g., each year students learn some more geometry, and arithmetic and algebra related contents. However, FNC-2014 does not mandate the grade in which specific topics are to be introduced, instead, this can be decided at the local level.

As a new approach, FNC-2014 emphasizes a general skill-set, which students should acquire during primary education. Each subject should promote the development of transversal skills, such as, thinking skills, multiliteracy including digital literacy; ICT competence; working skills and entrepreneurship. In addition, FNC-2014 requires each school to provide cross-disciplinary learning modules that integrate multiple subjects, increase project- and phenomenon-based learning, and employ multiple learning environments also outside the school premises.

Racket Programming Material

Teachers must follow both national and local curricula, otherwise, they are free to select teaching methods and material. A few major school book publishing companies provide text book series. Due to the programming addition, the publishers started to update their math books. So far, the range of programming languages covers Scratch, Python and Processing. Free Racket online courses for teachers exist as well (Partanen et al., 2017).

To design, create and publish open programming material for math teachers, MAOL (2018) applied funding from the National Board of Education and from the Technology Industries of Finland Centennial Foundation. The founded working group consists of five experienced comprehensive school teachers, a university researcher and a freelance coder. For grades 1-6, the group developed material for Scratch (grades 1-6), ScratchJr (grades 1-2) and Lego Mindstorm (grades 3-4). For grades 5-9, the working group decided to exploit Racket (2018) language, the justification being the close conceptual linkage between algebra and functional paradigm. Racket is pure functional if limited to the Beginning Student language only. In addition, it has a strong pedagogical basis for computer science, and a versatile support for Windows, Mac OS, Linux and browser-based use. Moreover, the working group knew it well and had gained a strong experience.

The material was implemented in compliance with the following design principles: i) theory and exercises should be related to corresponding mathematical concepts, ii) based on FNC-2014, material for grades 5-6 should cover about 12 hours and the same amount for grades 7, 8, and 9 each, iii) the emphasis on 'learning by doing', iv) freely available material, except the answers which will be shared with teachers only, and v) exercises ranging from easy to difficult to enable differentiation. Racket material was published piecewise starting from fall 2016, and the latest additions were made at the end of 2017 (MAOL Racket, 2017). Altogether, it consists of 37 pages for grades 5-6, 54 for grade 7, 31 for grade 8, and 55 for grade 9 in print.

FNC-2014 mandates grades 3-6 to learn graphical programming. Complementarily, the material provides textual programming basics for grades 5-6 to foster fluency with keyboard, mouse, file saving,

and file system browsing – by and large, such practical skills they are often observed lacking. For these grades, learning targets familiarization with programming environment and basic syntax rules, arithmetic operations, functions for drawing simple geometric shapes, comparisons and their truth values, and variables for storing data. These concepts were integrated with math exercises, such as calculating percentages, fractions and decimals, and problem solving in geometric contexts. The content in more detail and the related curriculum specifications for grades 5-6 can be found in Table 1.

The material for grade 7 was built with no requirement of prior Racket knowledge. Often, grade 7 classes are formed of students coming from schools that have used different material, time, and languages for programming. If necessary, the previous grade 5-6 material can be used as a tutor for these students: programming content for 5-6 and 7 is analogous, but the grade 7 covers more advanced math problems including negative numbers, absolute value, powers, square root, a circumference of a circle, etc. Turtle geometry serves as a means to study the geometric properties of lines, polygons (especially triangles), a coordinate system, and the concepts of reflection and mirroring (see Fig. 1).

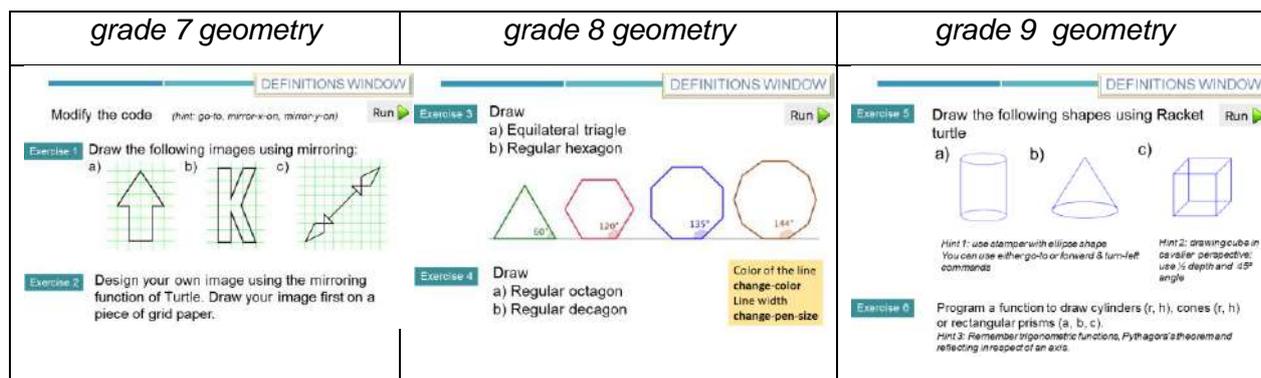


Figure 1. Exercise material on drawing Turtle images in grades 7-9

Fig.1 exemplifies geometry knowledge that gradually deepens starting from free drawings and advancing to more formal 2D shapes and 3D shapes whose properties are examined to induce the underlying mathematical rules. This gradual abstraction and formalization process demonstrates deepening van Hiele levels; visiting the same topics in a yearly basis manifests the spiral curriculum approach. Good coding conventions include a descriptive naming of variables, adding comments, and indenting code to improve its readability. The content in more detail and the related curriculum specifications for grade 7 can be found in Table 2.

The grade 8 material introduces plenty of core programming concepts such as truth values (binary values), comparisons, predicates and conditional expressions, function definitions, function parameters as variables, validity tests for user input, and error messages for invalid input. The listed programming content is practiced with polynomials (user-defined, tested with different arguments) and with geometry calculations, such as perimeter and area. User input is validated with conditions to eliminate negative side lengths and zero dividers, for example. In addition, material exploits turtle geometry to scaffold perceiving shapes that require repetition (circle) and/or parameterization.

Good coding conventions include descriptive naming for function and parameters, as well as a proper function signature (domain and range). The content in more detail and the related curriculum specifications for grade 8 can be found in Table 3.

The material for grade 9 covers more advanced topics such as lists, higher-order functions and recursion. The math topics origin from statistics (mean, mode, median, proportional frequencies, drawing bar charts), algebra (study of linear and quadratic functions and graphs), and arithmetic (calculating values using recursion, such as number sequences, loan payment schedules). In geometry, more advanced turtle graphics illustrate 3D shapes. A couple of quiz-style programs are provided for customization. Quiz applications could be implemented as math concept reviews, such as volumes of 3D-shapes, for example.

Table 1. Racket-material concepts in grades 5 and 6.

Module	Programming concepts	Math concepts	FNC-2014
Arithmetic: Introduction to programming	<ul style="list-style-type: none"> - types (numbers, strings) - function call (arguments) - simple syntax rules for expressions - nested function calls 	<ul style="list-style-type: none"> - natural numbers - arithmetic operations (+, -, *, /) - order of operations 	Practice basic arithmetic operations in varying situations and using needed tools.
Algebra: Variables	<ul style="list-style-type: none"> - defining variables (constants) - expressions using variables 	<ul style="list-style-type: none"> - positive rational numbers (decimal number) 	Familiarize with the notion of unknown. Examine equation and find solutions heuristically. Study decimal numbers as a part of numbers in decimal system and practice calculations using them.
Logic: comparisons	<ul style="list-style-type: none"> - truth values - comparison operations (<, >, =) - decimal numbers 	<ul style="list-style-type: none"> - truth values of propositions - equality and inequality 	
Geometry: Image programming	<ul style="list-style-type: none"> - library - calling a function with different arguments to get a different output - pixel graphics - combining images from smaller images 	<ul style="list-style-type: none"> - function machine (input, output) - measures in rectangles (width, height) - area and perimeter - reflecting with respect to an axis - axial symmetry 	Practice skills in finding similarities, differences and patterns. Design and implement programs in graphical programming environment. Measure and calculate the area and perimeter of different geometric shapes. Observe axial symmetry.
Arithmetic: Fractions and percentages	<ul style="list-style-type: none"> - fractions (improper/mixed fractions) 	<ul style="list-style-type: none"> - rational numbers (improper/mixed fractions) - percentage (calculating percentage of a number) - presenting fractions visually as images - ratio of distances in the context of scaling images (smaller/bigger) 	Learn the notion of fractions and practice arithmetic operations in different situations. Create the basis for understanding percentages and percentage of a value, practice calculating these. Use the relations between fractions, decimal numbers and percentages. Familiarize with ratio of distances and use it in the context of reduction and enlargement.

Table 2. Racket-material concepts in grade 7.

Module	Programming concepts	Math concepts	FNC-2014
Arithmetic	<ul style="list-style-type: none"> - integers, decimal number, fractions (improper/mixed) - function call (arguments) - simple syntax rules for expressions - evaluation of expressions - nested function calls 	<ul style="list-style-type: none"> - rational numbers - arithmetic operations (+, -, *, /) - order of operations - approximating result - rounding - exponentiation - square root - absolute value - remainder 	Practice arithmetic operations also with negative numbers. Strengthen numeracy in the context of fractions and learn multiplication and division with fractions. Deepen the know-how of decimal number operations. Familiarize with the notion of multiplicative inverse and absolute value. Strengthen understanding of

	<ul style="list-style-type: none"> - rounding (floor, ceiling) - math functions (expt, sqrt, abs, modulo, random) 	<ul style="list-style-type: none"> - random numbers 	<p>the difference between exact and approximate numbers and rounding. Practice calculating powers and exponents with integer indices. Learn the principle of square root and its use in calculations.</p>
Algebra: Expressions	<ul style="list-style-type: none"> - definition of variables (constants) and expressions using variables - expression evaluation - comments 	<ul style="list-style-type: none"> - variable expressions - value of expression - areas and/or perimeters of rectangles, triangles and circles - angles in triangles and of intersecting lines - word problems 	<p>Learn the concept of variable and how to calculate the value of an expression.</p> <p>Study qualities of lines, angles and polygons.</p>
Strings	<ul style="list-style-type: none"> - types (numbers, strings, images) - string operations - type conversions 		
Geometry: Images	<ul style="list-style-type: none"> - library - calling a function with different arguments to get a different output 	<ul style="list-style-type: none"> - measures in basic shapes (circle, square, rectangle, triangle) - points and lines in coordinate system - rotation and scaling 	<p>Study qualities of lines, angles and polygons.</p>
Geometry: Turtle geometry	<ul style="list-style-type: none"> - loading a library extension - using library functions - defining a list of drawing functions in correct sequence 	<ul style="list-style-type: none"> - angles and side lengths in polygons - different triangles (right triangle, isosceles triangle, equilateral triangle) - parallel and perpendicular lines - coordinate system - reflecting with respect to an axis and mirroring relative to a point 	<p>Understand concepts of point, line segment, line, angle [and study the concepts of path and half-open line segment].</p> <p>Practice geometric constructions. Observe axial symmetry. Study the first quarter of the coordinate system and extend it to contain all quarters (grades 3-6)</p>
Good coding conventions	<ul style="list-style-type: none"> - descriptive naming of variables, code indentation 		<p>Program and practice good coding conventions.</p>

Table 3. Racket-material concepts in grade 8.

Module	Programming concepts	Math concepts	FNC-2014
Algebra: Function	<ul style="list-style-type: none"> - function definition (names of function and parameters) - function output with different input - understanding error messages 	<ul style="list-style-type: none"> - polynomials - function (polynomial function) - calculating the value of a function with different arguments 	<p>Learn the concept of variable and how to calculate the value of an expression. Familiarize with the concept of polynomials.</p>
Logic: Truth values	<ul style="list-style-type: none"> - bits - truth values - comparison operations (<, >, =, <=, >=) - Boolean operators (and, or, not) 	<ul style="list-style-type: none"> - binary and decimal systems - truth values - propositional logic connectives (and, or, not) - defining intervals in number line ($0 < x < 4$) 	<p>Practice logic thinking such as finding rules and dependencies and presenting them appropriately. Ponder and study the number of alternatives. Practice</p>

	- predicates (number?, even?)	- parity	logical thinking by reasoning truth values of propositions.
Logic: Conditional structure	- conditional structure (if-else) - testing the validity of the input and giving relevant error messages - nested if-expressions and multiple selection (cond)	- defining domain and range for a function - detecting undefined situations: division by zero, square root of negative number, negative radius - calculating percentage (progressive tax of income) - naming angles	Practice calculating percentage of the whole.
Geometry: Plane geometry		-area of geometric shapes (rectangle, parallelogram, circle) -circumference of geometric shapes (square, rectangle, circle) -Pythagorean theorem (right triangle)	Adopt usage of a larger number set (real numbers). Learn to use Pythagorean theorem and inverse Pythagorean theorem. Calculate perimeters and areas of polygons. Practice calculating the area and circumference of a circle, the area of a sector and the length of its arc.
Geometry: Turtle geometry	- repetition - function that returns a list as output	- regular polygons - approximation of a circle as a regular polygon - arcs	Study qualities of lines, angles and polygons.
Good coding conventions	- descriptive naming of functions - function signature e.g. input and output types		Program and practice good coding conventions.

The rest of the material is targeted for the students who seek more challenge. The exercises include basic algorithms: recursion (list processing), search (binary search), sorting (bubble sort), and optimization (greedy algorithm and dynamic programming). Good coding conventions in grade 9 include writing unit tests for functions. The content in more detail and the related curriculum specifications for grade 9 can be found in Table 4.

Table 4. Racket-material concepts in grade 9.

Module	Programming concepts	Math concepts	FNC-2014
Statistics: Lists and statistics	- list and list operations - nested lists - range - higher order functions (apply, map) - sorting - filtering	- arithmetic sequence - a bar chart - mean, mode, median, frequency, relative frequency - divisibility - prime numbers with Aristoteles sieve	Deepen skills for gathering, classifying and analyzing information. Understand the concept of mean, mode and median. Practice finding frequency, proportional frequency and median. Study and create different diagrams. Learn about the divisibility of numbers to their prime factors. Deepen skills to study and form number sequences.
Algebra: Functions and graphs	- loops - vector - recursive functions - test cases for functions - local parameters (let)	- graphs of functions (first and second order) - number sequences (arithmetic and geometric)	Describe dependencies using both graphs and algebra. Study the concept of a function.

	- creating a list recursively	- interest (calculating loan payment schedule) - compound interest - random numbers - factorial	Draw lines and parabolas in coordinate system. Learn the principles of slope and constant. Study graphs. Practice calculating percentage of the whole.
Geometry: Turtle graphics	- using higher order functions and range to create lists	- division of geometric shapes into similar regions - number sequences in images (spirals etc.) - 3D shapes (cylinder, cone, cube)	Deepen skills to study and form number sequences. Learn to use Pythagoras Theorem and trigonometric functions. Study 3D shapes.
Applications: a quiz game and calculator for geometric properties	- user interactions (usage of a simple GUI library) - modifying a readymade code to serve a new purpose - dividing code into helper functions	- quiz on any suitable math related topic (for example functions, 3D-geometry or unit conversions) - formulas to calculate area and/or volume of a 3D-shape - rounding - units for area/volume	Apply self- or ready-made programs into math studies. Learn to calculate surface areas and volumes of sphere, cylinder and cone. Ensure and enlarge the mastery of units of measure and unit conversions.
Good coding conventions	- test cases for functions		Program and practice good coding conventions.

Evaluation

A part of this material was exploited in the series of in-service math teacher trainings. The majority of course participants were math teachers (95.7%), female (78.1%) and teaching in grades 7-9 (92.4%). In general, the participants had quite long work histories: >15 years (38.3%), 6-15 years (39.5%), 0-5 years (22.2%). The majority of the participants had some previous programming experience (55%) or lots of experience (6.4%), however, 38.6% had no experience at all.

On the first day of the training, the following material was used: i) Introduction to programming environment (grades 5-6); ii) Image programming (grade 7); and iii) Functions, Truth values and Conditional structure (grade 8). On the second day, the material was: iv) Lists and statistics (grade 9); and v) Turtle geometry (grades 7-9). The participants were assigned to give a programming lesson to their students in-between the course dates by exploiting the material. After the second day, the course participants (N=329) assessed the material in a five-point Likert scale. The statements were: i) The used material is articulate, ii) The material is well suited for teaching purposes, iii) The material is comprehensive, and iv) The material contains enough exercises suitable for different types of students.

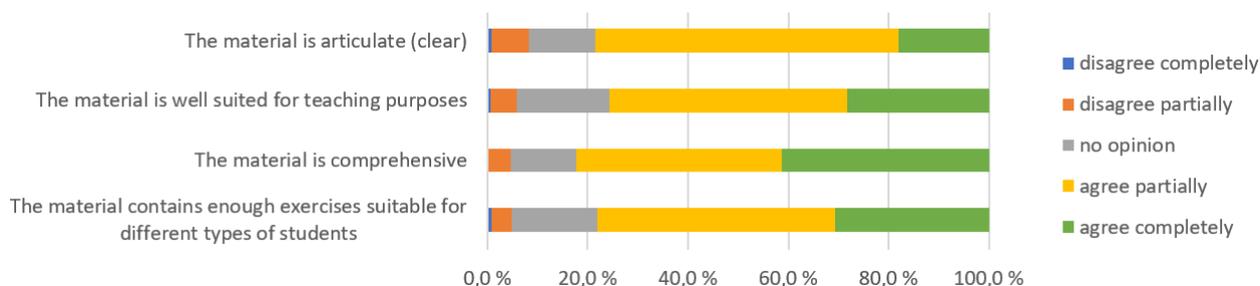


Figure 2. Racket material feedback after in-service training

Based on the feedback, the comprehensiveness scored the best, in addition, the material was considered rather articulate, suited for teaching purposes, and to include a sufficient number of exercises for differentiation. However, the review of the FNC-2014 requirements revealed the lack of integrative programming projects. In addition, some mathematics topics are still missing, such as:

- Interpretation and production of mathematical texts, basic proofs.
- Changed and base value, change- and comparison percentages.
- The addition, subtraction and multiplication of polynomials.
- The reduction of exponentials
- Linear, partial quadratic, simultaneous equations, roots.
- Graphical and algebraic solving.
- First-degree inequalities.
- Proportion in problem solving; direct and indirect proportionality.
- Concepts of similarity and congruence.
- Inscribed and central angles [in circles] and Thales's theorem.
- The concept of dispersion, probability.

Conclusion and future work

The examined Racket programming material for grades 5-9 was successfully used in teacher training and it received positive feedback from the math teachers. Further research should evaluate the current utilization of the material by the course participants and other mathematics teachers, collect feedback systematically and analyse students' learning outcomes both in math and programming, also in comparison with other corresponding materials. After the analysis, the content could then be revisited and developed further in correspondence.

References

- Akker, J.V., Gravemeijer, K., McKenney, S. and Nieveen, N. (eds.), (2006) Educational design research. London, Routledge.
- Arnon, I., Cottrill, J., Dubinsky, E., Oktaç, A., Fuentes, S. R., Trigueros, M., and Weller, K. (2013) APOS theory: A framework for research and curriculum development in mathematics education. Springer Science & Business Media.
- Bandura, A. (2006) Guide for constructing self-efficacy scales. Self-efficacy beliefs of adolescents, vol. 5, no. 307-337.
- Bruner, J. S. (2009). The process of education. Harvard University Press.
- Clements, D. H. and Battista, M. T. (1992) Geometry and spatial reasoning. In Handbook of research on mathematics teaching and learning, pp. 420 – 464. MacMillan.
- Dewey, J. (1902) The child and the curriculum. University of Chicago Press, no. 5.
- Dubinsky, E. and McDonald, M. A. (2001) APOS: A constructivist theory of learning in undergraduate mathematics education research. In The teaching and learning of mathematics at university level. Springer, pp. 275 – 282.
- The Finnish Association for Teachers of Mathematics, Physics, Chemistry and Informatics (2018) MAOL ry., <http://www.maol.fi>.
- Finnish National Board of Education (2014), Finnish National Curriculum 2014 (FNC-2014).
- Kolb, D. A. (2014) Experiential learning: Experience as the source of learning and development. FT Press.
- Lampinen, A. and Korhonen, H. (2010) Suomessa opitaan matematiikkaa Varga-Neményi -menetelmän mukaan. Dimensio, vol. 74, no. 2, pp. 24 – 28.
- MAOL Racket (2017) Programming in primary school mathematics, in Finnish, MAOL, <https://peda.net/yhdistykset/maol-ry/materiaalit/kpm>
- Papert, S. (1980) Mindstorms: Children, computers, and powerful ideas. Basic Books.
- Papert, S. (1996) An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning, vol. 1, no. 1, pp. 95 – 123.

Partanen, T., Niemelä, P., Mannila, L. and Poranen, T. (2017) Educating computer science educators online - A Racket MOOC for Elementary Math Teachers of Finland. In Proc.: 9th International Conference on Computer Supported Education 2017, pp. 47-58.

Piaget, J. and Duckworth, E. (1970) Genetic epistemology. *American Behavioral Scientist*, vol. 13, no. 3, pp. 459 – 480.

Racket (2018) Racket programming language, <http://racket-lang.org/>

Tikkanen, P. (2008) "Helpompaa ja hauskeempaa kuin luulin": matematiikka suomalaisten ja unkarilaisten perusopetuksen neljäsluokkalaisten kokemana. *Jyväskylä studies in education, psychology and social research* 337. University of Jyväskylä.

Van de Walle, J. A., Karp, K. S., and Bay-Williams, J. M. (2004) *Elementary and middle school mathematics*. Boston: Allyn and Bacon.

Van Hiele, P. M. (1999) Developing geometric thinking through activities that begin with play. *Teaching children mathematics*, vol. 5, no. 6, p. 310.

Using Agent-based Modelling of Collaboration for Social Reflection

Evgeny Patarakin, patarakined@mgpu.ru
Moscow City University, Russia

Abstract

This paper presents techniques that unite the production aimed at creation of mutual stories with the research activity based on the analysis of the relationship between the participants of the productive activity. The production is delivered through different mediums where all actions of the participants are recorded in an electronic log. Log records are used as a base material (substrate) to build sociograms, which are the start for the research activity. Different cycles of collaboration based on the usage of social objects are described in the paper.

Keywords

Collaboration; ABM; Netlogo; wiki; social reflection

Introduction

Social networks, data on relations between collaboration participants, social network analysis techniques are hardly used for inquiry-based learning. It comes from the impression that the data is hard to access or the methods are too complicated. In this paper we present simple methods of learning analytics based on the data on collaboration as well as on the dynamic NetLogo models. The source material for the research is log records which are a by-product of digital storytelling creative process. The side results of productive activity are used as a feed substrate for the network research based on the analysis of sociograms. It provides an advanced pathway to developing learning programs that utilize the science of complex networks as a vehicle through which students can learn analytical skills for network-oriented data analysis.

Agent-based modelling of collaboration

Immersing school students in the field of network science begins with the study of maps that are based on data from different areas of research. For example, science maps for kids are based on the avalanche of data generated by scientific research today. These maps invite students to see, explore, and understand science (Börner et al., 2009). Introducing students and teachers to the network science may begin with mapping their own activities in the networked communities. The advantage of this approach is a network lens is used to understand situations in which students and teachers are drawn into. Consequently the network science shows its strength in the areas of immediate experience and students and teachers become researchers of their own activity. The desired future of modern learning can be described as a situation where teachers and students are active agents that produce knowledge. A necessity of formation participant's agency requires mentors of learning to put special attention to the problems of self-control and self-determination of behavior of participants. Schwartz (Schwartz, 1999) termed this formulation «productive agency», because it emphasizes production through the environment. Schwartz defined productive agency as a recursive system where people take advantage of their available means to produce outwardly, and they see their ideas embodied and modified by the material or social world. Production distinguishes between situations that involve external production and those that do not and in this regard productive agency is akin to constructionism. Constructionism argues that learning occurs best when constructing a public artifact.

Constructionism shares constructivism's connotation of learning as 'building knowledge structures' irrespective of the circumstances of the learning. It then adds that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sandcastle or a theory of the universe... (Papert & Harel, 1991).

Public entity or a public artifact can be not only examined or discussed but it can also be used by other people. That is to say a public artifact is inherently shareable.

Comparing the known learning communities close to the constructionism theory allows us to see that almost all of them use the idea of a cycle or a spiral of actions performed by agents over objects of actions:

- In the Letopisi.org (Y. Patarakin & Shilova, 2015) teachers and students perform the following actions with wiki pages: Read -> Create -> Edit -> Connect -> Share -> Read.
- In the Globaloria Social Learning Network (Reynolds & Caperton, 2009) students perform the following actions with computer games: Play -> Plan -> Prototype -> Program -> Publish.
- In the Scratch Social Learning Network (Brennan, Hernández, & Resnick, 2009) students perform the following actions with Scratch projects: Imagine -> Create -> Play -> Share.
- In the NetLogo Modeling Commons (Lerner, 2014) NetLogo modelers perform the following actions with NetLogo Models: Create -> Run -> Share -> Comment -> Modify -> Create variations.
- In the StarLogo TNG Social Learning Network there are two linked circles (Klopfer, Scheintaub, Huang, Wendel, & Roque, 2009) and StarLogo TNG modelers perform the following actions with StarLogo Models. Research circle: Observe/Collect Data -> Generate Questions -> Test/Tinker/Play -> Observe/Collect Data. Design circle: Design -> Build -> Test/Tinker/Play -> Design.
- In the Looking Glass Social Learning Network (Kelleher & Pausch, 2007) 3D modelers perform the following actions with Alice Models: Create -> Animate -> Remix -> Share.

In all the above mentioned communities there is a definite social object (Engeström, 2005) – a page, game, a story, a model or other “virtual chips” over which agents perform their actions. Objects or actions can be different, but all agents are permitted to carry actions over one and the same object. If agents perform actions over one and the same object, they become indirectly connected by the social object. The links are similar to those between movie actors, or scientists creating an article, or Wikipedia editors working on a page together.

Diagrams methodology help to analyze and discuss situations that develop during a network collaboration in different domains. A sociogram is a powerful analysis tool, helping researchers identify points of interest and other structural properties that otherwise would not be obvious in numeric data. We make maps not just of the physical world but also of our social worlds (Mehra et al., 2014).

The study of dynamic networks greatly benefits from visualizations that can illustrate ideas and concepts not immediately visible in a static sociogram. Moody and others' research illustrates the need to visualize how networks develop and change over time (Moody, Mcfarl, & Bender-demoll, 2005). Among the tools developed in the complexity field, agent-based modeling and network analysis are very important in sustaining the process of bringing complexity to bear on the policy world. The combination of the two methods can increase enormously the potential of complexity-based policies (Fontana & Terna, 2015). Since agent-based modeling is inherently dynamic, the problems with static networks are overcome naturally. Agent-based modeling permits the desired richness of behaviors and attributes that might bridge the gap between agent-nodes and the real world.

Termites with Logs

It was mentioned in the previous section that actors perform actions over same objects becoming interlinked via this social object in many various situations. Simple actions that students perform over objects of actions in learning communities are very much alike the procedures performed by turtles in a famous Termites model: search-for-chip -> find-new-pile -> put-down-chip

As Resnick wrote (Resnick, 1997) each individual termite should obey the following rules:

If you are not carrying anything and you bump into wood chip, pick it up.

If you are carrying a wood chip and you bump into another wood chip, put down the wood chip you're carrying.

Each turtle performs a sequence of steps in procedures search-for-chip find-new-pile put-down-chip which leads to the result when chips scattered randomly over the screen are gradually gathered into

one round pile. Termites model seems most optimal because it contains chips as objects for collaboration. Experimenting on this model we can get a deeper understanding of collaboration phenomenon. Let's imagine that termites record their labor actions over chips in a log. I.e. if an actor performs a meaningful action on a chip, he leaves a record on this in the log. To understand how we can benefit from collateral records we have modified the original text of Termites model by adding new variables and rules. We have inserted a variable called list WIKILOG where turtles make records of their actions. There also have been made some additions to the procedures `search-for-chip` and `put-down-chip` (E. D. Patarakin, 2017).

The interface for the modified model is shown in Figure 1. Logs_to_sociogram button switches all the chips off the screen and only shows linkage between agents carrying one and the same chip.

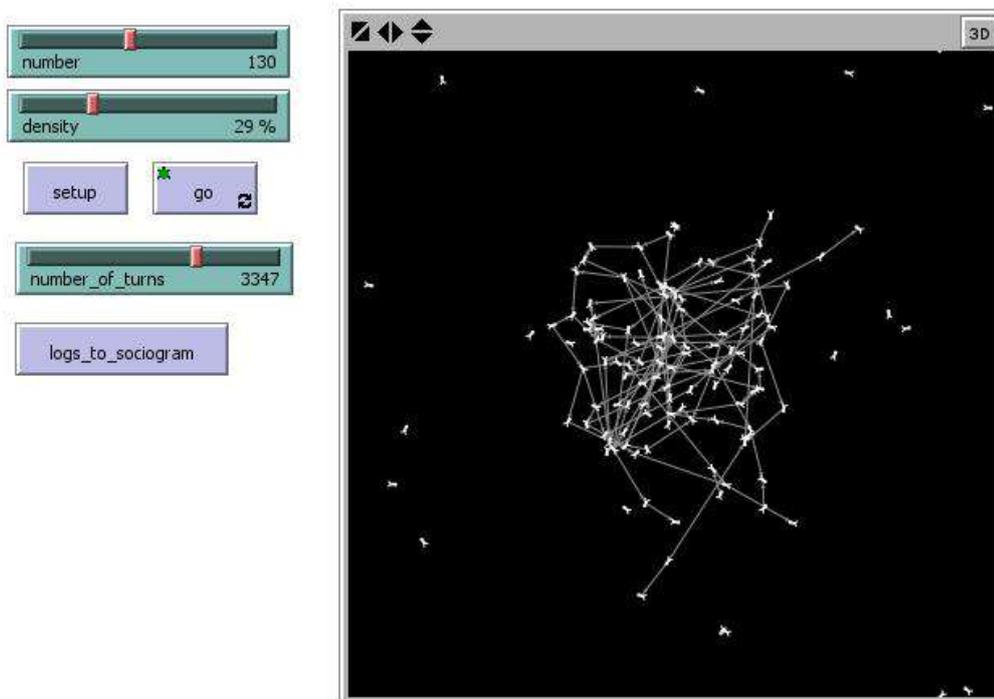


Figure 1. Interface of Termites with Logs model

Termites with Log Model was implemented in NetLogo language 5.2 and its source code is available on the Internet http://modelingcommons.org/browse/one_model/4749

This simplest model for sociograms creation based on the actions with social objects can be applied to various collaboration situations. In all cases when actors perform actions over shareable objects, the log for these actions can help build sociograms as well as conduct social research. Full history can be presented as a record of a game, consisting of many moves. Each move in a game contains three required elements:

Agent ID| Object ID| Type of an action

Every action of an agent towards an object leads to the formation of a link between them. If the agents perform action over one and the same object they become agents of the collaborative activity, indirectly linked with one another by the mutual object of activity. The collaborative activity network could be presented as the bipartite graph combining agents with objects of collaborative activity.

Our second learning analytics application was implemented in NetLogo language 5.2 and its source code is available on the Internet as Dynamic Wikigram Model http://modelingcommons.org/browse/one_model/4769

The use of NetLogo allowed us to present the collaboration as a dynamic diagram in which each actor can interact with each other tens of thousands of actors. The created model used NetLogo features such as breeds and agentset. Interface of Dynamic Wikigram Model presented in Figure 2

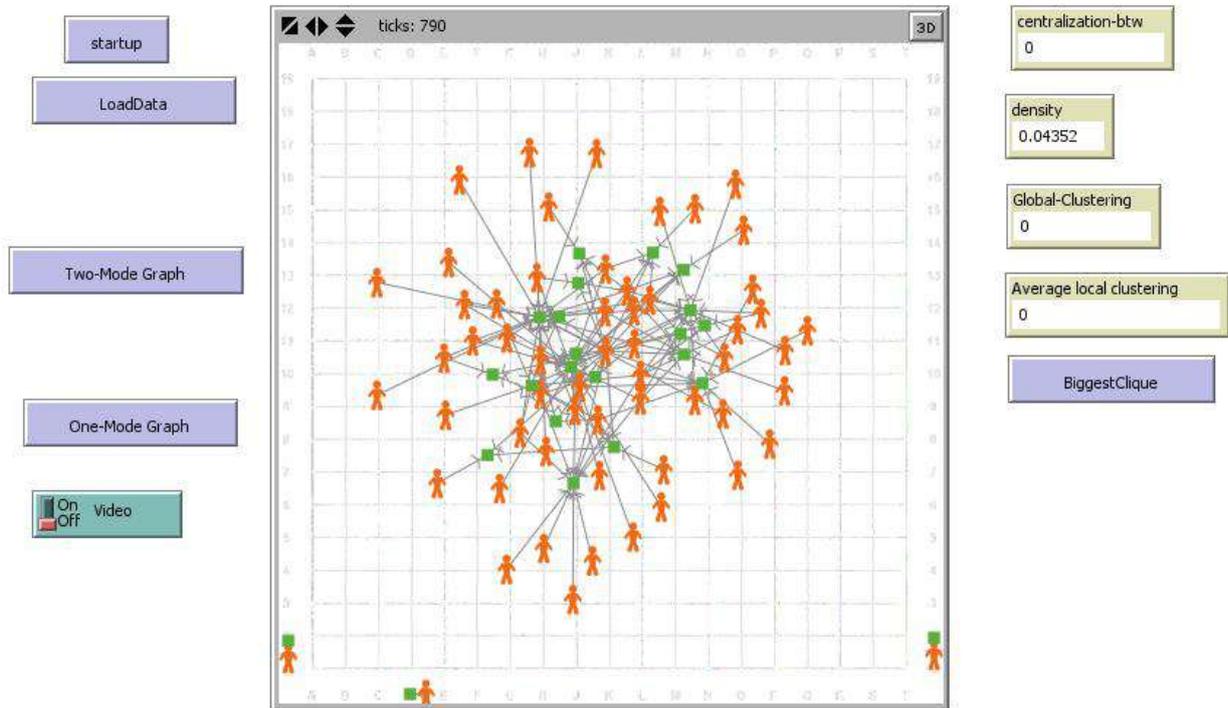


Figure 2. Interface of Dynamic Wikigram Model

Dynamic Wikigram Model can be used as a simple and illustrative tool of analysis. During his/her research, a student uses the technique of dynamic agent-based sociograms to:

- Trace how and based on what objects' links between participants of collaborative productive activity are formed.
- Identify key players and stable biggest cliques, which serve as cores that support the operation of network communities

Key players are those elements in the network that are considered important, in regard to certain criteria. One of the most popular criteria is the betweenness centrality of vertex. The betweenness of an edge is the number of these paths running through it. It is clear that, when a graph is made of tightly bound clusters, loosely interconnected, all shortest paths between nodes in different clusters have to go through the few interclusters connections, which therefore have a large betweenness value. A node with high betweenness centrality is responsible for connecting many pairs of nodes via the best path, and deleting that node should cause many pairs of nodes to be more distinctly. The idea behind betweenness centrality is that being in between actors makes actor powerful because he may be able to control the flow of information between them (Borgatti, 2006). Nodes with high betweenness centrality are often called key-players. To calculate the betweenness centrality of a Netlogo turtle, you take every other possible pairs of turtles and, for each pair, you calculate the proportion of shortest paths between members of the pair that passes through the current turtle. The betweenness centrality of a turtle is the sum of these. Top ten key players are determined in NetLogo as

```
sublist reverse sort-on [norm-betweenness] users 0 9
```

Key players are important in themselves and as nodes that connect communities. Qualitatively, a community is defined as a subset of nodes within the graph such that connections between the nodes are denser than connections with the rest of the network (Radicchi, Castellano, Cecconi, Loreto, & Parisi, 2004). The detection of the community structure in a network is generally intended as a procedure for mapping the network into a tree. For the divisive class of algorithms one starts with the whole graph and iteratively cuts the edges. The crucial point in a divisive algorithm is the selection of

the edges to be cut. Girvan and Newman have introduced an “edge betweenness algorithm” where the selection of the edges to be cut is based on the value of their edge betweenness centrality (Girvan & Newman, 2002). The single step of the edge betweenness algorithm consists in the computation of the edge betweenness for all edges in the graph and in the removal of those key players with the highest score. The iteration of this procedure leads to the splitting of the network into disconnected subgraphs, until the whole graph is divided in a set of isolated nodes. At each step of the edge betweenness algorithm NetLogo model created the sociogram, which was used to discuss the role of a key players (Figure 3).

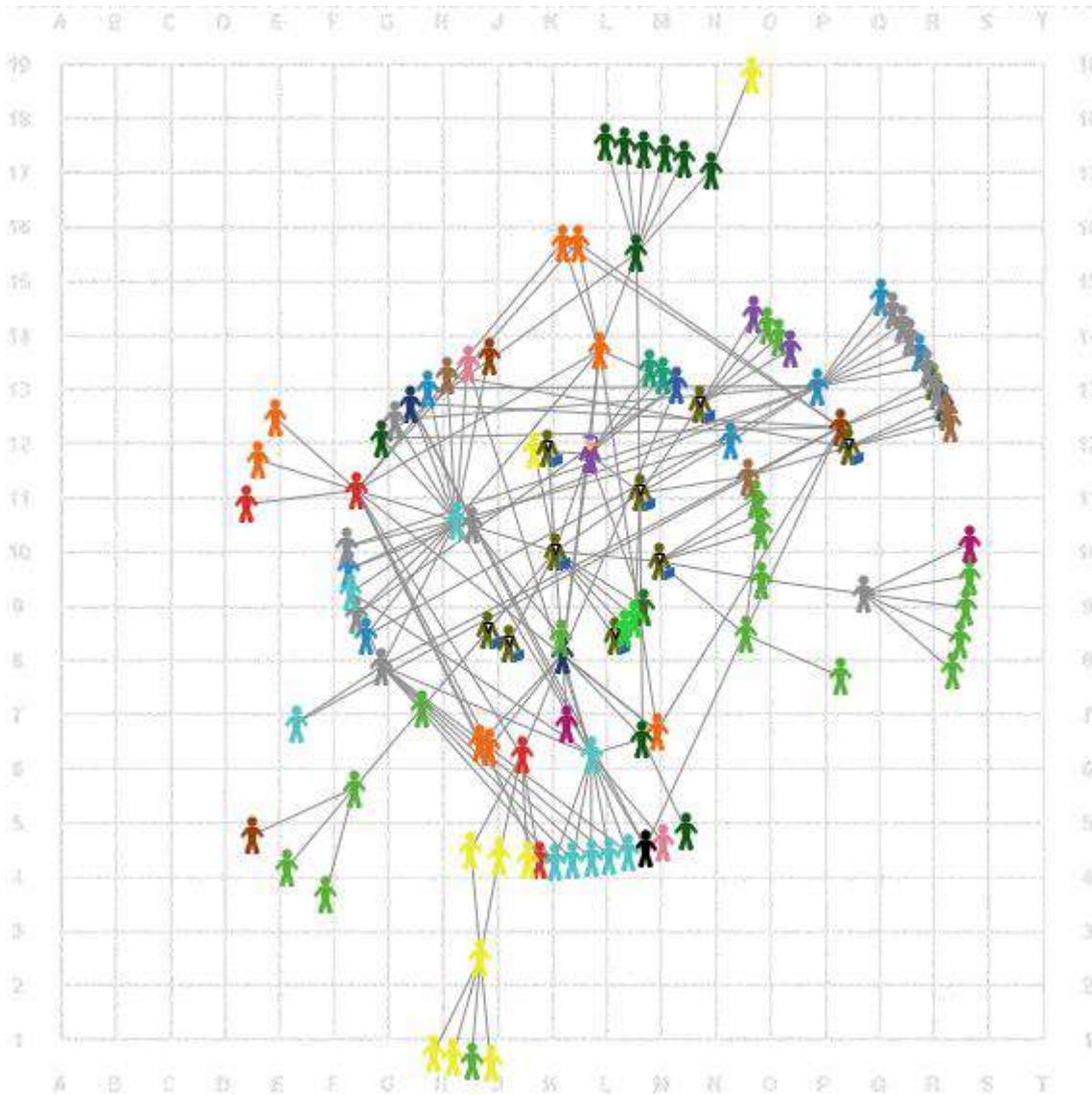


Figure 3. Wikigram with key player in the center

A clique is a subset of a network in which every node has a direct link to every other node. A maximal clique is a clique that is not itself contained in a bigger clique (Figure 4). Cliques containing more than N members are determined in NetLogo as

```
nw:maximal-cliques [if (count ?) > N
```

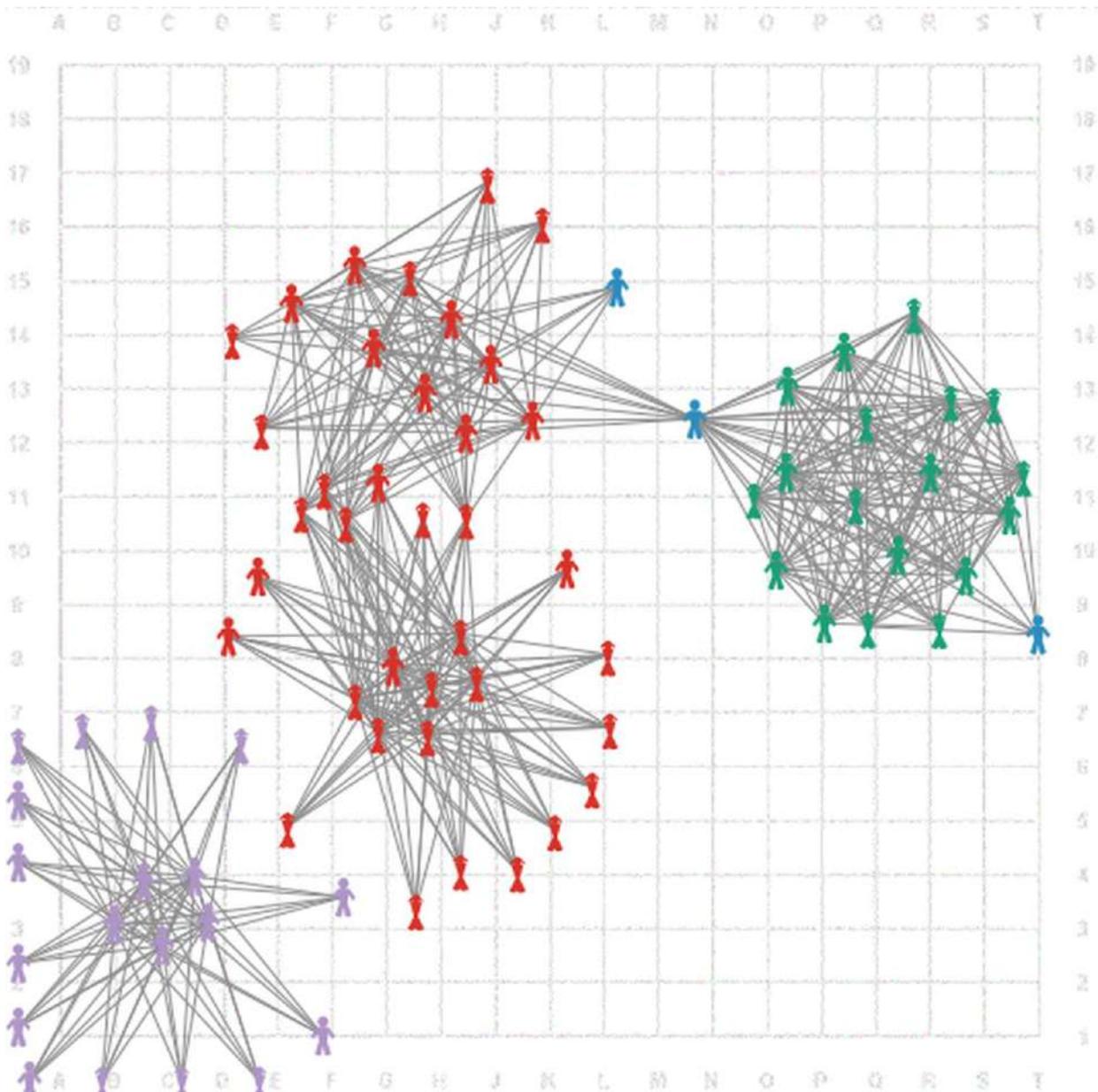


Figure 4. Sociogram of maximal cliques

The projected system of the collaborative network activity gives extra possibilities not only for the productive activity but also for the analysis and reflection on the processes inside the system. The tools should open the possibility to assess from the network point of view both the position of each participant and the degree of the whole system development as the learning network. The analysis of the activity of each participant inside the acting community combined by mutual story or mutual game, allows us linking the act of activity and the development of one participant with the development of the whole community.

We have successfully used NetLogo wikigram application to support several socio-educational projects.

Letopisi.org In Russian education MediaWiki is represented first of all by Letopisi.org project and its regional clones in several teaching colleges and universities. Letopisi - <http://Letopisi.org> is the national educational project with international participation and has continued for more than twelve years. The leading idea of the Letopisi concept is that the collaborative network activity and the network cooperation of the learning agents are aimed at the creation of various types of learning products which in general could be marked by a widely recognized term “digital story”. The basic scheme of the concept is the following: the digital story and the constituent elements of it could be used by other participants of the

collaborative activity in creation of new stories. In the Letopisi.org students and teachers perform the following actions with wiki pages:

Read -> Create -> Edit -> Connect -> Share -> Read

Preobra.ru - collaborative environment for creation, improvement and promoting bills within public and legislative projects (Burov, Patarakin, & Yarmakhov, 2012). Enacting a new law means that a community devises out new rules which help it to become more efficient. In the general case, the web site contains a complete text of the document, which chapters and items were split in small segments. The project participants can create their own segment versions and vote for segments created by other participants. Public construction of a document aiming at complex cloud issues has high educational value. This collaborative practice helps not only produce a quality document and build a community of people interested in its implementation, but promote the innovative document, maintain a new level of its understanding and perception by the society. In the Preobra.ru teachers perform the following actions with parts of document: Writing, Editing, Voting

Google Apps or G suite. Information environment of the complex of schools that share Google Apps domain. Participants can create, view, and edit digital objects of various types. Participants vary by positions, teaching subjects and campuses they work in. Learning analytics and diagrams methodology help to analyze and discuss situations that develop during a network collaboration and build school communities of practice. The data, required for such a research can be extracted from a school domain. Netlogo package contains all the features, needed for such data analysis. Our research has shown clusters of most connected teachers and staff and located collaborative documents that play the role of boundary objects, connecting clusters of school administrators, classroom teachers and staff members. NetLogo allows you to define different breeds of turtles and breeds of links. Once you have defined breeds, you can go on and make the different breeds look differently and behave differently. We have used various breeds of agents to separate the subjects and objects of activity, as well as to separate the actors into different classes. An agentset Netlogo is exactly what its name implies, a set of agents. But what's powerful about the agentset concept is that you can construct agentsets that contain only some turtles or some links. Due to this we are able to identify network characteristics that are typical for certain groups of actors. For example, only administrators, only for teachers of literature, only to employees of a particular territorial office etc (Figure 5).

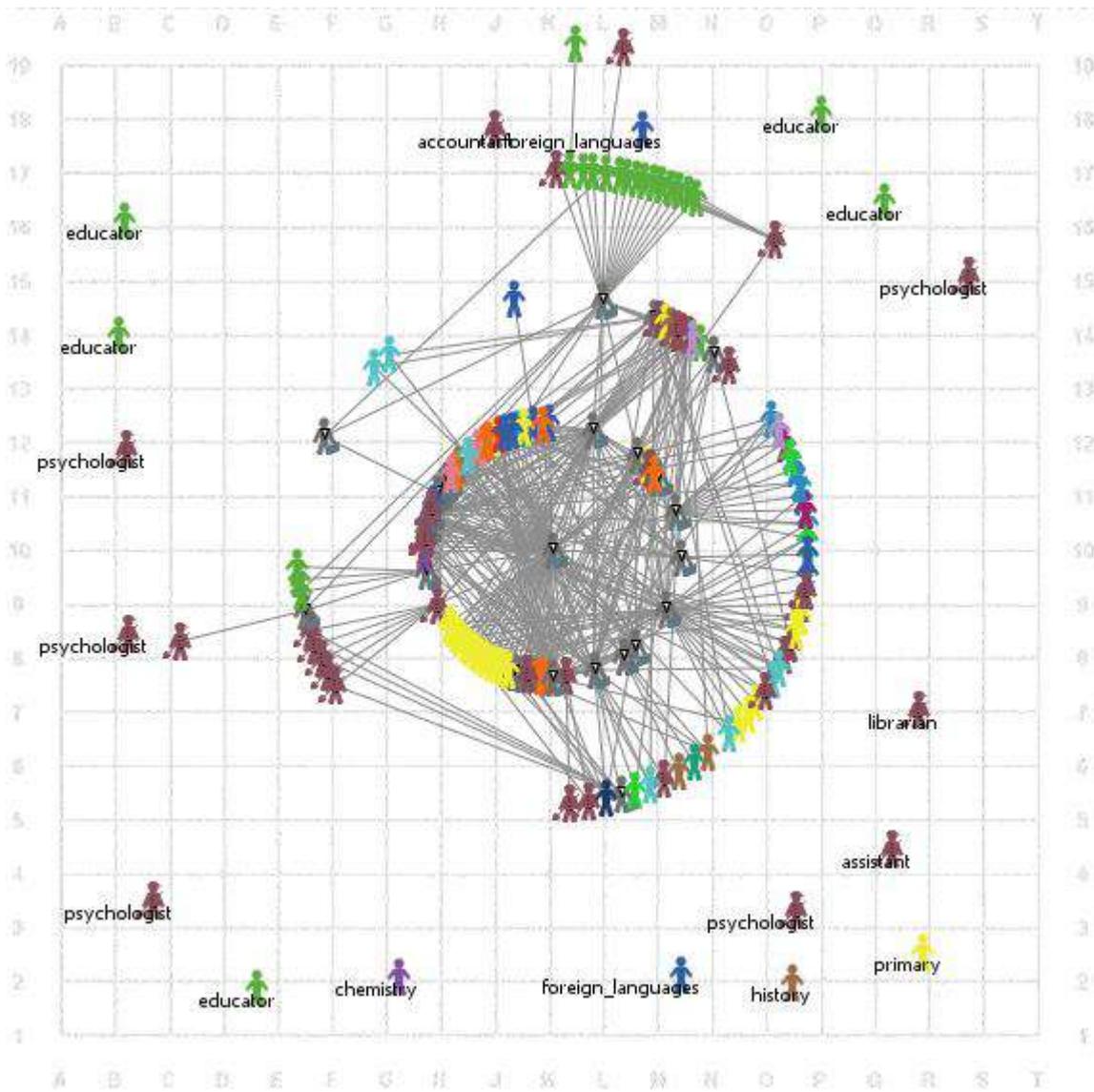


Figure 5. School domain sociogram

Scratch.mit.edu. Since all productive networking systems have common principles, we assume that a similar apparatus may be used for building sociograms based on an analysis of the actions performed by the Scratch community members (scratcher) within studios built for their collaboration. We also believe that the functions of Scratch studios are similar to Media Wiki categories. For experiment purposes, we have gathered data on member activities in a number of Scratch studios and generated collaboration sociograms based on the data. An example of such sociograms is shown in Figure 6. The human shapes refer to members/scratchers, code sheets – projects. A solid line from a scratcher to a project means that the scratcher is the project author. A dotted line means that the scratcher has commented on the project. As we can see in the sociogram, the studio is characterized by average productivity, high connectedness and cohesion, and low sustainability. The removal of the central scratcher and his projects will entail the studio chart disintegration into a large number of unconnected members.

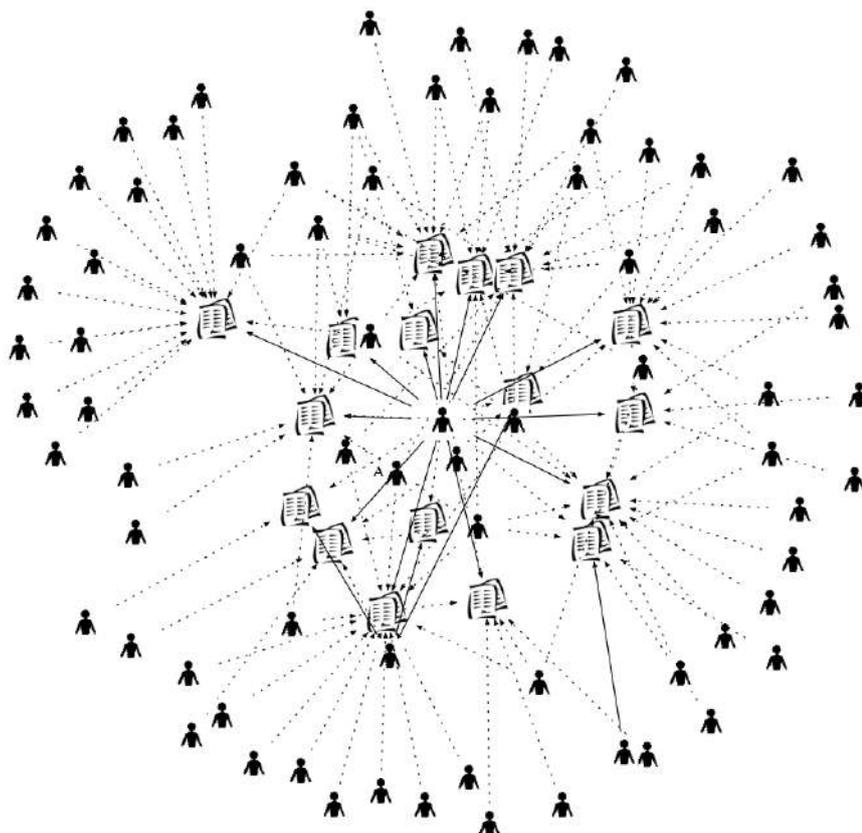


Figure 6. Sociogram of a Scratch studio

Conclusion and Discussion

We believe that socio-educational co-creative projects are not only productive environments but developmental environments also. As John Raven wrote “in developmental environments people can think about their organizations and their society and come to understand and perceive these institutions (and their operation) in new ways that have marked implications for their own behavior” (Raven & Stephenson, 2001). The projected system of the collaborative network activity gives extra possibilities not only for the productive activity but also for the analysis and reflection on the processes inside the system. The tools should open the possibility to assess from the network point of view both the position of each participant and the degree of the whole system development as the learning network. The analysis of the activity of each participant inside the acting community combined by mutual story or mutual game, allows us linking the act of activity and the development of one participant with the development of the whole community. The point of the co-creative projects is not only the created product of the project, but also the social structure itself. The value is in the creation process of this social structure during collaborative work and the formation history of this structure. Usually the social structure and the history of its formation are hidden for the participants. In best cases the subject for participants' discussion is the number of created objects and the number of comments and ratings. Meanwhile a social structure is a significant characteristic in many respects determining the success of collaboration. We believe that visualization of the social structure can support the process of group reflection. Social reflexivity is associated with the social functioning part of a team, deals with interpersonal relation, strengthens collaboration among team members, and therefore leads to better performance. While various studies have found that social reflexivity has significant positive relationship with team outcomes (Gurtner, Tschan, Semmer, & Nägele, 2007; Schippers, Den Hartog, & Koopman, 2007), little is known regarding the mechanisms underlying group social reflexivity in network collaborative projects. We start from the hypothesis that social reflection can be triggered by sociograms and we argue that learning as ‘building knowledge structures’ happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, that can be discussed,

evaluated and reused by other participants to create new entities, and data on the interactions of agents of learning can be presented as a map.

References

- Borgatti, S. P. (2006). Identifying Sets of Key Players in a Social Network. *Comput. Math. Organ. Theory*, 12(1), 21–34. <https://doi.org/10.1007/s10588-006-7084-x>
- Börner, K., Palmer, F., Davis, J. M., Hardy, E., Uzzo, S. M., & Hook, B. J. (2009). Teaching children the structure of science. In *IS&T/SPIE Electronic Imaging* (pp. 724307–724307). International Society for Optics and Photonics. Retrieved from <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1334930>
- Brennan, K., Hernández, A. M., & Resnick, M. (2009). Scratch: creating and sharing interactive media. In *Proceedings of the 9th international conference on Computer supported collaborative learning - Volume 2* (pp. 217–217). International Society of the Learning Sciences. Retrieved from <http://dl.acm.org/citation.cfm?id=1599503.1599576>
- Burov, V., Patarakin, E., & Yarmakhov, B. (2012). A crowdsourcing model for public consultations on draft laws. In *Proceedings of the 6th International Conference on Theory and Practice of Electronic Governance* (pp. 450–451). New York, NY, USA: ACM. <https://doi.org/10.1145/2463728.2463814>
- Engeström, J. (2005). Why some social network services work and others don't — Or: the case for object-centered sociality. Retrieved January 24, 2015, from <http://www.zengestrom.com/blog/2005/04/why-some-social-network-services-work-and-others-dont-or-the-case-for-object-centered-sociality.html>
- Fontana, M., & Terna, P. (2015). *From Agent-based models to network analysis (and return): the policy-making perspective* (Department of Economics and Statistics Cognetti de Martiis. Working Paper No. 201507). University of Turin. Retrieved from <https://ideas.repec.org/p/uto/dipeco/201507.html>
- Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821–7826. <https://doi.org/10.1073/pnas.122653799>
- Gurtner, A., Tschan, F., Semmer, N. K., & Nägele, C. (2007). Getting groups to develop good strategies: Effects of reflexivity interventions on team process, team performance, and shared mental models. *Organizational Behavior and Human Decision Processes*, 102(2), 127–142. <https://doi.org/10.1016/j.obhdp.2006.05.002>
- Kelleher, C., & Pausch, R. (2007). Using Storytelling to Motivate Programming. *Commun. ACM*, 50(7), 58–64. <https://doi.org/10.1145/1272516.1272540>
- Klopfer, E., Scheintaub, H., Huang, W., Wendel, D., & Roque, R. (2009). The Simulation Cycle: Combining Games, Simulations, Engineering and Science Using StarLogo TNG. *E-Learning and Digital Media*, 6(1), 71–96. <https://doi.org/10.2304/elea.2009.6.1.71>
- Lerner, R. M. (2014). *Agent-Based Modeling as a Social Activity*. NORTHWESTERN UNIVERSITY. Retrieved from <http://gradworks.umi.com/36/69/3669272.html>
- Mehra, A., Borgatti, A., Soltis, S., Floyd, T., Halgin, D. S., Brandon, O., & Lopez-Kidwell, V. (2014). Imaginary Worlds: Using Visual Network Scales to Capture Perceptions of Social Networks. In *Contemporary Perspectives on Organizational Social Networks* (Vol. 40, pp. 315–336). Emerald Group Publishing Limited. Retrieved from <http://www.emeraldinsight.com/doi/abs/10.1108/S0733-558X%282014%290000040016>
- Moody, J., Mcfarl, D., & Bender-demoll, S. (2005). Dynamic Network Visualization. *American Journal of Sociology*, 110(4), 1206–1241.
- Papert, S., & Harel, I. (1991). Situating Constructionism. In *Constructionism* (pp. 193–206). Ablex Publishing Corporation. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>

Patarakin, E. D. (2017). Wikigrams-Based Social Inquiry. In *Digital Tools and Solutions for Inquiry-Based STEM Learning* (Vol. 1, pp. 112–138). IGI Global. Retrieved from <http://www.igi-global.com/chapter/wikigrams-based-social-inquiry/180861>

Patarakin, Y., & Shilova, O. (2015). Concept of Learning Design for Collaborative Network Activity. *Procedia - Social and Behavioral Sciences*, 214, 1083–1090. <https://doi.org/10.1016/j.sbspro.2015.11.709>

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., & Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9), 2658–2663. <https://doi.org/10.1073/pnas.0400054101>

Raven, J., & Stephenson, J. (Eds.). (2001). *Competence in the Learning Society*. New York: Peter Lang International Academic Publishers.

Resnick, M. (1997). *Turtles, termites, and traffic jams: explorations in massively parallel microworlds*. MIT Press.

Reynolds, R., & Caperton, I. H. (2009). Comparison of Middle School, High School and Community College Students' Wiki Activity in Globaloria-West Virginia: (Pilot Year-two). In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration* (pp. 29:1–29:2). New York, NY, USA: ACM. <https://doi.org/10.1145/1641309.1641350>

Schippers, M. C., Den Hartog, D. N., & Koopman, P. L. (2007). Reflexivity in Teams: A Measure and Correlates. *Applied Psychology*, 56(2), 189–211. <https://doi.org/10.1111/j.1464-0597.2006.00250.x>

Schwartz, D. L. (1999). The productive agency that drives collaborative learning. In *In P. Dillenbourg (Ed.), Collaborative learning: Cognitive and computational approaches* (pp. 197–218). NY: Elsevier Science/Permagon.

Reconstructing Constructionism by Construal

Nicolas Pope, *nwpope@gmail.com*
University of Turku, Finland

Jonathan Foss, *jonathan.foss@warwick.ac.uk*
University of Warwick, Coventry, CV4 7AL, UK

Meurig Beynon, *wmb@dcs.warwick.ac.uk*
University of Warwick, Coventry, CV4 7AL, UK

Abstract

Noss and Clayson (2015) sets out an agenda for 'reconstructing constructionism', identifying as a major challenge for research: "... to transform constructionism from a framework for action into a set of ways of conceptualising what people do in constructionist environments that can simultaneously assist in designing those environments." This paper relates this challenge to an approach to constructionism based on 'making construals' within the alternative conceptual framework for computing of Empirical Modelling (EM). Making construals is a digital skill based on three fundamental concepts: observables, dependency and agency ("ODA"). Its application will be illustrated using the online Construit environment. Making construals promotes an empirical and experiential perspective on computing that is complementary to 'computational thinking'. Its epistemological roots reflect the deep appreciation of the role of personal experience that characterises the pragmatism of William James and John Dewey.

This paper reviews and illustrates an EM approach to constructionism with respect to six topics: provocative modelling, programming paradigms, objects-to-think-with, definitive programming, Scratch as a 'broader constructionist methodology', and emerging empirical perspectives on computing and learning. The progression of topics relates an EM approach to ideas, tools and practices to which current trends in computer support for constructionism are converging.



Jonathan Foss's construal of the Artiphon: learning music theory through construction

The concluding section considers a central problem, articulated by diSessa and Cobb (2004) and cited by Noss and Clayson, concerning constructionism as a framework for action where theories "are relatively inexplicit, complex, and often involve multiple very diverse elements that cannot plausibly be brought under a single umbrella." It argues that a pragmatic approach to constructionism can dissolve this problem, bringing unity to the plurality of approaches to computing and learning it involves.

Keywords

constructionism; objects-to-think-with; making construal; Empirical Modelling; radical empiricism

Introduction

In their paper "Reconstructing Constructionism", Noss and Clayson (2015) identify a central challenge: "... to transform constructionism from a framework for action into a set of ways of conceptualising what people do in constructionist environments that can simultaneously assist in designing those environments." A key observation informs this challenge: the recognition that constructionism "is as much a theory of epistemology as one of pedagogy [in that] understanding the development of knowledge is part of and integral to the encouragement of an inclusive and powerful pedagogic theory and practice". And if constructionism is conceived "merely as a pedagogic strategy, [it] does not offer in concrete terms much more than a host of other worthy slogans such as 'discovery' ... 'exploratory' ... 'enquiry based' learning".

Computing has had a seminal influence over constructionism. Though constructionist practices may not require computing technology, Papert's original vision for constructionism took inspiration from the potentially transformative impact of the computer on education. This has inevitably meant that the prototype conceptual frameworks for constructionism have been heavily influenced by the semantic framework in which interaction with computers is conceived, that of *computational thinking* (CT) (Wing, 2006). Papert himself recognised the potential conflict between the 'hard' science of computing and the scope for being 'vaguely right' that is characteristic of construction, as he discusses in his chapter on 'Computerists' in *The Children's Machine* (cf. Beynon, 2017).

This paper is one of several publications by Beynon and others (cf. Beynon *et al*, 2016) that attribute the difficulty of meeting Noss and Clayson's central challenge of finding appropriate ways to conceptualise "what people do in constructionist environments" to the compromising influence of CT. The *raison d'être* of CT is to conceptualise the processes by which we construct and exploit objective contexts in which abstract computational activities can be carried out. Computing-in-the-wild is of its nature an activity in which 'construction' in the sense discussed by Latour (2003) plays an essential role. We argue that a conceptualisation of construction can be based on the principles of Empirical Modelling (EM) ("EM Website"). These principles make prominent the role of appropriate personal experiences during the construction activity which enable the formation of relevant knowledge structures, and are general enough to apply both to computing practice in its full generality and to constructionist practices in education. In practical terms they are associated with an activity called 'making construals' that is not necessarily computer-based but thrives on rich technologies for interaction and perceptualisation. In many contexts, computer use is more appropriately regarded as 'making construals' rather than 'programming' as conceptualised in core computer science. This applies, in particular, where the computer is being used to support constructionist practices in education.

There are two aspects to the contribution that EM can make to constructionism. From a conceptual perspective, it introduces three foundational concepts (*observables*, *dependency* and *agency*) which inform both the medium and process of construction and identifies semantic principles to account for these that are rooted in William James's philosophical stance of radical empiricism. From a practical perspective, EM has explored a number of environments for making construals (of which the most recent is the online "Construit" environment) whose designs - in keeping with the terms of Noss and Clayson's challenge - have been incrementally informed by the underlying concepts and principles. The interested reader can find more details of these two topics in papers presented at the Constructionism 2016 conference: (Harfield *et al*, 2016) and (Beynon *et al*, 2016). For convenience, we shall refer to making construals using an environment such as Construit as 'an EM approach' to constructionism.

The responses of experts in the field of constructionism to an EM approach reflect an interest tempered by scepticism. Such experts (cf., in particular, personal communications from James Clayson and Ken Kahn) draw on rich experience of environments for supporting constructionism, of practices in software construction and of paradigms for programming. In their critiques, the merits of the Construit environment, its practical significance and what purport to be its distinctive qualities are questioned. Their call for concrete examples of how Construit differs from other approaches reflects the many different ways in which the environment can be viewed and deployed: as a programming environment to be compared (e.g.) with Jupyter, Scratch/Snap, NetLogo Web, ToonTalk Reborn, App Inventor; as representing a programming paradigm to be compared with (e.g.) prototype-based object-oriented

programming, pure functional programming or a modern Lisp such as Clojure; as an integrated development environment (IDE) to be compared with (e.g.) the Mathematica and iPython notebook IDEs; or as a presentation environment in which natural language and 'code' are blended to be compared with (e.g.) Literate Programming, what Alan Kay calls Active Essays [www.playfulinvention.com] and what Bret Victor calls Explorable Explanations [worrydream.com].

In responding to the feedback from expert commentators, and their injunction to focus on concrete comparisons, it has been important to adopt a holistic viewpoint on how environments support constructionism. The Construit environment is just one component of the EM approach to constructionism: its significance can only be understood in conjunction with the epistemically and experience focused mindset that has informed its design. The contextualising framework has a radical impact on the mode of interpretation of a construal. It is not appropriate to consider a construal simply as a program to serve a specific functional objective. For that reason, to compare construals with pure functional programs is to expose a chasm between two culturally distinct ways of thinking about computer artefacts: one that highlights the concrete and experiential and one that highlights the abstract and computational. In practice, many artefacts that have been developed as environments for developing programs admit interpretations -- or partial interpretations -- similar to those of construals, though they are not explicitly conceived with such interpretations in mind and may be hybrid and incoherent as a result. Well-conceived construals highlight the epistemological merits of domain models based on observables, dependency and agency. A faithful construal of a typical Scratch program reveals the generally chaotic mash-up of abstract computational and concrete experiential ingredients, and its limitations as an epistemic model of an intended behaviour.

The next, and main, section of the paper illustrates some of the above themes with reference to specific examples: it is followed by a concluding section in which one of the key problems confronting theories for constructionism identified in Noss and Clayson (2015) will be discussed.

Construal, construction, comparison and contrast

This section briefly reviews six topics that give useful insight into the nature and potential contribution of making construals. In three of these, construals are compared and contrasted with computer artefacts that address a related subject. The primary emphasis is on modelling as a key "characteristic of a constructionist agenda" identified by Noss and Clayson (2015). Our objective is twofold:

- to highlight different ways in which this modelling for constructionism has been approached, and to illustrate the way in which these approaches, without explicitly spelling out a clear conceptual framework, can be related to and distinguished from an EM approach;
- to assess the degree to which the Construit environment meets the criterion identified by Noss and Clayson (2015): promoting the *learning of powerful ideas* by enabling learners "[to create] external building blocks by a process of building, reflecting and debugging", in such a way that they "can develop relevant internal knowledge structures".

The basic principles and concepts behind making construals will be sketchily and informally introduced as appropriate: for more details of these, see Harfield et al, 2016 and the prototype online course on making construals at <http://jseden.dcs.warwick.ac.uk/construit/?load=344>. Each topic will cite one or more construals that can be further explored via the Constructionism2018 construal in the Construit environment ("The Constructionism2018"). The six topics loosely reflect the evolution of computer-based support for constructionist practices, and serve to illustrate ways in which some of the ideas and techniques behind making construals are, at least implicitly, represented in these practices.

Topic 1: Provocative Modelling

In *Mindstorms* (1980), Papert emphasises many important themes that motivate constructionism. His interest was not in computer programming *per se*, but in how interaction with computing technology could serve to promote 'epistemological reflection' in learners and teachers, and how potentially this might help to 'externalise intuitive expectations' and expose 'the non-obviousness of what we consider obvious'. These aspirations are not well-aligned to the accepted motivations for learning to program, which are typically aimed at realising specific functional objectives rather than provoking reflection on

the nature of human agency. For Papert, the role of Logo programming in provoking learning about a subject domain was very clear, but also limited. It may be that the almost universal use of Logo-like programming constructs in subsequent environments to support constructionism is an aberration in this respect – an inappropriate attempt to generalise Papert's principles as they apply to a specific learning objective as if they were universally applicable.

A comparative study may be helpful in this context (see ("The Constructionism2018")). This concerns modelling of a room layout such as might appear on an architect's sketch. The layout is a 2d line-drawing floorplan featuring lines to represent the walls, door, a filing cabinet and a table with a lamp upon it connected by a cable to one of several possible sockets. The study contrasts a most elementary application of basic Logo to draw the components of the layout with an equally unsophisticated application of the Construit environment to construct an image that is visually indistinguishable.

Drawing the floorplan of the room in Logo has an educational significance that is non-obvious to the accomplished programmer. To appreciate it, it is necessary to consider how a child might learn to make a map in a concrete situation, as when recording the findings of an archaeological dig. The relevant epistemological ingredients are precisely concerned with the orientation and length of lines, and how these might be exposed through projection and informed by intuitive expectations of what form the floorplan of a buried room might take. The provocation to learn is concerned with how the child's experience of orienting, moving and counting paces can be connected with an appreciation of entities and coordinates in a 2d space (cf. the Experiential Linear Algebra construal ("The Constructionism2018")).

It is quite apparent that such a primitive form of Logo is ill-suited to provoking reflection about the more sophisticated concerns that might arise in interpreting such an archaeological find - or in other situations motivating the 2d sketch. This raises the question of what kind of extensions to the Logo programming language, alternative choice of language, techniques in the program development or changes to the programming paradigm would be appropriate in addressing a broader agenda. It is in this connection that making a construal of the room layout is of interest.

The construal of the room is loosely determined in an open-ended fashion by an explicit set of current definitions ("a script") and an implicit cloud of potential interactions. Each definition expresses the current value of an observable: these correspond to entities that might be directly registered by inspecting an actual room (or conceiving an imaginary one). The north-west corner of the room, the east wall, the door and the door status, whether open or closed are simple examples of observables in the script. Observables may include entities that (for diverse reasons) are not directly visible, such as whether the door is locked, whether the electricity is switched off, whether an alarm is sounding. The criterion for observability is a personal subjective one: it differs according to who is the observer or maker and what agency they have to interact.

A definition may attach an explicit current value to an observable, as in "the north west corner is at the location [100,100]" or specify it by a 'spreadsheet-style' dependency, as in "the north east corner of the room is at the location [100+room_width, 100]". As another example: "the orientation of the hinged door depends on whether or not it is open". Dependencies are always associated with some latent agency (cf. the 'What if?' aspect of spreadsheet interaction): as in "What if the room was wider?" or "Let's shut the door". Though the construal of the room is visually quite primitive, as a casual sketch might be, interaction with the construal through redefinition is unconstrained, and meaningful in so far as it respects connections being made in the experience of the maker. "At what point is a door too wide or too narrow to be a door?" Interaction with the construal is through redefining observables; this encompasses commonsense room actions (such as "door_open = not door_open") and more complicated or subtle semantic changes, such as introducing a sliding door, or refining the model of the door so that it's orientation reflects 'how open' it is (and redefining the observable 'door_open' accordingly).

In the simple illustrative example script, the only agency is that of the human observer, who can make redefinitions in the role of (e.g.) a room user, room designer or expositor of Empirical Modelling principles. With each role, there is an associated tacit context that shapes what interactions by way of redefinition are appropriate, or plausible. This is a discretionary context pragmatically determined by

what connections in experience make sense in that role: a room user would not ordinarily move a wall for instance. It is possible to introduce autonomous agency, but even where such agency is added the construal always conceptually corresponds to a state rather than a behaviour. As a simple example: the maker of the construal may add an automatic closing agent to the door, but still conceives the script as representing the current state of the room, where both script and room state are dynamically changing. This admits the possibility of intervention in automated behaviour without undermining the model of current state. In commonsense terms, this reflects the possibility that, even though the door closes automatically, you can still obstruct it.

Juxtaposing the room construal with a primitive Logo program to draw the room layout highlights the challenge in promoting Papert's vision for "writing programs as promoting epistemological reflection". The construal exposes the richness and subtlety of the experience and interpretation that surrounds potential conversations around the room layout. In particular, it reveals the extraordinary capacity of the human mind to make connections in experience of diverse kinds in a flexible and fluid manner: connecting a simple line drawing with an image of a physical room, real or imagined; attributing the relocation of the NE corner of the layout to a change in the value of 'room_width'; conceiving the current state of the layout as part of 'a behaviour'. The aspiration to strengthen the semantic connection between 'writing a program' and 'understanding the subject domain in which it executes' has been a leitmotif of software development that has informed the design of PLs to support constructionism. This helps to motivate the next topic: consideration of a declarative programming paradigm.

Topic 2: Programming Paradigms

This study examines the relationship between a model that is described in a declarative rather than a procedural fashion and a construal to address a similar theme, viz. modelling 'playing the game of noughts-and-crosses'. The declarative program in this case is written in the 'pure functional' Miranda PL, as formerly deployed in teaching first year university computer science at the University of Warwick. As the previous study illustrates, a procedural program can generate a bewildering complex of states that have no semantic significance as far as the subject domain is concerned. The rationale for declarative PLs was to eliminate such states so as to make a more intimate connection between the program text and the subject domain. This ambition is clearly well-aligned in principle to the aspirations of epistemic modelling. The contrast in this case is with a construal that identifies the observables that are involved in playing a game of noughts-and-crosses (*What is the state of the grid? What are the winning lines? How are the symbols displayed on the grid? Whose turn is to play? When is the game over? What is a winning move?* etc) and expresses the dependency relationships between these together (if automated play is desired) with agents to make moves autonomously.

It is indeed the case that constructs in the Miranda program have direct counterparts in a noughts-and-crosses game. Since the declarative framework precludes modelling the explicit states of a game move by move, there are representations for such things as "the sequence of positions that arise in the game" - but no observable to represent what can be deemed in the construal to be the current position in the game. By contrast, the construal can readily be exercised in such a way as to simulate mistakes in play (such as a player missing a turn), cheating (such as a piece being removed from the board), changes in the winning lines (even in the course of play). In these respects, the construal (aka 'the OXO laboratory') has a better claim to the character that Papert attributes to "a programming language" (Mindstorms, 1980, p15): "[being] like a natural, human language in that it favors certain metaphors, images, and ways of thinking."

As construals dating from several years ago, both the room construal and the OXO laboratory illustrate a problematic issue in learning to make construals. In framing dependencies, it is in general necessary to express relationships of a relatively complex form. For instance: "The player to move is X if X makes the first move and the number of Xs on the grid is the same as the number of Os and O otherwise". A definition of this nature is then framed by writing a small conventional program (in a basic procedural programming notation embedded in Construit) to compute the number of Xs and Os on the grid from the current position. Since each definition in the script expresses the value of an observable as a pure, and generally simple, function of other observables, the (declarative) programming task is much more

straightforward both technically and semantically than developing a procedural program in which there is complex agency and state change, but the hybrid nature of the activity presents a barrier to the maker.

Topic 3: Objects-to-think-with

Towards the end of *Mindstorms*, Papert acknowledges the limitations of Logo and refines his position: "I see [the Turtle] as a valuable educational object, but its principal role here is to serve as a model for other objects, yet to be invented. My interest is in the process of invention of "objects-to-think-with," objects in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification." A significant point for emphasis here is that Papert is thinking in concrete experiential terms about what the computer offers to the learner: *the Turtle* is the object-to-think-with, and not the abstract Logo program. This is not a mode of thinking about computer programs that has foundational support from theoretical computer science, though it has developed considerable pragmatic endorsement from software practice. By way of illustration, object-oriented approaches to software development, which were once regarded as a promising candidate foundational framework for software development, have been undermined by a (sometimes bitter) controversy between those who champion object-orientation as a formal approach and those who declare it to be the basis of an empirical pragmatic framework in which the emphasis is on how objects are crafted in accordance with how they are experienced by developers (cf. (West, 2004)). To highlight the distinction that is being made here, the term 'computer artefact' will be used to refer to what computer programs offer in experiential terms: 'objects-to-think-with' are to be classified as artefacts, not programs.

It seems difficult to identify an iconic example of an object-to-think-with that is implemented as a computer artefact. In *Mindstorms*, Papert (1980) describes such an object in connection with a Penny Rolling conundrum posed by Martin Gardner, but no implementation is described. In any event, realising such an object on the computer is a task beyond the range of a novice programmer. Microworlds can be viewed as preconstructed environments in which learners can learn about a subject domain through experiment and similar claims can be made for environments to support educational robotics. Such environments invite the "worthy slogans ... of 'discovery ... exploratory ... enquiry based' learning", but their constructionist credentials perhaps depend on the extent to which learners can exploit them to make connections that are beyond the scope of what has been preconceived by their designer.

Of all the established approaches to creating objects-to-think-with, creating a spreadsheet might be seen as most promising, as it embodies the principles of 'what if?' modelling. Though spreadsheets have been advocated in education (cf. the *Spreadsheets in Education* journal) there seems to be little reference to their application in constructionist practices. Construals, to which spreadsheets are semantically the closest well-known relatives, nonetheless seem well-suited to specifying objects-to-think-with. Examples include: construals of objects-to-think-with relating to the penny rolling puzzle (including Papert's own proposal (Beynon,2017)); a construal of a purse; a number representation construal that gives visual expression to a mental model of how integers are represented in different bases. An important feature of these construals (all of which are available online in the Construit archive – see ("The Constructionism2018")) is that they are constructed without the explicit use of conventional procedural or declarative programming techniques, a theme to be elaborated in the following sections.

Topic 4: Definitive Programming

The aspiration to exploit the computer fully in constructionist practices inevitably means that much attention has been given to how to make *writing computer programs* more accessible. Enabling computing novices to experiment with prebuilt objects and environments does not offer the same potential for learning in a constructionist idiom that enabling them to construct their own computer artefacts does. What is more, it has long seemed that the kind of procedural thinking that is exemplified by Papert's Turtle is an essential component in constructing any computer artefact, and is also one to which novices can most easily relate. This has meant that helping novices to engage with procedural thinking has been a prominent characteristic of environments to support constructionism, whether through introducing blocks, as in Scratch and App Inventor, or more semantically motivated animation as in ToonTalk.

The effect of ‘writing a program to meet a specific functional requirement’ can be emulated in Construit by crafting the agency that is to be automated and invoking this in a context where the maker’s interaction is constrained to be that of the intended user. This activity is closely parallel to programming behaviour in the room construal by restricting redefinitions to plausible actions (such as ‘opening the door’ or ‘moving the table’ by the room user, and automated ‘closing of the door’ by the door mechanism). The discretionary context is one in which actions that redesign the room (e.g. ‘changing the width of the room’) are suppressed, but the option to reinstate them is open at any stage – even whilst programmed actions are being executed. In this way, program-like behaviours expressed using a construal can be seamlessly blended with actions that might be construed as re-programming.

By way of illustration, consider the Whack-a-Mole application that is featured as an introductory programming exercise in App Inventor. By modelling the application in terms of observables, dependencies and agency, a very concise script can be framed to express the way in which the mechanism that moves the mole and the player’s mouse actions interact. In framing this script, dependency plays a crucial role in ‘taming’ the procedural ingredients of the application. As the maker of the Whack-a-Mole construal, Jonathan Foss, remarks: the main feature of the development was “the directness of the experience. Rather than having to ‘plan’/‘design’ and ‘program’, it was more a case of drawing some holes and a mole, then adding functionality by describing how these things are connected to each other ... compared with App Inventor, there was a lot less abstract thought/planning required, with all statements having a direct connection to the scenario that was being modelled.” Further informal evidence of the qualities of the construal was obtained from a Key Stage 3 pupil at a schools event. In the space of 20 minutes, with no previous experience of Construit or making construals, and with only the most basic indication about the nature of the script, he was able to add more holes, introduce an extra mole, and make the speed at which the mole moved dependent on the score.

A significant feature of Foss’s Whack-a-Mole construal is the absence of any explicit procedural programming ingredients. This is a characteristic of many more recent construals, which exploit an innovative feature of the Construit environment that has transformed the practice of making construals: the introduction of the **with**-construct. Informally, the significance of **with** is that it makes it possible to frame succinct definitions to express new categories of observables that correspond to commonplace modes of observation. For instance, the array of holes in Whack-a-Mole is defined by defining a prototype hole with index *i* and specifying the entire as a collection of holes, based on this prototype, with indices ranging from 1 to the number of holes. More generally, whenever an existing observable is defined by dependency in terms of other observables (its ‘dependees’), a new observable can be specified as the counterpart of this observable with a value defined *as if* specific changes to the valued to its dependees had been made. Through this enhancement, the expressive power of scripts is transformed. It becomes possible to refer to entities that routinely form part of commonsense observation (such as ‘the height of the tallest building on the skyline’) without needing to introduce obfuscating procedural constructs. The full implications of this are to realise a vision that has been a long-standing aspiration in EM research: an observation-oriented approach to programming that is based solely on definition and redefinition (aka “definitive programming”).

Topic 5: Scratch and a ‘broader constructionism methodology’

Noss and Clayson (2015) identifies Scratch as having “shown signs of contributing to the creation of a broader constructionist methodology”, as suggested by Brennan & Resnick (2013). There is empirical evidence for this in the millions of learners, some collaborating with each other, who have used the online Scratch environment to create a vast archive of projects. Such a cultural phenomenon opens up unprecedented scope for ‘construction’ leading to the development of objective insights from subjective personal understandings. Scratch has patently been the stimulus for many people to begin to learn to program. What is not so clear is to what extent the Scratch programming revolution illustrates the central principle of constructionist practice: understanding of the subject domain informs the construction of a computer artefact and interaction with the computer artefact informs understanding of the subject domain.

With its visual blocks interface, sprites and costumes, stage and backdrops, Scratch makes a significant move towards ‘constructing a computer artefact’ rather than ‘writing an abstract program’. It also

incorporates several features that resonate with an EM approach to construction. Agents act concurrently; built-in dependencies (e.g. to simplify networking and so promote collaboration) give learners easy access to high-level agency; sophisticated examples of observables and agency (such as enable sprites to detect whether they intersect a line of a certain colour, or to bounce at the edge of the stage) are built-in as primitives. This makes it easier for the learner to program and encourages them to explore basic concepts in computational thinking.

The impact of introducing all these features on program comprehension is more problematic. The benefit of block-based programming has itself been questioned (Hermans and Aivalogou, 2016). From a constructionist viewpoint, as shown in the room modelling exercise, procedural ingredients need to be handled with care. Without principles to guide program development, having a wide choice of possible representations may not be advantageous: If I wish to create a vertical line from the top to the bottom of the stage, should I draw it using a Logo like command? should I ensure that it has the full height of the stage by using the bounce feature? or should I simply draw it on the backdrop? How do I subsequently move the line? How can I determine the explicit coordinates of its endpoints? Building-in instances of powerful dependency and agency to support key activities is helpful to a degree, but the merits of an EM approach stem from the automatic support for *user-defined* dependency to enable simpler and more intelligible specification of agency. Such considerations make it hard to attribute clear 'epistemic modelling' content to casually constructed Scratch programs and places the somewhat self-referential claim that Scratch supports constructionist practices that include 'the construction of computer programs' in a different light.

Many of the above concerns are potentially just as topical for the Construit environment. It too offers many alternative modes of representation and techniques for development. It is quite possible to write traditional procedural programs within Construit, and there is even provision for incorporating JavaScript. Until the advent of the **with**-construct, some procedural coding element was essential in creating all but the simplest construals. In order to respect the principles of ODA modelling, such coding could then only be exploited to define the (pure – and simple) functions that expressed non-trivial dependencies and the elementary procedures to effect the redefinition associated with automated agents. In this respect, the absence of guiding principles to inhibit the *ad hoc* construction of Scratch programs seems problematic.

The issues discussed in this section are illustrated by the Giving Change construal, developed to complement a Scratch program called 'Coins' developed by Phil Bagge for use in teaching programming at primary school. For details, see <http://code-it.co.uk/scratch/coins/coinsoverview> and <http://jseden.dcs.warwick.ac.uk/construit/?load=138>. The construal addresses the problem of converting a sum of money into a set of coins – a topic that is recognised to be challenging to teach young children because of the complexity of the skills that are involved. Bagge's program is primarily intended to illustrate basic techniques in Scratch programming: the construal raises issues that bear directly on pedagogical strategies, and exemplifies the kind of exposure of epistemological questions that Papert advocates. The most important realisation in making the construal was that in order to express the agency of giving change in 'definitive programming' terms, it was essential to consider what modes of observation could be assumed to be plausible for the change-giving agent. For instance, if the agent is an automated AI, it is perfectly reasonable to presume that it can 'inspect' a set of coins and select the one with largest value. By contrast, for a young child to make the same selection, they have first to master several observational challenges, such as assessing whether the value of one coin is greater than that of another irrespective of their relative size and having regard to their colour (bronze, silver or gold). A most interesting feature of the construal is that it can be framed in such a way that the pattern of observation that serves to realise the required behaviour is exactly matched to the pattern of observation that might be used to teach a child to give change. In effect, the definitive program sets out a sequence of observational challenges of the form "*can you recognise from the colour of these coins which has the greater value?*", "*can you identify whether this coin is worth more than 32 pence?*", "*can you identify which coin in this set has the largest value?*" Carry out these same challenges with heads and tails reversed. etc. This intimate relationship between recipes for programming the computer and learning strategies is precisely in tune with Papert's intentions in advocating the use of the computer to support learning.

Topic 6: Empirical perspectives on computing and learning

Over recent years, there has been strong interest in devising systematic empirical ways of using the computer in learning applications, to teach programming, mathematics and other subjects. Bret Victor has been prominent in this: the Learnable Programming, Kill Maths, Exploratory Explanations projects share a common theme: exploiting the power of the computer to offer rich interactive experiences to address learning goals that have generally been treated in the classroom in more abstract mathematical terms. These projects promote computer artefacts as objects-to-think-with and to-converse-with, and Victor explicitly acknowledges the inspiration of Papert's constructionist vision.

Current trends have giving great impetus to the idea of 'coding', but also reflect dissatisfaction with the way in which programming is conceived. Chris Granger's blog post: 'Coding is not the new literacy' echoes Papert's emphasis on the epistemological implications of using the computer: "We need the equivalent of composition, the skill that allows us to think about how things are computed. This time, we're not recording our thoughts, but instead the models of the world that allow us to have thoughts in the first place." Granger goes on to identify 'modelling with spreadsheets' as the most pertinent computing skill. Novel environments that have emerged also have features in common with Construit. In keeping with the desire to integrate construction and interaction with an artefact with learning about the domain, these typically include support for notebook style presentations, where natural language descriptions and code are blended. What is generally lacking in such innovations is a clear conceptual foundation. For instance, though observables, dependencies and agency feature quite explicitly in the Apparatus Editor, and enjoy support through a menu-driven interface, it is unclear how general the scope for defining dependencies is intended to be or to what extent the broader semantic issues addressed in EM have been taken into account.

In comparing an EM approach to constructionism with other approaches, certain characteristics stand out:

- In developing a construal, there is semantic significance in every redefinition: if this is not the case, the principles of making construals are not being properly respected. Redefinition by redefinition, the thought processes of the maker are being meticulously registered by the maker, even if subconsciously, and recorded in the script. In part, the fine detail in which thought processes are being traced accounts for the aura of naivety that surrounds the representation in a script. Specification via declarative constraints or by very sophisticated pure functional expressions in an FP idiom might appear to be much more powerful, but cannot serve the same expressive role. Whether the maker can understand the effect of dependencies by inspection of the script, or by experimental redefinition, are important pragmatic considerations. A formal definition may be so complex that the maker is unable to recognise that there *is* a dependency.
- The fact that the script invariably bears interpretation in state-based rather than exclusively behavioural terms, and has an essential ambiguity that allows it to relate simultaneously to different agent roles (such as designer and user) is another crucial characteristic.

These qualities have implications for learning through both construction and interaction.

Nicolas Pope's construal of an internal combustion engine, developed for a science festival workshop for Key Stage 3 pupils, highlights the virtues of being able to blend automated action to a preconceived pattern with opportunistic experimental intervention by the learner or teacher. The construal supports a nuanced explanation of the principle behind the internal combustion engine where the learner can emulate the agency of the engine and develop an understanding of how it operates. It features a 'drinking straw' engine where the learner simulates the effect of internal combustion in maintaining the rotation of the wheel by blowing and sucking at appropriate points as it revolves. By emulating the behaviour of the engine in this fashion, it is possible to account for the need for a starter motor (or a crank handle!) and to illustrate the significance of issues such as timing, a blown gasket, possible causes of misfiring, and the principles of engine braking. In effect, making such a construal opens up the possibility of exploring the meaning of observables that are not ordinarily exposed in a conventional non-interactive simulation. Many of these were not preconceived in the evolutionary development of the construal.

Jonathan Foss's construal of a newly developed electronic musical instrument, the Artiphon, illustrates the manner in which Construit can support design and documentation. In his own words: "Making the Artiphon construal entailed a large musical learning experience. If I'd have built the program conventionally, I feel (although can't necessarily prove), that traditional programming languages would have constrained my thoughts, I'd have needed to do a lot more thinking about the musical theory before actually doing anything with it. However, with Construit, I could just start by drawing what I could see, and then expressing how the physical observables interacted with each other. The style of the 'book' layout allowed the code building to be much more conversational than comments would normally be when writing code. The individual statements that were written to guide the construction process were much simpler than traditional programming - particularly the fact that definitions can be overridden (rather than having to change a particular line of code in place), meant that it was easier to log (e.g. a simple form of version control) and gain an understanding of faults."

Prior to making the Artiphon construal, Foss had little knowledge of music theory. Reflecting on his experience of learning music theory by constructing the construal, he observes:

"I think the strength of EM is that things can be built without a clear idea of the subject domain. ... In other programming languages, you'd have to already know lots about the subject domain and then translate it into code that is rather abstracted away from the actual subject domain. The learning process becomes more about learning how to code a representation of the subject domain. ... Whereas with a construal, the directness of experience means you can start with a basic understanding of the subject domain and fine tune it as you learn. You are thus spending much more time thinking about the subject domain than about how to translate that subject domain knowledge into code."

As an experienced software developer, Foss also reflected on the significance of version management in Construit, whereby the new versions are derived from the old (and *vice versa*) by overriding definitions. Contrasting this activity with "rewriting code then committing to a version control system", he remarks: "I suspect version control for most casual programmers is an advanced way of saving/backing up. Whereas for construal making, overriding definitions provides a much richer experience that can be easily 'reviewed'."

Other example construals illustrate qualities of making construals relevant to issues raised by Noss and Clayson (2015): editing and extending (cf. the 'Construing the Moment' and 'Solar System' construals), layering (cf. Hex Colouring), exploratory design (cf. The OXO lab) – see ("Constructionism2018") for more details. Further discussion is beyond the scope of this paper.

Conclusion and Discussion

The introduction to this paper refers to making construals 'being rooted in William James's philosophical stance of radical empiricism'. The topics discussed in the previous section make a clear justification for considering making construals within its broader *conceptual* framework. The relevance of a *philosophical* stance is not so easily appreciated.

In formulating the central challenge that has informed this paper, Noss and Clayson follow diSessa and Cobb (2004:82), who characterise constructionism as a 'framework for action'. They also cite diSessa and Cobb's comment to the effect that frameworks for action typically "do not cleanly separate their scientific claims and validation from their suggested actions. That is, the theory or theories behind frameworks for action are relatively inexplicit, complex, and often involve multiple very diverse elements that cannot plausibly be brought under a single umbrella."

Any theory for constructionism has to confront the problem of dealing with the learner's subjective world. To speak of the learner's subjective experience is to consider matters about which it is scarcely appropriate to rationalise. There is no constraint on the irrationality of a learner's misconceptions, no guarantee that they will bear scrutiny even by the learner themselves, or that they will be consistent across changing contexts. Indeed, even referring to the 'learner's misconception' betrays a perspective that might be defensible when learning mathematics, but is more questionable when applied (say) to musical interpretation.

William James's radical empiricist stance addresses the problem of discussing the subjective world in a principled way. Its focus is upon the connections that the learner directly experiences, such as come into play when we understand a word, recognise a person, or attribute an effect to a cause. The semantics of a construal as a computer artefact are mediated by such connections: the principle is similar to what Noss and Hoyles (1996) call 'situated abstraction'. In reviewing the significance of situated abstraction in constructionism, Mackrell and Pratt (2017) observe: 'Situated abstraction is ... about connecting: learning is about increasingly knowing connections. However, the nature of such connections is left unexamined, which is problematic; it is tempting to see a situated abstraction as an inferential proposition drawn on the basis of what is known about a context and what this would entail in action, but this is not yet warranted.'

The key phrase in the above quotation is "However, the nature of such connections is left unexamined, which is problematic ... "; this betrays a philosophical disposition to seek a rational explanation for situated abstraction that is at odds with a fundamental principle of radical empiricism (James, 1909): "the relations between things, conjunctive as well as disjunctive, are just as much matters of experience, neither more nor less so, than the things themselves". A similar philosophical sentiment is implicit in diSessa and Cobb's critique of frameworks for action that "do not cleanly separate their scientific claims and validation from their suggested actions". The EM approach to constructionism advocated in this paper endorses a more pragmatic perspective in which we accept that the authenticity of personal connections in experience is not amenable to validation in the strict objective sense that is expected in science; it can only be to a degree witnessed through performance in context. What is more, it is the very notion of trying to impose such traditional 'theories' upon constructionism that accounts for their 'relatively inexplicit, complex' nature and the implausibility of bringing their 'multiple very diverse elements ... under a single umbrella'. It is in this respect that we hope that making construals can bring unity and coherence to the plurality of approaches to computing and learning associated with constructionism.

References

The Apparatus Editor. Online at <http://aprt.us/>

Beynon, M. (2012) Modelling with experience: construal and construction for software, Chapter 9 in *Ways of Thinking, Ways of Seeing* (ed. Chris Bissell and Chris Dillon), Automation, Collaboration, & E-Services Series 1, Springer-Verlag, January 2012, p.197-228

Beynon, M. (2017) Mindstorms Revisited: Making New Construals of Seymour Papert's Legacy, in Alimisi D., Moro M., Menegatti E. (eds) *Educational Robotics in the Makers Era. Edurobotics 2016*. Advances in Intelligent Systems & Computing, vol. 560. Springer, Cham, p.3-19

Beynon, M., Foss, J., Harfield, A., Hudnott, E. and Pope, N. (2016) Construing and Computing: Learning through Exploring and Exploiting Agency. In Proceedings: *Constructionism in Action 2016*, February 1-5, Bangkok, Thailand. Bangkok: Suksapattana Foundation. p.69-78.

Brennan K. & Resnick M. (2013) Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. In: Lavigne N. & Mouza C. (eds.) *Emerging technologies for the classroom: A learning sciences perspective*. Springer, New York: p.253–268.

Clayson, J. (2017) personal email communication, 28th April 2017

The Construit environment for making construals: [Error! Hyperlink reference not valid.](#)

The Constructionism2018 construal: <http://jseden.dcs.warwick.ac.uk/construit/?load=362>

diSessa A. A. & Cobb P. (2004) Ontological innovation and the role of theory in design experiments. *Journal of the Learning Sciences* 13(1): p.77–103.

The EM website at go.warwick.ac.uk/em

Harfield, A., Alimisi, R., Tomcsanyi, P., Pope, N. and Beynon, M. (2016) Constructionism as making construals: first steps with JS-Eden in the classroom. In Proceedings: *Constructionism in Action 2016*, February 1-5, Bangkok, Thailand. Bangkok: Suksapattana Foundation. p.42-52

Hermans, F, and Aivalogou, E, (2016) Do code smells hamper novice programming? A controlled experiment on Scratch programs. In *Proceedings 2016 IEEE 24th International Conference on Program Comprehension (ICPC)*

James, W. (1909) Preface to *The Meaning of Truth*. <https://www.gutenberg.org/files/5117/5117-h/5117-h.htm>

Kahn, Ken (2017) personal email communication, May 2017

Latour, B. (2003) The Promises of Constructivism, In Don Ihde and Evan Selinger (Eds.), *Chasing Technoscience: Matrix for Materiality*, Indiana University Press, p27-46

Mackrell, K and Pratt, D. (2017) Constructionism and the space of reasons, *Math. Ed. Res. J.* DOI 10.1007/s13394-017-0194-6

Noss, R. and Clayson, J. (2015) Reconstructing Constructionism. <http://www.univie.ac.at/constructivism/journal/10/3/285.noss>

Noss, R. and Hoyles, C. (1996) *Windows on mathematical meanings: Learning cultures and computers*. Kluwer, Dordrecht.

Papert, S. (1980) *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York.

Spreadsheets in Education, Bond University, School of IT, <http://epublications.bond.edu.au/ejsie/>

West, D. *Object Thinking*. (2004) Microsoft Press

Wing, J. M. (2006) Computational thinking. *CACM* Vol. 49:3, March 2006, p.33-35

Constructionist STEM Activities Using the Bridge21 Model

Brendan Tangney, *tangney@tcd.ie*

The Trinity Centre for Research in IT in Education, School of Computer Science & Statistics and School of Education, Trinity College Dublin, the University of Dublin, Dublin 2, Ireland

Ian Boran, *ian.boran@tallaghtcs.ie*

Tallaght Community School, Dublin 24, Ireland

Tony Knox, *tonyknox@kilkennycollege.ie*

Kilkenny College, Kilkenny, Ireland

Aibhín Bray, *brayai@tcd.ie*

The Trinity Centre for Research in IT in Education, School of Computer Science & Statistics and School of Education, Trinity College Dublin, the University of Dublin, Dublin 2, Ireland

Abstract

This paper explores how the use of a particular model of 21st Century teaching and learning (Bridge21), which aligns easily with a constructionist approach, can be integrated into an authentic classroom context, and the extent to which this leads to increases in students' metacognitive problem-solving skills as well as their engagement and confidence with STEM disciplines. The research is framed as an exploratory case study with two embedded units: the first focused on developing problem-solving skills in a group of 21 students (15-16 years old) in an after-school STEM club, using technologies that included a microworld for the construction of bridges; the second focused on the teaching of mathematical functions using a graphics calculator, with a view to improving student confidence and level of mathematical engagement, in a class of 24 students (15-17 year old) of mixed gender and weak to modest mathematical ability. Statistically significant improvements in students' metacognitive problem solving skills are reported from Intervention A, while Intervention B reports statistically significant improvements in students' attitude to mathematics and technology.

Keywords

STEM; 21st century teaching and learning; bridge21; constructionism

Introduction

Education systems around the world face a challenge in empowering teachers to create learning activities that foster the development of so called "21st century skills", while at the same time helping students master curriculum content (Danah, Punya, & Petra, 2016; Fullan & Langworthy, 2014; Voogt & Roblin, 2012). This issue is particularly pertinent in the area of STEM education, where skills such as problem solving, communication and collaboration are fundamental to successful practice in associated disciplines. However, internationally, the number of students considering careers in STEM-related areas is declining (Marginson, Tytler, Freeman & Roberts, 2013). The challenges that exist in STEM education, which are at least partially responsible for this fall off, are well documented and include: the use of didactic teaching styles, the restrictive nature of traditional classroom environments, low student motivation, overloaded curriculum content, over-reliance on text books, a lack of discussion of topics of interest, and the absence of opportunity for creative expression (Henriksen, Dillon, & Ryder, 2015).

In this paper, we explore how the use of a particular model of 21st Century teaching and learning (Bridge21), which aligns easily with a constructionist approach, can be integrated into an authentic classroom context, and the extent to which this leads to increases in students' metacognitive problem-solving skills as well as their engagement and confidence with STEM disciplines.

In order to set this research in context, this paper first presents some background literature. This is followed by an overview of the study, which includes a description of the local context as well as a section that provides a rationale for the methodological choices that were made. Two separate interventions and their findings are then discussed, and finally the concluding section discusses how the constructionist, 21st Century approach, scaffolded by the Bridge21 activity model, has addressed some of the challenges in STEM education through increasing student motivation, engagement and problem-solving abilities.

Background

Innovative approaches to STEM education such as Realistic Maths Education (Gravemeijer, Rainero, & Vonk, 1994), Project-Based Learning (Capraro, Capraro, & Morgan, 2013) and Inquiry-Based Learning (Edelson, Gordin, & Pea, 1999; Maaß & Artigue, 2013), which are promoted as ways to address some of the challenges in area, are also well suited to supporting the development of 21st century skills, especially when making creative use of technology. Constructionism, with its emphasis on the building of artefacts as part of the knowledge construction process, is an ideal pedagogy to use as part of such innovative approaches to 21st century STEM education. However, neither a constructionist approach, nor the pedagogies listed above, lend themselves well to traditional classrooms, which tend to embrace didactic teaching and learning methodologies (Fullan & Langworthy, 2014; Tangney, Bray, & Oldham, 2015).

In previous work Tangney et al. (2015) argued that innovative, technology-mediated learning activities need to be embedded in an appropriate, or sympathetic, pedagogical model rather than being accommodated and constrained within a traditional classroom approach. Activities of this type would lie at the higher, or *transformation* layers, of the SAMR hierarchy for classifying technological-based learning interventions (Puentedura, 2012), as they are tasks that are significantly changed through the use of the technology (*modification*), or that use the affordances of the technology to design new tasks that would previously have been inconceivable (*redefinition*). Bridge21 is a particular model of technology-mediated, collaborative learning which promotes teamwork (Lawlor, Conneely, Oldham, Marshall, & Tangney, 2018), improves student motivation (Lawlor, Marshall, & Tangney, 2015), and helps develop key skills (Johnston, Conneely, Murchan, & Tangney, 2015). The Bridge21 approach has been used to support innovative learning activities across a range of subject areas including mathematics (Bray & Tangney, 2015) and computer programming (Byrne, O'Sullivan, & Sullivan, 2016). Of particular relevance to this paper is a study that reported on the use of Bridge21 to support a constructionist approach to teaching physics (Wickham, Girvan, & Tangney, 2016). However, the aforementioned studies were, for the most part, carried out in a learning laboratory on the university campus and not in an authentic school context.

The Study

In order to explore whether constructionist activities that are scaffolded by the Bridge21 model can be successfully integrated into authentic classroom contexts, and whether such activities have a positive effect on student confidence, engagement and problem solving, this paper reports on the experiences of two teachers who designed and researched transformative learning activities for use in their own classrooms. The teachers were engaged in postgraduate (masters level) professional development which emphasized the teacher as a designer and researcher of their own innovative practice. The activities were in the area of STEM and used a constructionist approach, aligned with the Bridge21 pedagogic model. Technology was central to the interventions and was used at the transformation layers of the SAMR model. Both interventions are considered constructionist in nature owing to their use of project-based learning in which students used technology to create learning artefacts, motivated by some real-world context, as a vehicle for collaboratively constructing understanding of the topic in question. Furthermore, as per the Bridge21 model, the teacher orchestrated learning rather than engaging in direct instruction. The interventions took place in school, either during normal classes or as part of an afterschool activity. (The latter occurred for reasons to do with the timing of the approval of research ethics. Approval was quicker to obtain for a voluntary, out-of-class study. In all other ways the activity mirrored a regular classroom experience.)

The Local Context and Bridge21

The Irish second level education system is going through a period of reform that is emphasising the development of key competencies, or 21st century skills, along with the acquisition of subject knowledge. The move is one in which “*curriculum and assessment arrangements will promote a focus on active and collaborative learning. In particular, learners will be enabled to use and analyse information in new and creative ways, to investigate issues, to explore, to think for themselves, to be creative in solving problems and to apply their learning to new challenges and situations*”, p7 (DES, 2015). Bridge21 is an approach to technology-mediated, collaborative learning which draws on a number of sources to create a model for 21st century teaching and learning that speaks to the current Irish curriculum reform process. It uses the teamwork model from the World Scout Movement (Bénard, 2002) and follows a lesson activity structure inspired by ideas from Design Thinking (Brown, 2008), see Figure .

Initially developed for use in workshops run on campus as part of the university’s social outreach agenda the overarching design-based research project (which this study is a component of) is now actively engaging with schools to adapt the Bridge21 model for use in the classroom. This overarching research is being conducted within the context of the current reform process in order to meet the twin goals of delivering curriculum content while at the same time promoting the development of key skills. An immersive model of teacher professional development is offered to interested schools (Girvan, Conneely, & Tangney, 2016) and for the past 3 years a university accredited Postgraduate Certificate in 21st Century Teaching and Learning has been offered, which makes extensive use of Bridge21 methodologies. One of teachers who co-authored this paper completed that certificate and both teachers completed M.Sc. dissertations on the use of Bridge21 in the classroom.

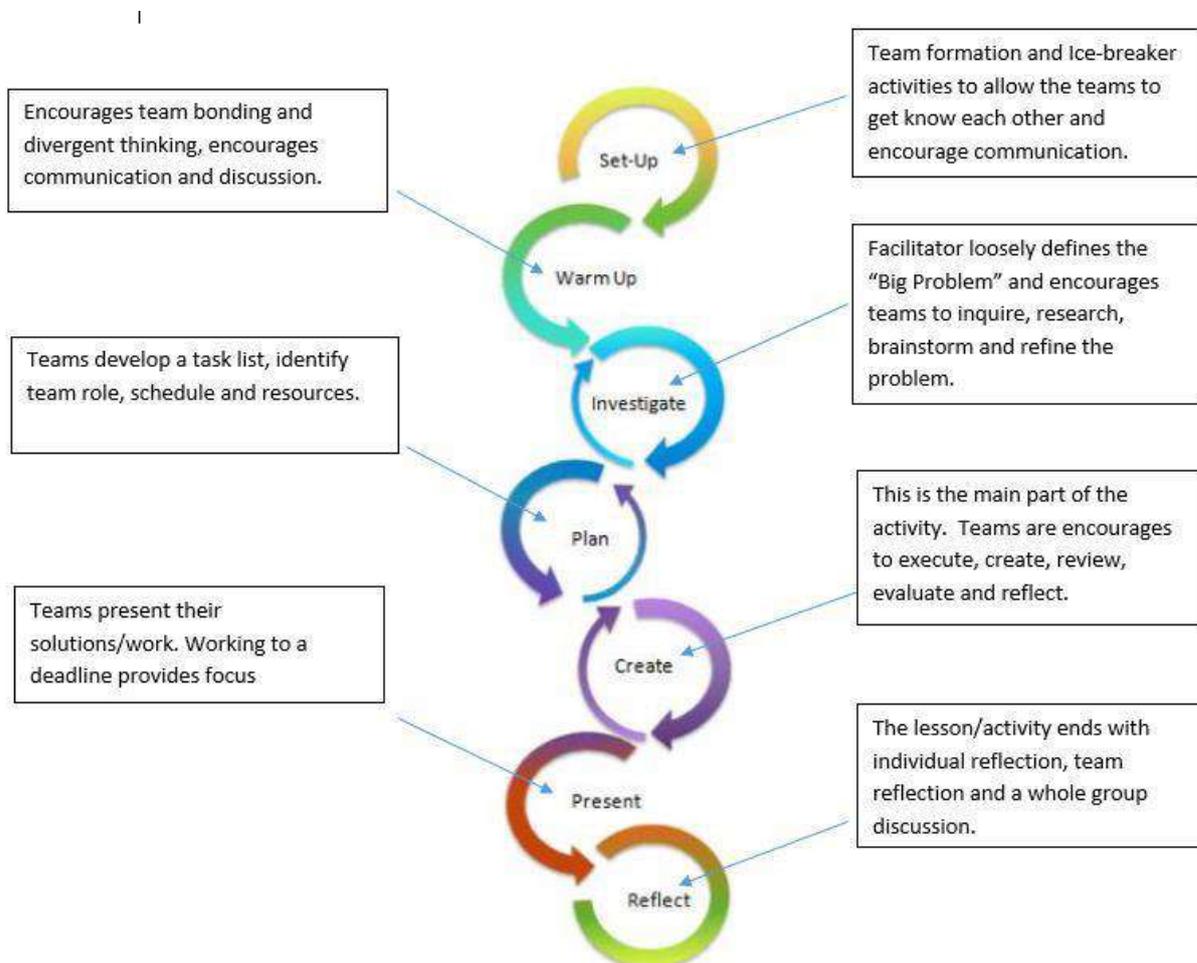


Figure 1 : The Bridge21 Lesson Activity Model. Image adapted from (Byrne, O’Sullivan & Sullivan 2017).

Methodology

This research is framed as an exploratory case study with two embedded units (Yin, 2014), and uses a mixed-methods approach to explore the effectiveness of the interventions, both in terms of their design and implementation in an authentic school setting, and the degree to which learners achieved the desired learning intentions. That is, the study is designed to explore whether a constructionist approach, aligned with the Bridge21 pedagogy and activity model, have a positive effect on learners' engagement, confidence and problem-solving ability in STEM subjects, within a traditional school context.

Mixed-methods research refers to studies in which the researcher synthesises ideas, techniques, approaches, methods, and concepts from quantitative and qualitative research, within a single study (Johnson & Onwuegbuzie, 2004; Johnson, Onwuegbuzie, & Turner, 2007). In this research, each embedded unit relates to a single intervention, in which both quantitative and qualitative data were collected. Both types of data were collected concurrently with emphasis given to quantitative data in the analysis.

Intervention A focused on developing problem solving skills in a group of 21 students (15-16 years old) in an after-school STEM club. The technologies used included a microworld for the construction of bridges.

Intervention B focused on the teaching of mathematical functions. A class of 24 students (15-17 year old) of mixed gender and weak to modest mathematical ability engaged in learning activities using a graphics calculator that focused on the teaching of functions with a view to improving their confidence and level of mathematical engagement.

In both interventions, pre and post tests were used to collect quantitative data, which were then analysed using paired *t*-tests in order to detect any changes that could be attributed to the activities. Qualitative data were gathered through focus group interviews. Open coding techniques were used to identify relevant codes and themes (Strauss & Corbin, 2008). These approaches will be discussed in more detail in the following sections.

The Interventions

The sections that follow provide an in-depth exploration of each of the two interventions, the data collection and analysis processes, and an overview of the findings of each of the teachers.

Intervention A - Problem Solving

Boran (2017) created an intervention that focused on developing students problem-solving skills. In order to provide a relevant context for the problem solving, the intervention focused on the challenge of bridge design using a microworld simulation. This topic requires students to use a high level of mathematics and physics as they must analyse, compare, and contrast the fundamental forces exerted on a load-carrying bridge. The problem-solving strategy they were encouraged to use was one devised by the teacher, based on a synthesis of ones found in the literature, all of which build on Polya's seminal work (Polya, 1945). A website⁵² was designed to scaffold the learning experience and to provide a detailed description of the open and closed problem-solving questions that students had to answer in the learning activities. Students worked in teams and the sessions followed the Bridge21 lesson activity model. The sessions took place in a standard classroom and students shared access to laptops. The problems they were presented with required them to construct, design, analyse and review a series of bridge designs which had to meet specific criteria. The activity required students to use their problem-solving skills to analyse the structural integrity of their designs and assess whether the design was cost-effective. As per the Bridge21 model students had to present their designs to their peers and defend the design decisions they made.

The microworld used was Bridge Designer⁵³, a free tool from West Point University. It was chosen as it has a low floor and high ceiling, encouraging students' exploration and creativity. The software is designed to provide primary and post-primary students with a realistic and contextualised introduction to the fundamental principles of bridge design and construction. The tool allows the student to interact and manipulate the key variables of bridge design (see Figure) and students are presented with a realistic 3D simulation of their design (see Figure).

⁵² <http://bridgedesignproblemsolving.weebly.com/>

⁵³ <http://bridgedesigner.org/download/>

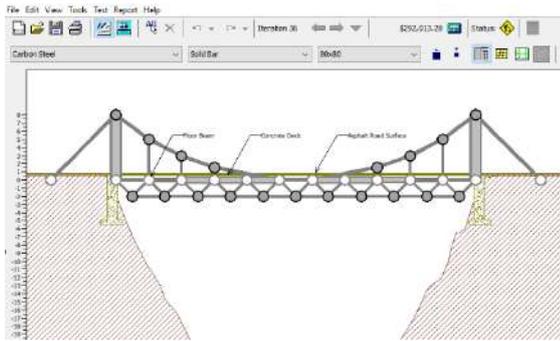


Figure 2: Microworld interface



Figure 3: Bridge simulation

The

microworld facilitates critical analysis, e.g. the load test allows the students to investigate the fundamental forces of compression and tension that are exerted on the individual joints and components in the bridge. Students can adjust the parameters in the structure and analyse the impact of these changes. Taken together the combination of the software and the Bridge21 methodology allow learning activities to be created which lie in the *redefinition layer* of the SAMR model. The learners are actively and collaboratively engaged in artefact construction and are thus facilitated in developing both their problem-solving skills and domain specific knowledge.

Participants & Duration

The study took place in the teacher's school as part of an after-school STEM Club. An initial pilot was carried out in which 20 students (15-16 years old) took part in activities of 4 hours in total duration. This pilot helped to test and tweak the design of the website, the suitability of the microworld, the details of the learning tasks and the appropriateness of the data collection tools. The main study involved 21 (different) students of the same age in 10 hours of learning spread over 4 weeks. The students attended the club on a voluntary basis and had a range of ability. The school is located in an area of socio-economic disadvantage.

Data Collection

Following the mixed-methods approach described above, the data collected were primarily *quantitative* but also included some *qualitative* data. The instrument used to collect quantitative data was the Metacognitive Activities Inventory (MCAI) (Cooper & Sandi-Urena, 2009). This was completed before and after the four weeks of the learning intervention, to assess any changes in student's metacognitive skills. The tool focuses on 4 main skills used when problem solving; planning, monitoring, reflection and general problem-solving skills. Focus group interviews were carried out with two groups, 7 students in each, pre and post intervention. Structured observation notes were recorded by the researcher, focusing on the participation of students in the activity.

Findings

The results from the MCAI questionnaire indicate an increase in students' metacognitive skills in each of the subscales post intervention, and a two-tailed t-test showed the change to be significant in each case at the 95% confidence interval (Table).

Table 1. Metacognitive Activities Inventory – Showing statistically significant changes, for $p < 0.05$, post intervention

	Mean-pre	SD-pre	Mean-post	SD-post	t(20)	p
Problem Solving Skills	2.3	.45	3.3	.51	-3.277	.004
Planning	3	.52	3.3	.49	-3.789	.001
Reflection	3	.37	3.3	.43	-2.677	.015
Monitoring	2.9	.64	3.2	.57	-2.461	.023

The qualitative data from interviews was transcribed and coded using open coding techniques. The themes that emerged were: evidence of the use/awareness of problem-solving strategies, the use of the problem-solving model, attitudes to problem solving, the effects of technology on learning, and the effect of the Bridge 21 learning model. The themes supported the positive findings arising from the quantitative data as typified in the following quotes.

- *“Yeah like I think the project was good at making us be creative and critically think about the design and structure of our bridges, sometimes in Maths we don’t get to be creative and innovative in Maths and it’s hard to reflect on our answers because we cannot check to see if they are right or wrong.” (Technology - Construction of an artefact, Bridge21 - Plan-Create-Reflect)*
- *“Yeah it was cool that you could learn from your mistakes and change your design, it kind of helped you and you could work out by analysing the load test and testing your bridge”. (Technology - Construction of an artefact, Bridge21 - Plan-Create-Reflect)*
- *“This was a fun problem, we discovered that the bars were too weak for the bridge to function, so we replaced the hollow tubes with solid bars so they would be more efficient, and then we tested our design to see what bars we could reduce the size of to be more cost effective.” (Technology - Construction of an artefact, Bridge21 - Plan-Create-Reflect)*
- *“It was fun I enjoyed the learning experience, you gave us a lot more control and we could make our own decisions, and you (the teacher) kind of let us work out the problems for ourselves which was good because we learned more by doing it ourselves.” (Bridge21 – Student responsibility for learning)*

In each of the above, the students emphasised the positive impact that the Plan-Create-Reflect cycle of the Bridge21 activity, combined with the construction of an artefact in a group setting had on their experiences. The teacher/researcher observations aligned with the analysis of the qualitative data from students in highlighting the role that the overall design of the intervention, the Bridge21 activity model, the open-ended problems and the constructionist tool, had on its effectiveness.

Intervention B – Mathematical Functions, Engagement & Confidence

In his work, Knox (2017), explored the use of a constructionist, Bridge21 approach to improve student engagement and confidence with mathematical functions. Within mathematics education, the topic of functions is recognised as difficult for students to grasp, and as one in which many students develop a number of misconceptions (Carlson, Oehrtman, & Thompson, 2005). This learning intervention was designed according to a set of heuristics (Bray & Tangney, 2015) that promote collaborative team-based learning and innovative use of technology to support the Realistic Math Education pedagogical approach (Gravemeijer et al., 1994). The technology used was the Desmos Activity Builder⁵⁴ and a number of challenges provided with it, including Marbleslides – see **Error! Reference source not found.**

The learning activities focused on students’ engagement with construction and problem-solving activities that required them to work with, and manipulate, different representations of functions, thus facilitating their discovery and ‘reinvention’ of the relevant mathematical ideas. The activities lie at the *modification layer* of the SAMR model as the affordances of the technology have permitted significant re-design of the task.

⁵⁴ www.desmos.com

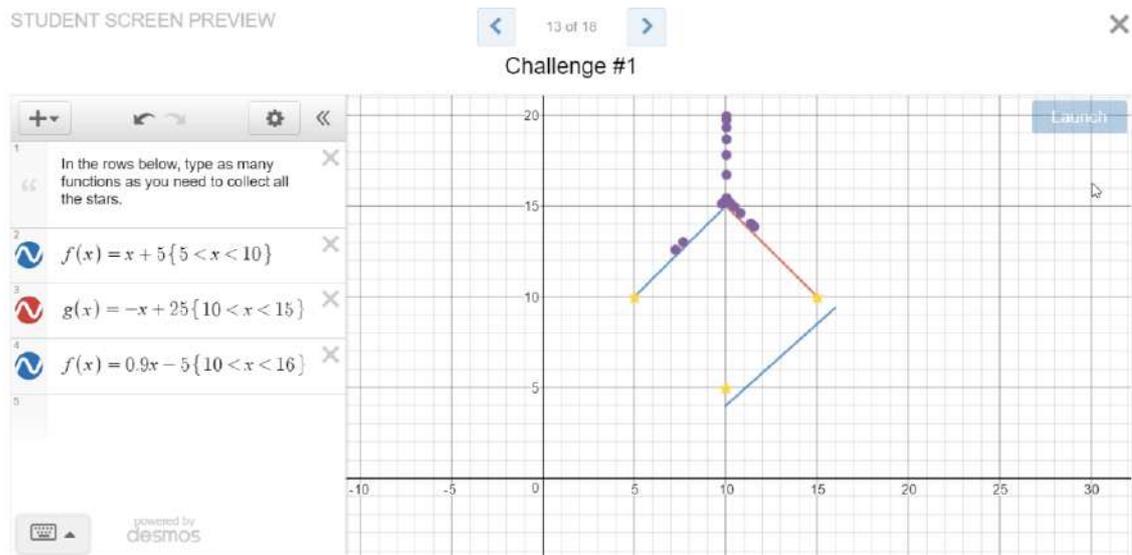


Figure 4: Marbleslides - built using Desmos Activity Builder. In this example students must create functions so that the falling purple marbles touch each of the yellow stars.

Participants & Duration

The lessons were conducted in the teacher's school with one of the teacher's own mathematics classes. The class was made up of 24 students (15-17 years old) of mixed gender and with weak to modest ability in mathematics. The learning experiences took place in normal timetabled class periods, of 40 or 80 minutes, and in total 8 hours and 40 minutes were spent on the intervention. The Bridge21 approach was used and the students worked collaboratively in teams to deal with the challenges presented. The Bridge21 Lesson Activity Model was utilised, but the short duration of some classes meant it had to be tailored to suit the time constraints. Once teams had been formed for the first class it was not necessary to repeat this part of the process. Some activities spread across multiple classes with, for example, one period devoted to the warm up/investigate steps, two periods to plan/create, and one period to present/reflect (Figure).

Data Collection

The Mathematics and Technology Attitudes Scale (MTAS) (Pierce, Stacey, & Barkatsas, 2007) was used to measure student confidence and engagement pre and post the overall learning intervention. MTAS is a validated instrument which has five subscales that measure: behavioural engagement (BE), confidence with technology (TC), mathematical confidence (MC), affective engagement (AE), and attitude to using technology for learning mathematics (MT). Qualitative data were collected through researcher observation, analysis of data captured by the web-based challenges, and through three semi-structured group interviews with a total of 11 participants, each interview lasting approximately 20 minutes.

Findings

The study shows evidence that the learning experience had a positive impact on the attitudes of participants. There were increases on each of the 5 MTAS sub-scales with the changes for Affective Engagement (AE) and Attitude to using Technology for Learning Mathematics (MT) both being statistically significant ($p < 0.05$) – see Table .

Table 2. Results of Two Sample *t*-tests for Means

	Mean-pre	SD-pre	Mean-post	SD-post	<i>t</i> (23)	<i>p</i>
BE	13.5	2.0	13.6	2.2	0.267	0.792
TC	14.3	3.4	15.8	2.3	1.934	0.065
MC	10.6	3.7	11.5	3.5	1.131	0.270
AE	11.5	2.4	12.8	2.6	2.734	0.012
MT	12.7	4.0	14.6	3.9	2.187	0.039

The focus group transcripts were transcribed and an open coding process followed which yielded 77 codes, which were categorised into 9 themes including: groups, attitude, engagement, learning and technology. These themes, and the observations made by the teacher/researcher during class, support the findings from the quantitative data that the students enjoyed the experience, found it engaging and saw the benefits of working in teams and with technology. These attitudes are typified by the following quotes.

- *“I thought they were beneficial, they were like really good and kind made you think about math in different ways.” (Attitude)*
- *“It was more interesting and I found that I looked forward more to coming to maths.” (Attitude)*
- *“I thought it was helpful because you got to see like every time you changed a number and like⁵⁵ something would happen on the screen and you'd see something and that helped.” (Technology – Constructionism)*
- *“Helpful because you had to work a few times at finding the numbers you needed for the graph.” (Technology – Constructionism)*

Discussion and Conclusion

Influenced by ideas to do with 21st century teaching & learning, the Irish second level education system (ages 12-18), as with others around the world, is undergoing a period of reform. A key challenge in that process is to create learning activities that promote the development of key skills while at the same time helping learners to master curriculum content, and to achieve both within the constraints of the regular school timetable and the system level requirements of covering a curriculum and meeting national assessment criteria. Even when the external regulatory framework, which Somekh (2008) identified as often being a barrier to change in classroom practice, is encouraging such innovation, teacher beliefs about teaching and learning remain what Ertmer (2005) describes as the “the final frontier in our quest for technology integration”.

Our research team has had considerable success in using the Bridge21 model of technology-mediated learning in the teaching of skills and content. However much of our results to date are based on work conducted in the learning lab on the Trinity campus, or in schools as one-off, boutique interventions that are not constrained by timetables. To assist teachers in changing their beliefs it is necessary, among other things, to show how innovation can be implemented within the constraints of the regular classroom, and to produce evidence of its benefits to students.

Focusing on the area of STEM education and innovative pedagogical approaches centred on ideas of constructivism and constructionism, this study looked at creative learning activities in two Irish secondary schools. We posited that the Bridge21 pedagogical approach, of team and project-based learning, and the associated activity model, which puts a structure on how lessons can be organized, was a pragmatic model for use in the mainstream classroom. The two cases showed that the model could be used in such settings, and resulted in statistically significant improvements in students' attitudes and meta-cognitive skills. The qualitative data, including the teacher-researcher observations,

⁵⁵ Many Irish teenagers make like excessive use of the word like when they are speaking.

supported the view that Bridge21 provides a suitable framework for the transformative use of constructionist technologies and practices.

There are of course limitations to a study of this scale. Within the timeframe, it was not feasible to measure student attainment and one of the studies took place in an after-school STEM club. However as explained above that was due to external factors to do with the timing of receiving university ethical approval rather than anything to do with the intervention. The afterschool sessions ran for 1.5 hours which is the equivalent of two consecutive class periods so it most respects it was an authentic in-school activity.

The study was framed as an exploratory one which as (Yin, 2014) points out is a useful methodology for the identification and refinement of research questions, hypotheses, or procedures that will be used in further research. The findings, while not being conclusive in any way, are however sufficiently positive to i) give encouragement to the researchers that the such research can be carried out in schools, and ii) to the teachers involved, and more importantly their peers in their own and other schools, that Bridge21 is a pragmatic model for use in schools to support the current reform process. To this end the research team continue to work with schools and teachers to provide the professional development support they need to empower them to innovate in their practices for the benefit of their students.

References

- Bénard D. (2002). *Handbook for Leaders of the Scout Section - A method of non-formal education for young people from 11 to 15*. Retrieved from <http://euroscoutinfo.com/wp-content/uploads/2012/05/Handbook-Scout-01.pdf>
- Boran I. (2017). *As Part of a 21st Century Learning Activity, an Investigation into the Effect of Utilising a Synthesized Problem Solving Model in a Microworld Simulation to Develop Problem Solving Skills in Maths Education*. (M.Sc.), Trinity College Dublin, the University of Dublin, Dublin. Retrieved from <https://www.dropbox.com/s/www6rabw9xgwkl4/MSc-IanBoran.pdf?dl=0>
- Bray A., & Tangney B. (2015). Enhancing student engagement through the affordances of mobile technology: a 21st century learning perspective on Realistic Mathematics Education. *Mathematics Education Research Journal*, 1-25. doi:10.1007/s13394-015-0158-7
- Bray A., & Tangney B. (2014). Barbie Bungee Jumping, Technology and Contextualised Learning of Mathematics. *6th International Conference on Computer Supported Education (CSEDU 2014)*, 206-213.
- Brown, T. (2008). Design thinking. *Harvard business review*, 86(6), 84.
- Byrne, J. R., O'Sullivan, K., & Sullivan, K. (2017). An IoT and Wearable Technology Hackathon for Promoting Careers in Computer Science. *IEEE Transactions on Education*, Vol 60, No 1, 50-58.
- Capraro RM, Capraro MM, & Morgan JR. (2013). *STEM project-based learning: An integrated science, technology, engineering, and mathematics (STEM) approach*: Springer Science & Business Media.
- Carlson M., Oehrtman M., & Thompson P. (2005). Key aspects of knowing and learning the concept of function. *Mathematical Association of America Research Sampler*, 9.
- Cooper, M. M., & Sandi-Urena, S. (2009). Design and Validation of an Instrument To Assess Metacognitive Skillfulness in Chemistry Problem Solving. *Journal of Chemical Education*, 86(2), 240. doi:10.1021/ed086p240
- Corbin, J., & Strauss, A. (2008). *Basics of qualitative research (3rd ed.)*. Thousand Oaks, CA: Sage.
- Danah, H., Punya, M., & Petra, F. (2016). Infusing Creativity and Technology in 21st Century Education: A Systemic View for Change. *Journal of Educational Technology & Society*, 19(3), 27-37.
- DES. (2015). *Framework of the Junior Cycle*. Retrieved from <https://www.ncca.ie/media/3249/framework-for-junior-cycle-2015-en.pdf>

Edelson, D. C., Gordin, D. N., & Pea, R. D. (1999). Addressing the Challenges of Inquiry-Based Learning Through Technology and Curriculum Design. *Journal of the Learning Sciences*, 8(3-4), 391-450. doi:10.1080/10508406.1999.9672075

Ertmer, P. A. (2005). Teacher pedagogical beliefs: The final frontier in our quest for technology integration? *Educational Technology Research and Development*, 53(4), 25-39. doi:10.1007/bf02504683

Fullan, M., & Langworthy, M. (2014). A rich seam: How new pedagogies find deep learning (pp. 100): London: Pearson.

Girvan C., Conneely C., & Tangney B. (2016). Extending experiential learning in teacher professional development. *Teaching and Teacher Education*, 58, 129-139.

Girvan C., Conneely C., & Tangney B. (2016). Extending experiential learning in teacher professional development *Teaching and Teacher Education*(58), 129-139. doi:10.1016/j.tate.2016.04.009

Glynn M. (2017). *An investigation into the use of the Bridge21 model to deliver the new Junior Cycle Science specification*. (M.Sc.), Trinity College Dublin, the University of Dublin, Dublin. Retrieved from <https://www.dropbox.com/s/glbgnimrvgtdey6/MSc-MaireadGlynn.pdf?dl=0>

Gravemeijer K., Rainero R., & Vonk H. (1994). *Developing realistic mathematics education*: Freudenthal institute Utrecht, The Netherlands.

Henriksen EK., Dillon J., & Ryder J. (Eds.), (2015), *Understanding Student Participation and Choice in Science and Technology Education*. Springer.

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed Methods Research: A research paradigm whose time has come. *Educational researcher*, 33(7), 14-26.

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a Definition of Mixed Methods Research. *Journal of Mixed Methods Research*, 1(2), 112-133.

Johnston K., Conneely C., Murchan D., & Tangney B. (2015). Enacting key skills-based curricula in secondary education: lessons from a technology-mediated, group-based learning initiative. *Technology, Pedagogy and Education*, 24(4), 423-442. doi:10.1080/1475939X.2014.890641

Knox T. (2017). *An Investigation into Bray's Heuristics for Mathematical Learning Activities as Applied to Functions*. (M.Sc.), Trinity College Dublin, the University of Dublin, Dublin. Retrieved from <https://www.dropbox.com/s/lqnhjtdvcr0rpml/MSc-TonyKnox.pdf?dl=0>

Lawlor J., Conneely C., Oldham E., Marshall K., & Tangney B. (2018). Bridge21: Teamwork, Technology and Learning - A pragmatic model for effective 21C Team-based Learning. *Technology, Pedagogy and Education*. doi:<https://doi.org/10.1080/1475939X.2017.1405066>

Lawlor J., Marshall K., & Tangney B. (2015). Bridge21 – Exploring the potential to foster intrinsic student motivation through a team-based, technology-mediated learning model. *Technology, Pedagogy and Education*, 1-20. doi:<http://dx.doi.org/10.1080/1475939X.2015.1023828>

Maaß, K., & Artigue, M. (2013). Implementation of inquiry-based learning in day-to-day teaching: a synthesis. *ZDM*, 45(6), 779-795. doi:10.1007/s11858-013-0528-0

Marginson S, Tytler R., Freeman B. & Roberts K (2013), *STEM: country comparisons: international comparisons of science, technology, engineering and mathematics (STEM) education. Final report.*, Australian Council of Learned Academies, Melbourne, Australia. <http://hdl.handle.net/10536/DRO/DU:30059041>

Pierce, R., Stacey, K., & Barkatsas, A. (2007). A scale for monitoring students' attitudes to learning mathematics with technology. *Computers & Education*, 48(2), 285-300. doi:<https://doi.org/10.1016/j.compedu.2005.01.006>

Polya, G. (1945). *How to solve it: A new aspect of mathematical method*. Princeton University Press.

- Puentedura, R. (2012). The SAMR model: Background and exemplars. Retrieved from <http://hippasus.com/rrpweblog/>
- Somekh, B. (2008). Factors affecting teachers' pedagogical adoption of ICT *International handbook of information technology in primary and secondary education* (pp. 449-460): Springer.
- Strauss, A. L., & Corbin, J. M. (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory 3e* (3rd ed.). Thousand Oaks, CA: Sage Publications, Inc.
- Tangney B., Bray A., & Oldham E. (2015). Realistic Mathematics Education, Mobile Technology & The Bridge21 Model For 21st Century Learning – A Perfect Storm. In Crompton H. & Traxler J. (Eds.), *Mobile Learning and Mathematics: Foundations, Design, and Case Studies* (pp. 96-105): Routledge.
- Voogt J., & Roblin N.P. (2012). A comparative analysis of international frameworks for 21st century competences: implications for national curriculum policies. *Journal of Curriculum Studies*, 44(3), 299-321.
- Wickham C., Girvan C., & Tangney B. (2016). Constructionism and microworlds as part of a 21st century learning activity to impact student engagement and confidence in physics. *Constructionism 2016*, 34-43.
- Yin, R. K. (2014). *Case Study Research: Design and Methods* (5 ed.). Thousand Oaks, CA: Sage Publications, Inc.

Exploring Girls' Values and Perspectives in Making for Others

Sawaros Thanapornsangsoth, st2839@tc.columbia.edu

Teachers College, Columbia University, USA

Nathan Holbert, holbert@tc.columbia.edu

Teachers College, Columbia University, USA

Abstract

This study explores a pedagogical framework on how “making for others” can influence and engage girls in maker activities. We are particularly interested in how the relationship between builders and their clients can influence the builders’ making process and their motivation throughout that process. Feminist literature suggests that girls tend to locate themselves in relation to the world and describe themselves through actions that bring them into connection with others (Gilligan, 1982, p. 35). Leveraging the school’s Big Sister - Little sister mentorship program, the fourth-grade builders were asked to make toys for their first-grade clients. Throughout the year, builders worked closely with their clients to iterate on their toy designs, developing a close relationship and personalizing constructed toys to align with the clients’ requests. The interview data indicate that builders were constantly thinking about their clients’ needs and that they were proud of seeing their client’s satisfaction with the handmade toys. Additionally, we have found that encouragement and emotional support from peers and teachers are also vital for young female builders in completing their projects.

Keywords

making; girls; constructionism; maker movement; diversity; elementary students

Introduction

Peppler and Bender (2013) refer to the *maker movement* as “a growing culture of hands-on making, creating, designing, and innovating.” (p.22) Similarly, Halverson and Sheridan sees the maker movement as a group of people who are involved in “creative production of artifacts in their daily lives” (2014, p.496). As a movement, a group of “makers” often share their processes and products with others. Makers’ distinctive identity is their do-it-yourself (or do-it-with-others) mindset. They find making, tinkering, inventing, problem-solving, discovering and sharing intrinsically rewarding (Kalil, 2013). *Maker activities* provide learners with hands-on experience to design, experiment, build, and invent as they engage in activities that involve science, technology, engineering, math (STEM), and art. It draws people with interest in a range of activities, from textile craft, robotics, cooking, wood-crafts, electronics, digital fabrication, or mechanical repair (Peppler & Bender, 2013). Making in education is not new. Constructionist scholars and progressive educators have been advocating for learning through making since the 1960s. When learners make, they construct personalized connection to the artifact that engages their thinking, feeling, and learning (Wilensky, 1991). The construction can be anything from programming to painting, to carpentry, to making a hypothesis for a scientific experiment, or even to writing poetry (Papert, 1980). The term “maker movement” became more widely recognized in the early 2000s with the establishment of Make magazine and Maker Faire (Dougherty, 2016). Nevertheless, the making practices that were portrayed through these popular media outlets have demonstrated a very narrow representation of making featuring mainly male dominated practices like electronics, vehicles, and robots (Buecheley, 2013).

The maker movement has a high potential to reach people from diverse backgrounds through hands-on making. However, the movement has predominantly targeted wealthy and highly educated men (Buecheley, 2013; Maker Media, 2014). In an effort to expand the space of what counts as “making” in this maker movement, researchers and designers have worked to bridge traditionally feminine domains such as craft and fashion to computing and engineering. For example, the LilyPad Arduino allows

learners to sew and program electronics. Many organizations have developed coding games for girls as well as organizing girl-only coding camps. These efforts have played an important role in increasing girls participations in maker activities; however, there is a danger in over-generalizing or simplifying womens' interests. Generalizing girls' preferences and superficially including features like fashion and beauty into the learning environment can be problematic. The act may aggravate gender stereotypes and community divides (Holbert, 2016). An alternative way to create a more inclusive maker-centered learning environment may be to directly consider women's values and goals and then evaluate how these values can be reflected in the design of maker activities and spaces.

This paper explores a pedagogical framework based on values cultivated by women to build connections with others (Belenky et al., 1986; Gilligan, 1982) that can engender more girls to participate in maker activities. It is a part of the wider two-year Bots for Tots design-based research project (Holbert, 2016) where we engage young learners from diverse communities to build toys for younger kids in their school rather than for themselves. Using data from the second iteration of the Bots for Tots program where all participants were girls, we investigate the relationship between fourth-grade builders (Big Sisters) and their first-grade clients (Little Sisters) and the relationship's impact on the builders' making process to explore a pedagogical framework on designing maker activities for girls. In this study. Our research aims to answer the following questions:

- How does making personalized toys for their clients motivate and help builders to persist through challenges in completing their toy design
- How do builders persist through times of discouragement encountered during their making process?

We present findings about how the relationship between builder participants and their clients motivated the builders during the making process.

Literature Review

Diversity in the maker movement: Engaging women in making

In the United States, the maker movement has struggled to expand its participations beyond affluent and well educated men. An attendee survey of the 2014 Bay Area Maker Faire showed that 70% were male, 97% attended or graduated college, and earned a median household income of \$130,000 (Maker Media, 2014). Moreover, 85% of the 41 people featured on Make magazine covers (2005-2013) were men. Likewise, a very narrow definition of maker activities had been portrayed on the covers featuring, 53% electronics, 31% vehicles, 22% robots, 8% rockets, and 5% music (Buechley, 2013). Revisiting these data three years later, Buechley (2016) called on the maker community to focus their efforts on inclusion, arguing that if the program is not inclusive, it is discriminatory. Instead of narrowly defining STEM and maker activities, Buechley argued for the inclusion of STEM-rich activities practiced by diverse communities and cultures (2016). For example, knitting should also be considered as a mathematically-rich activity as creating designs and patterns requires intricate calculation.

At the same time, we should not force learners to take on an externally imposed identity but should support them in building their own unique one. For instance, not all learners will be drawn to the "hacker" identity predominantly advertised by many maker communities (Worsley & Blikstein, 2016). Women in particular may be find this framing uninviting (Fisher & Margolis, 2002). Martinez (2015) suggest educators to be sensitive of their classroom environment as women can react to surroundings that reflect stereotypical hacker culture by denying that they are interested in science and engineering.

Linking science and technology with learners' values and area of interest can help create an inclusive STEM and maker-centered learning environment for women (Rosser, 1990). Findings from Margolis and Fisher's (2003) longitudinal research show that context is very important for women. They suggest bridging other disciplines such as medicine, arts, and environmental science to computer science as well as connecting learners with local communities to sustain women's interests in technology-focused fields.

Margolis and Fisher's (2003) research also aligns with Belenky, Clinchy, Goldberger and Tarule's (1986) seminal work on ways of knowing cultivated by women. They examine two distinctive forms of knowing: separate and connected knowing. Generally, the separate knowers tend to be more critical and detached from feelings and emotions. On the other hand, connected knowers are more empathetic. They try to understand others' perspectives and share their own experience to foster relationship. They are driven by the desire to connect. Gilligan (1982) used the term "separate" and "connected" to describe two different conceptions or experiences of the self (separate from others) and in relationship (connected to others). Interviewing eleven-years-old boys and girls, Gilligan found that girls tend to locate themselves in relation to the world and described themselves through actions that bring them into connection with others (p.35). To examine separate and connected knowing, Belenky et al. (1986) conducted a survey that holds statements indicating "separate knowing" and "connected knowing" and found that their female participants had higher tendency toward connectedness. It is noted that separate and connected knowing are not gender specific and both ways of knowing have no correlation to intellectual capacity. Though not all women are connected knowers, this epistemological belief can help us design an inclusive and supportive maker-centered curriculum suitable for women without relying too much on activities or discourses related to gender stereotypes such as girls prefer pink toys or beauty related products (Holbert, 2016).

As Belenky et al. (1986) and Gilligan (1982) have suggested, women tend to value social interactions and a sense of community. They talk about their negative feelings (stress, anger, and disappointment) with others significantly more often than men (Simon & Nath, 2004). In a stressful situation, women are more likely to express their feelings and cope with their emotions by seeking social support, as compared to men (Thoits, 1995). In maker activities, women work collaboratively, provide support, and help their peers more often than men (Intel Corporation, 2014). Women are more motivated by the social service aspects of making, particularly, they want to help or give. This act can be as simple as creating gifts for family and friends (Intel Corporation, 2014). Literature suggest women have higher tendency than men to favor collaborative relationships to competition and enjoy helping or giving back to their community (Mosatche et al., 2013). Leveraging these findings, maker-centered environment that support working together and working for others can leads to an increase in confidence and performance for women in making.

Methodology

Population and Site

The data presented here is from 41 fourth grade builders (aged 9 to 11) taking an Engineering and Design class at an all-girls private school in a suburban area in the North-Eastern United States. Leveraging school's Big Sister - Little Sister mentorship program, the Bots for Tots project had an explicit goal of having the fourth grade builders or "Big Sisters" to design and build toys for their first grade clients or "Little Sisters". The class began with two sessions of making with a 2D to 3D objects with cardboard in order to familiarize the builders with tools and materials in the lab as well as warming them up for creative activities ahead. After the cardboard sessions, the builders interviewed their clients using the "Client Profile" worksheet. They questioned their clients about the toy they liked and disliked. Then, the builders used the Client Profile worksheet to discuss with their classmates and brainstorm toy ideas within a small group. After two sessions of prototyping, the builders met their clients again to show prototypes they made and receive feedback for further iteration. The builders spent seven sessions to make their final designs. The instructor also allowed the builders to take the toy home for the last weekend if they could not complete the toy in the class time. The builders met their clients for "play date session" on the last day of the semester where the girls exchanged the toys and played together. All names used in the paper are pseudonyms chosen by the participants.

Data Collection

Interview

Of the 41 builders participating in the study, 12 were randomly selected for one-on-one interviews. We interviewed the builders at the beginning of the year, before the Bots for Tots project had begun to

determine builders' experience with technology, construction, and crafts as well as knowledge of relevant making and engineering concepts or skills. Interviews lasted approximately 30 to 40 minutes per builder. We interviewed the same builders again at the end of the year after the class had concluded. In the post interview, builders were asked about their experience of making toys for their clients. Our goal was to understand how making for their clients influenced the overall making process and how builders may have developed an interest in making. The interview was video recorded and transcribed. The data was coded "bottom-up" where themes and coding categories emerged from patterns in the data (Miles et al., 2014). For example, one code categorized instances where the participants changed their initial designs after receiving feedback from their clients and another identified instances where participants reference their clients while making their toys. A subset of the codes relevant to this analysis were applied by coders outside of the primary coding team. Interrater reliability was computed for each data set with all sets achieving greater than 0.7 Cohen's Kappa initially and improving to greater than 0.9 after discussion.

Field notes

Detailed field notes were taken during observations of the Bots for Tots project. These observations focused on fourth-grade girls' conversation with classmates and instructors about their clients' toy preferences and feedback. Particularly, we looked into the builders' development of toy ideas, feedback from interview with clients, changes that builders had implemented after the feedback session.

Artifacts

A variety of artifacts were produced by builders throughout the Bots for Tots project. These include in class worksheets, photographs of builders giving toys to their clients, as well as photographs of their toy designs throughout the construction process. Following are artifacts investigated:

- *My Client Profile worksheet*: Builders filled out the My Client Profile worksheet, indicating their client's toy likes and dislikes, during the first client interview. Here, we investigated the requests their clients originally made.
- *Brainstorming worksheet*: Using the data gathered from the client interview, builders discussed and brainstorm toy ideas with their classmates. Next, they finalized their toy idea and materials that they would use by drawing and filling in their toy design plans. We wanted to see how builders brought in clients' preferences in coming up with project ideas.
- *Client's feedback worksheet (Figure 2)*: Approximately half of the way into the project, builders presented their prototypes to their clients to receive feedback. This form guided that process by providing questions that included clients' preferences and concerns about the prototype and whether or not the prototypes met their expectation. We wanted to look into feedbacks from the clients and how that the feedback had led to the builders' changes in their designs.
- *Prototypes and Final Project*: Builders built prototypes of their toys as a quick and tangible way to express their ideas and receive feedback from their clients. The final project was the last iteration of the toy given to the clients on the play date session. We investigated the builders' prototypes and final projects to see the changes they had made in each stage of designing toys for their clients.

Study Design

41 fourth grade builders from two classes (20-21 builders per class) participated in the study as a part of their Bots for Tots project, a bi-weekly class which ran 45 minutes per session. Throughout the academic year, builders participated in 19 sessions in total as described on Table 1.

Table 1. Structure of Making and Engineering class

Name of activities	# of sessions the activity occurs	Major activities
Pre workshop interview with twelve builders	1	Interview 12 builders about their experience and familiarity with tools and toys related to maker activities
Make 2D to 3D cardboard animals	3	Created 3D animals from cardboard
Interview with clients	1	Interviewed first-grade clients about their dream toys
Brainstorm with small group	2	Shared information gained from interviewing with clients and asked classmates for inputs about their design ideas.
Prototype	2	Made prototypes
Receive feedback from clients	1	Showed clients the prototype and asked for feedback
Complete final toy design	7	Revisited the feedback from clients and planned for improvements. Finalized toy construction.
Toy delivery and play date	1	Met clients for toy delivery
Post workshop interview with twelve builders	1	Interviewed the 12 builders about their experience in the Making and Engineering class.

Results



Figure 1. from left to right. 1) Amy's prototype of orange emoji pillow inspired by her My Client Profile worksheet. 2) Amy's final project. 3) Amy and her client at the playdate, holding the card her client made.

We divide the investigation of artifacts made by fourth grade builders into three phases: (1) after the first client interview, (2) after the client feedback session, and (3) before a playdate with clients. We found 27 out of 41 students kept their original project ideas from phase (1) to phase (3). The 14 remaining students changed their toy ideas after phase (2) but still showed alignment to their clients' requests. Out of 41 projects, 40 projects aligned with what was requested by their first grade clients. Throughout the three phrases, builders also made changes to their design to serve their clients' preferences. For example, Amy's client profile sheet suggested her client likes emojis: happy, hearts, and sparkly eyes. The client preferred the toy to be "kind of hard with a lot of stuffing and puffy." Amy made an orange emoji pillow from felt that was roughly sewn together as her prototype and showed it to her client for feedback. Her client liked the pillow and only asked if Amy could change the color of the fabric from orange to yellow and added "sparkly eyes" using pink and yellow fabric. On the final project, Amy took her client's feedback and made a heart eye yellow emoji pillow. She also recorded her voice on a recorder module to play, "have a nice day!" once the pillow was squeezed (Figure 1).

Amy was one of the 27 builders who held onto their original toy design requested by the clients since the first client interview, making only minor changes. 14 builders completely changed their designs. These changes were made because of several reasons including difficulties in making original ideas (8 cases), feedback from clients (2 cases), materials constraint (1 case) and peer influence (3 cases).

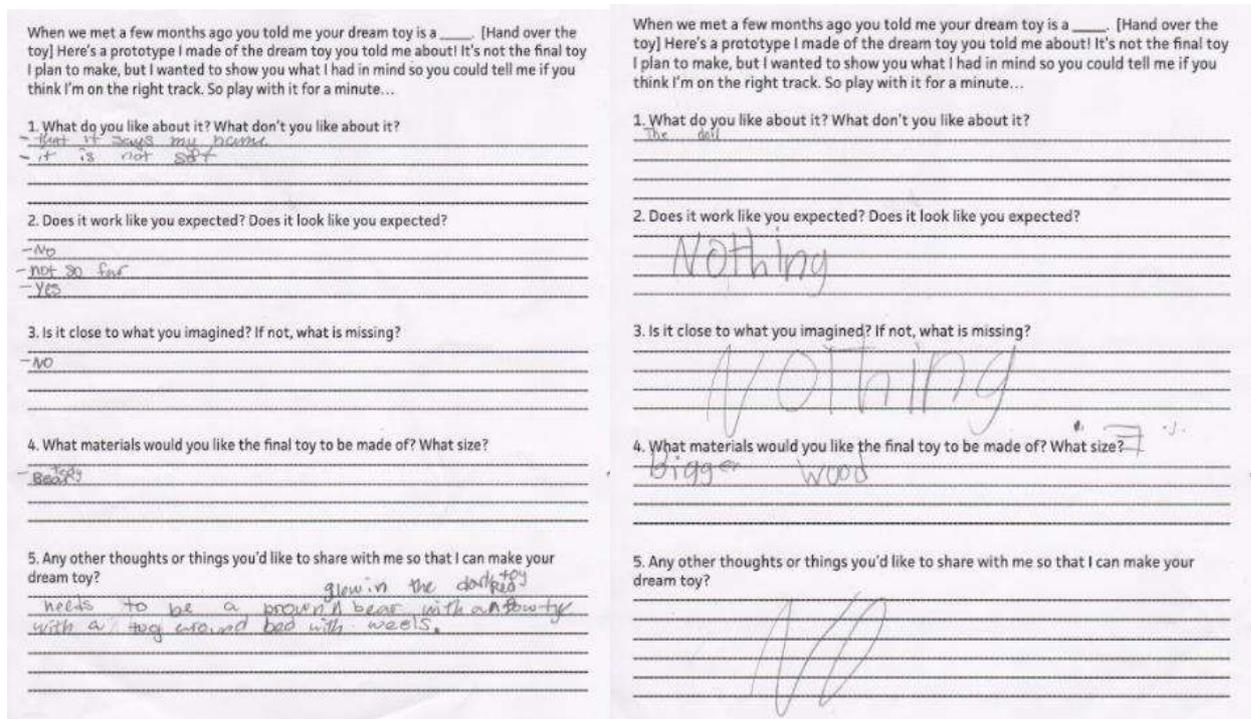


Figure 2. Ruth (left) and Betty's (right) responses from their Client's feedback worksheet

Unlike Amy, whose client loved her design and only requested minor changes, 10 out of 41 builders indicated negative feedback. For example, Ruth answered "No" on her client feedback worksheet for the question: It is close to what [client] imagined? (Figure 2). Ruth's client did not like her prototype at all. She made a pillow with a voice recorder that spoke "I love you, [name of the client]" when squeezed, but her client wanted a doll house. Ruth was discouraged and worried that she would not be able to make a doll house. She spent the next session listlessly looking at other projects and playing with materials. Her friends and teacher encouraged her to try to make the dollhouse. The teacher asked Ruth to cut plywood into five rectangular pieces and showed her how to assemble the pieces into a cube with one side open. Ruth was excited to see an initial structure of the doll house. However, the structure collapsed when she returned for the next session. Ruth was disappointed that she had to start all over again. Feeling discouraged, the teacher had to work closely with her and suggested she use a glue gun instead of masking tape. Once the structure was finished and had a strong foundation, Ruth

knew exactly what to do. She picked up several colorful tubes of acrylic paint and started to paint her doll house (Figure 3). At the end of the class, Ruth said “I’m so proud of myself today!” When asked why, she answered “I worked so hard. I’m almost done!”

Like Ruth’s client, Betty answered “NOTHING” on her client feedback worksheet (Figure 2). Betty’s client did not like her prototype and it was “NOTHING” like what she expected. Betty intended her dollhouse to have two levels. But, she cut one piece of wood slightly smaller than the others so when she assembled her house together, it was slanted. She looked unhappy with her design, so her classmate provided encouragement by saying that it looked “like a cool modern loft house!” Another agreed and said Betty could decorate her dollhouse as a fun, “whimsical witch house”. However, Betty still appeared unsatisfied and said that she would not continue her dollhouse project and would make a pillow instead. Seeing that Betty already poured in a lot of time and effort for her project (cutting and gluing wood), the teacher suggested that she take the second floor out and redesign her roof. Betty agreed and took out the top floor. She added a piece of plastic on the top to create a roof with skylight. She also made a loft floor for a bedroom (Figure 3). During the post interview, Betty told us that her favorite part of Bots for Tots was to see how surprised her client was to receive the toy she made. She said, “I liked when I gave it to my little sister. I like to see how like happy and like surprised she was.” Betty was proud of her creation, even though she almost changed her design to a pillow.



Figure 3. Ruth (left) and Betty’s (right) final designs for their doll houses on the day of Play date.

Pillows were a popular choice among builders that did not pursue the original idea. While only one client originally requested a pillow, ten girls made pillows. For example, Panni told us during the post interview that her client originally wanted a stuffed animal that has a “head of shark with a dolphin body”. She tried drawing and cutting a shark’s head but she was not able to make it; she said “It was kind of hard.” So Panni asked her client if she could make her a pillow instead. As pink was her client’s favorite color, Panni made a pink polka dot pillow with rainbow mesh. She wrote her client’s name a side of the pillow. Out of ten pillows, nine showed personalization features specific to each client. Several personalized details were requested by the clients such as the clients’ initials, soft circuits, voice records, and sketches.

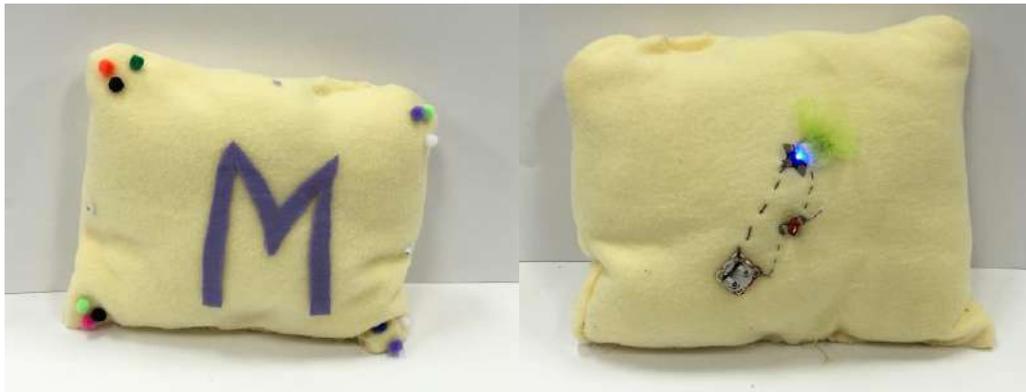


Figure 4. Lightbulb's pillow for her client (front and back)

One of the ten builders who made a pillow, Lightbulb, described her project as not a “boring pillow”. She decorated one side of her pillow with pom-poms and an “M” for her client’s name. On the other side, she made a soft circuit where her client could “flip the switch a little star thing would light up” (Figure 4). She told us during the post interview that she felt like her client “wanted more than just a boring pillow”. She added that, “I wanted to do more for her, instead of just a pillow that said, [client’s name]. I wanted to make it special and personalized, that’s why I did the—because not a lot of people have light up pillows.” Since Lightbulb finished her project two weeks before the playdate, she decided to make a set of wooden blocks, a toy her client originally requested. She admitted that she made a pillow because she initially did not know what to do, “I just didn’t know what to do so I was like, why don’t I make a pillow.” She then realized that her pillow was not relevant to her client’s preference and recalled that her client liked to play with Lego bricks. She said, “well these [wooden blocks] are kind of like Legos, so maybe it would add something more to the project that [her client] would like.” Additionally, Lightbulb told us during the post interview that not only did she like making things, but she also liked seeing the reaction of her client after receiving the toys she made. She felt really proud as she “liked watching [her client] like play and enjoy the blocks.”

Lightbulb was not the only builder who felt proud when her client enjoyed her handmade toy. Out of 12 builders interviewed, 11 also said that they felt proud or happy when they saw that their clients liked their projects. For instance, Aditi said that she was happy when her client was hugging and playing with her bunny stuffed animal. LillyJane said that she liked seeing reactions of her client when she received the gift. Even though the making experience was challenging, it was really fun for her. Like LillyJane, Hailey thought that it was hard to know what her client liked but it was fun making for her client and seeing her love the personalized pillow. She said, “I really wanted to make something that she would remember and really love.” Sarah thought that her project was “not as hard as some other people’s” but she was still proud of herself for putting in effort and for satisfying her client’s requests.

Furthermore, the post interview data showed that the builders often talked about their clients even when they were not asked about them. Our first seven questions on the post interview protocol aimed to understand the builders’ feelings toward the Bots for Tots project and their perceptions on technology and craft. We did not ask the builders about their clients in these questions. 11 out of 12 builders talked about their clients unprompted in the first seven questions, six girls said that making toys for their clients was their favorite part of the class. For example, LillyJane told us that her favorite part was interviewing her client because it allowed her to know her client better and to personalize the toy for her. Hailey explained that her favorite part was helping her classmates with making and creating a toy that her client would love. Hailey liked prototyping so that she could receive feedback from her client and improve. Moreover, when asked about their feelings toward the toy they made or about things they would have liked to do differently seven girls mentioned their clients. For example, Erica answered that she “really, really liked” a colorful dollhouse she made. She then added, “and I think [her client] really liked it too.”

One girl, Aditi, mentioned her client on four out of seven unprompted questions. During her pre interview Aditi was timid and quiet often answering questions with, “I don’t know” and “Mm-hmm.” However, in the post interview, she gave lengthy responses and often referred to her client. Aditi said that

brainstorming session was the most challenging task because her client said “yes to everything” that it required extra work to come up with a project idea. Aditi ended up making a pink bunny stuffed animal with a letter “J” (her client’s initial) on its belly. She said that she enjoyed the making experience and it was fun designing for her client. When asked Aditi if in a future Bots for Tots project she would prefer to build for herself or for someone else, she said she would prefer making for someone else because the receiver would be surprised and happy. She would not enjoy the making experience that much if she were to make for herself.

Similar to Aditi, nine other builders interviewed would prefer making for someone else if they were participated in the Bots for Tots project again. Panni wanted to make for others because she loved hugs and the receivers often gave her hugs of appreciation. Another builder, Margie, explained that she wanted to make for others as she would try hard because she did not want her receivers to feel sad. Erica wanted to make for her parents and her good friends because she knew exactly what they liked. Nevertheless, five of them said that they also wanted to build something for themselves. The most common reason was that they felt proud of what they had made and wanted to keep it.

Discussion

A goal of this study was to explore a pedagogical framework on how “making for others” can influence and engage girls in maker activities as well as exploring the emerging supports that the girls need throughout their making process. We wanted to provide an alternative way to create a more inclusive maker-centered learning environment that directly consider women’s values in cultivating personal and community connections. We also suggest that providing girls opportunities for social and emotional support is important in the design of maker activities and spaces.

Making for others: Building toy, building relationship

Belenky and her colleagues (1986) suggested that women are likely to be driven by the desire to connect with others at a personal level. The school’s “Big Sisters-Little Sisters” program encouraged the fourth grade builders to build a year-long mentor-like relationship with their first grade clients. This intimacy and relationship with their first grade clients motivated the builders throughout their making practices. Apart from making toys for their clients in Bots for Tots project, the fourth grade builders also provide care and guidance to their client, as it was their first year of elementary school. They shared the same building and often met during recess and school activities.

While the Bots for Tots project was part of a formal school experience, activities and artifacts would not be graded. Nevertheless, the builders were motivated by their clients’ preferences, as 40 out of 41 projects showed alignment with what was requested by their clients. The builders took what they had learned from client interviews to come up with project ideas and more than half (27 builders) were committed to the original ideas throughout the year. In the 13 other instances where the builders could not carry out their original plans, they added personalized details for their clients. For example, in making pillows, the builders chose fabric based on their client’s choice of colors and often included clients’ initials, voice recording, and sketches. This is demonstrated by Lightbulb, who incorporated a soft circuit to her pillow because her client did not want “just a boring pillow.” She wanted to make her project special and personalized. Lightbulb was the only builder in the class who made the soft circuit. It was also her first time experiencing with the materials.

In addition, the post interview data showed that builders were constantly thinking about and referring to their clients even when they were not asked about them. They try to understand the other’s point of view and tend to position themselves through actions that bring them into connection with others (Galligan, 1982). The clients were important to the builders’ making experience. Making the toys for their clients connects and tightens the relationship between the builders and the clients. Interestingly, 11 out of 12 builders talked about their clients when discussed about their favorite parts of the Bots for Tots project and their feelings toward their projects. LillyJane’s favorite part of the class was interviewing and getting to know her client better so that she could make the toy that her client would love. While Panni regretted about the toy she made because she could not make the toy that was originally requested by her clients. The builders empathized and connected with their clients

The builders are particularly motivated by the social aspect in making and their desire to help or to give. They tend to feel a sense of accomplishment in making and often depend on personal and community connections (Intel Corporation, 2014). When asked whether the builders prefer building for themselves or for someone else if they were to participate in the Bots for Tots project again, ten answered that they wanted to build for someone else. The most common reason was that the builders felt happy when they saw that their clients appreciated their handmade toys. 11 girls interviewed talked about how seeing their clients like their toys made them feel proud or happy. For instance, Aditi was happy when she saw her client hugging and playing with the bunny stuffed animal she made. LillyJane also liked seeing her client's reactions when they received the gift. This finding confirmed how relationship and feelings were tied to the builders' motives in making.

Persisting through challenges: Motivated by clients, encouraged by peers and teacher

The feedback and reactions from clients toward the toys affected builders' feelings and confidence. The comments-- both positive and negative--prompted the builders to improve their designs. Builders made changes on their designs based on their clients' feedback. While 31 clients were satisfied with the prototypes, only requesting builders to make minor changes, ten of the builders received negative and discouraging feedback.

Despite the tough feedback, the builders overcame challenges and finished toys that satisfied their clients. They took comments that may be discouraging to improve their toys and ensured that their clients were happy. Here, the builders have higher tendencies toward being connected knowers and "learn through empathy" (Belenky et al., 1986, p. 115). For example, Margie's client told Margie that her pillow looked like a sausage, when her intention was to a tennis ball pillow after learning that her client liked to play tennis. Margie improved the design of tennis ball pillow and added a small tennis court made of wood, knowing that her client loved tennis. At the end of the project, Margie was proud of what she made and told us that her client also liked her toys. She said, "[her client] really liked the pillow. That's her favorite. She was like leaning on it." Despite receiving harsh comments on their prototypes, they tried to understand their clients' perspectives and made personalized toys to foster their relationship.

Nevertheless, negative feedback can also demotivate the young female builders, especially when the feedback came from a person who they had fostered meaningful relationship with. In a stressful situation, young female builders need close facilitation and emotional support from peers and teachers (Thotis, 1995). Sometimes, the form of support can be as simple as giving a compliment to the builders. Words of encouragement like: "I'm so proud of you today" or "I like the living space that you added here in your doll house. What are you planning to do on the top floors? Good job. You're almost done." As shown through the case of Ruth, who was emotionally affected by a comment which took her several weeks to recover from and get back on track. The teacher provided her with encouragement and guidance to regain her confidence. She spent the last four weeks (and even some extra time at home) to finish her two-story dollhouse. Ruth was proud of her work, and was also rewarded with her client's satisfaction and happiness. In another example, Betty was disappointed by her slanted two-level dollhouse and wanted to make a pillow instead. She received feedback and encouragement from her peers and teacher that the house looked "cool" and "whimsical". After this reassurance, she decided not to start a new project, but figured out a way to improve her existing one. The teacher supported her by reframing her project and her classmates supported her by providing encouragement, and easing her disappointment so that she was able to progress past this mistake.

Without close support, female builders may feel demotivated and may give up on their intended project. This was the case for Panni, who chose to make a pretty pillow instead of a disproportionate shark-dolphin stuffed animal. It was also the case for other builders who switched from their clients' original requests to a pillow. Though they made personalized designs for their clients, these builders lost confidence in their ideas, became demotivated, and wanted to settle on less complex designs.

Conclusion

The intimacy and personal connection with their clients influence how the young female builders engaged in the processes of making. This connection with their clients make the construction process

“personally meaningful”, because the person for whom they were building was meaningful. Though these young builders have not yet developed to the “connected knowers” stage as defined by Belenky and her colleagues (1986), they have a tendency to value social interactions and a sense of community. Making for others is an effective exemplar of how we can leverage an epistemological approach that is cognitively and culturally unique to women in order to motivate female learners in making.

However, it is not enough to merely provide a space for girls to make. How we design the learning experience matters. The teacher needs to provide support. At the same time, facilitation and activities should also be designed to build a community of learners that supports one another. A better understanding of the values of diverse learners that make up our schools and makerspaces must be considered in order to create an inclusive maker-centered learning environment.

References

- Belenky, M. F., Clinchy, B. M., Goldberger, N. R., & Tarule, J. M. (1986). *Women's ways of knowing: The development of self, voice, and mind* (Vol. 15). New York: Basic books.
- Buechley, L. 2013. Thinking about making. Retrieved December 6, 2017 from <http://edstream.stanford.edu/Video/Play/883b61dd951d4d3f90abeec65eead2911d>
- Buechley, L. 2016. Inclusive Maker Education: STEM is Everywhere. Retrieved January 2, 2018 from <https://edstream.stanford.edu/Video/Play/a33992cc9fb2496488c1afa9b6204a571d>
- Dougherty, D. 2016. *Free to Make: How the Maker Movement is Changing Our Schools, Our Jobs, and Our Minds*. North Atlantic Books.
- Fisher, A., & Margolis, J. 2002. Unlocking the clubhouse: the Carnegie Mellon experience. *ACM SIGCSE Bulletin*, 34(2), 79-83.
- Gilligan, C. 1982. *In a different voice*. Harvard University Press.
- Halverson, E. R., & Sheridan, K. 2014. The maker movement in education. *Harvard Educational Review*, 84(4), 495-504.
- Holbert, N. (2016). Leveraging cultural values and “ways of knowing” to increase diversity in maker activities. *International journal of child-computer interaction*, 9, 33-39.
- Intel Corporation. 2014. *MakeHers Report: Engaging Girls and Women in Technology through Making, Creating, and Inventing*. Retrieved November 12, 2016 from <http://www.intel.com/content/www/us/en/technology-in-education/making-her-future-report.html>
- Kalil, Thomas. 2013. Have fun—learn something, do something, make something. *Design, make, play: Growing the next generation of STEM innovators*, 12-16.
- Margolis, J., & Fisher, A. 2003. *Unlocking the clubhouse: Women in computing*. MIT press.
- Maker Media. 2014. *Attendee Study Maker Faire Bay Area 2014*. Retrieved from http://makermedia.com/wp-content/uploads/2013/01/MFBA-2014-research-deck_FINAL.pdf
- Martinez, S. 2015. Making for All: How to Build an Inclusive Makerspace. Retrieved August 26, 2017 from EdSurge: <https://www.edsurge.com/news/2015-05-10-making-for-all-how-to-build-an-inclusive-makerspace>
- Miles, M. B. Huberman, A. M., & Saldana, J. (2014). *Qualitative Data Analysis: A Methods Sourcebooks*.
- Mosatche, H. S., Matloff-Nieves, S., Kekelis, L., & Lawner, E. K. 2013. Effective STEM programs for adolescent girls: Three approaches and many lessons learned. *Afterschool matters*, 17, 17-25.
- Papert, S. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- Peppler, K., & Bender, S. 2013. Maker movement spreads innovation one project at a time. *Phi Delta Kappan*, 95(3), 22-27.

Rosser, S. V. 1990. *Female-friendly science: Applying women's studies methods and theories to attract students*. Pergamon.

Simon, R. W., & Nath, L. E. (2004). Gender and emotion in the United States: Do men and women differ in self-reports of feelings and expressive behavior?. *American journal of sociology*, 109(5), 1137-1176.

Thoits, P. A. (1995). Stress, coping, and social support processes: Where are we? What next?. *Journal of health and social behavior*, 53-79.

Wilensky, U. 1991. *Abstract meditations on the concrete and concrete implications for mathematics education*. Epistemology and Learning Group, MIT Media Laboratory

Worsley, M., & Blikstein, P. (2016). Children are not hackers: Building a culture of powerful ideas, deep learning, and equity in the Maker Movement. In *Makeology* (pp. 78-94). Routledge.

The Construction of Knowledge in Maker Education: A Constructivist Perspective

José Armando Valente, *jvalente@unicamp.br*
State University of Campinas, UNICAMP, Brazil

Paulo Blikstein, Paulo, *paulob@stanford.edu*
Stanford University, USA

Abstract

The objective of this paper is to reflect on the contributions of makerspaces to the process of knowledge construction. Initially we discuss aspects related to the theory of constructionism, and, subsequently, using Piaget's notions of conceptualization, we discuss how knowledge can be constructed in a makerspace. We show that in makerspaces students can develop sophisticated artifacts by using digital technologies, and that besides the product, this process allows for the representation of the actions provided to these machines, expressed as concepts and strategies used. This representation constitutes the "window into the mind" of the learner, allowing for one to understand and to identify the knowledge used and, with that, help the learner reach a new level of knowledge construction.

Keywords

makerspaces; maker education; fabrication technologies; knowledge construction; constructionism

Introduction

Makerspaces are being introduced in K-12 education as an alternative to traditional approaches so that students can have more agency, engage in project-based learning activities, and generally be more active. In makerspaces, students learn how to produce artifacts by using traditional objects and materials combined with digital fabrication technologies, which are increasingly present in the contemporary world (see Figure 1 for examples of students' projects.) These activities are based on the constructionist approach to learning proposed by Papert (1986) and are being inserted in education so that learners can develop objects of their interest and, with this, explore and build knowledge about various curricular concepts.

Seymour Papert and collaborators developed, in the late 1960s, the Logo programming language, which aimed to allow children to "teach" the computer, an activity that, according to these researchers, would be much more efficient than passive teaching strategies used in the traditional classroom. Papert called the approach through which the learner constructs knowledge when s/he produces an object of her/his interest, such as a work of art, a report, or a computer program, *constructionist* (Papert, 1986). Papert emphasized the importance of learning through the "hand" on and the "head" in process: the learner is involved in building something of her/his interest, and in doing so, is faced with unexpected problems for which there is no pre-established explanation.

This belief in the development of an increasingly complex and multidisciplinary problem-solving capacity in students brings Papert's constructionism ideas closer to the current maker movement, which seeks to take protagonism and technological innovation to learning spaces.

A central aspect of a makerspace or digital fabrication lab is the construction of objects using different materials such as scrap, wood, cardboard, electromechanical and electronic components, which can be combined with computer programming activities and the use of fabrication tools such as a laser cutters and 3D printers. The emphasis is on promoting engagement and a strong sense of experimentation with media and the materials, while constructing knowledge, collaborating, and building a learning community. Making involves trying to solve a specific problem, creating a physical or digital artifact, and sharing that product with the public. The interaction between participants and the process of knowledge sharing is often mediated by social media, as well as online repositories of objects, tools,

and “how-to” manuals. Much of the spirit of maker labs resonates with the “do it yourself” (DIY) culture, as Hatch states:

Making is fundamental to what it means to be human. We must make, create and express ourselves to feel whole. There is something unique about making physical things. The things we make are like little pieces of us and seem to embody portions of our soul. (Hatch, 2013, p.11)

Despite issues regarding equity of participation and culture mismatch (Blikstein & Worsley, 2016), makerspaces have great potential to contribute to progressive education and to create multiple paths for students to learn topics that are more relevant to them. Researchers have been suggesting that making, associated with learning methodologies such as constructionism, can create conditions for student to be creative, critical, as well as able to solve problems and to work in groups (Martinez & Sager, 2013; Halverson & Kimberly, 2014; Kurti, Kurti & Flemming, 2014).

In many maker labs, the focus is on building a product, and learning how to operate different machines and devices. However, when something is produced, multiple ideas and concepts that the learner already has are put into action. This knowledge goes beyond technical skills and may involve disciplinary content or can be constructed as learners interact with their objects and machines. *However, through trial and error, a product can be successfully constructed without the learner necessarily being able to understand all the concepts involved in the process.*

Piaget studied the development of certain concepts, which are constructed as the result of the interactions between the learner and everyday objects or people; process that Papert called “Piagetian learning” or “learning without being taught” (Papert, 1980, p.7). Other researchers such as Vygotsky, for example, understood that the construction of scientific concepts does not result from the simple interaction between the learner and objects, or is a natural result of the development of “hands-on” activities. The learners’ construction of knowledge goes to a certain point, and from then on, no matter how much effort the learner makes, the content cannot be assimilated. The learner needs the help of a more experienced colleague or a specialist, who will assist in the construction of these new concepts (Vygotsky, 1989).

This article aims to understand educational makerspaces and how these contribute to learning; discuss the theories underlying knowledge-building processes, especially regarding hands-on activities; trying to understand how knowledge can be represented and conceptualized in makerspaces.

Makerspaces and education

From the point of view of technological diffusion, the maker concept has its roots in the Mechanics’ Institutes, created in Edinburgh, Scotland, during the beginning of the 19th century for the provision of technical education for craftsmen, professionals, and workers in general. These institutes have revolutionized access to science and technology education (Holman, 2015). With the dissemination of digital technologies, the 1980s and 1990s saw the creation of the hacker movement, or hackerspaces, in several cities across the United States and Europe. These were places where technology enthusiasts could work together to invent devices, reuse and exploit new technologies such as low-cost microcontrollers, and were inspired by the open software community (Blikstein, 2018). In this context, the term “hacker” does not refer to the transgression of rules, but rather describes the use of existing everyday objects to understand a phenomenon, or the production of new objects or systems. A classic example is the disassembly of electronic devices and the reuse of their parts for the creation of new appliances.

From the educational point of view, the interest in a student-centered or learning by doing based education is not new either. One of the first educators to use this pedagogical approach was Dewey during the beginning of the last century. This author criticized expository teaching as being old-fashioned and ineffective, and proposed the implementation of hands-on based learning situations (Dewey, 1916). Other educators and thinkers such as Freinet (1998), Montessori (1965), and Freire (2008) have devoted special attention to the relationship between mind and artifact-production as part of the educational process. More recently, during the first decade of this century, new trends in educational, social, economic, and technological character have contributed to the growth of these movements into

formal and non-formal educational environments, such as schools, museums, and makerspaces in communities.

According to Blikstein (2018), the interest in the creation, dissemination, and popularization of makerspaces can be attributed to five trends: the greater social acceptance of ideas and principles of progressive education; countries' interest in establishing a base for an innovative economy; the growth of public awareness, in addition to the popularity of computer programming combined with the creation and production of artifacts; the sharp reduction in the cost of digital information and communication technologies (DICT), as well as digital fabrication technologies (DFT); and the development of tools that are more powerful and easier for students to use, along with studies and publications in academic research focused on the effect and impact of these new technologies on learning.

Since 2005, makerspaces have gained great popularity as a result of the emergence of the broader "maker movement" (Anderson, 2012), the publication of Make Magazine, and the first Make Faire in 2006, idealized by Dale Dougherty (2013). In addition, these spaces received a great deal of attention from educators and researchers after the former US President, Barack Obama, launched an initiative to promote learning environments that "encourage young people to create and build and invent - to be makers of things, not just consumers of things" (The White House, 2009).

Papert's constructionist ideas are the rationale behind the dissemination of making in schools, since, in these spaces, learners can learn from hands-on and "heads-in" experiences. Several researchers and research groups focused on this area of study have emphasized that students use different concepts throughout the activities developed in these spaces (Martinez & Stager, 2013; Halverson & Kimberly, 2014; Kurti, Kurti & Flemming, 2014).

However, before using constructionism as a conceptual basis for the creation of the maker activities, it is relevant to understand the context in which this concept was developed in the mid-1980s. First, researchers believed it was important to introduce an alternative to the uses of computers in education, which at the time were still totally focused on the idea of transmitting information through tutorials, or exercise and practice programs, which Papert called "instructionism" (Papert, 1991, p.8). Second, for Papert, constructionism builds on constructivist theories: "this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" (1991, p. 1). The emphasis was, therefore, on the fact that learning is not only the result of the learner's interaction with objects and people around her/him, as proposed by Piaget's constructivism, but the result of the learner's engagement in the construction of something of her/his interest, which can be done with or without the use of computers. Papert makes it clear that "computers figure prominently only because they provide an especially wide range of excellent contexts for constructionist learning" (1991, p. 8). Perhaps when these ideas were proposed it was not as important to emphasize the presence of computers, since they were not yet widely disseminated, and learning was not centered on the "connections between computers and real-world artifacts" (Donaldson, 2014, p.7). Third, constructionism as a theory required further elaboration. Although the concept is "much richer and more multifaceted, and very much deeper in its implications" (Papert, 1991, p.1), Papert even went so far as to comment on the irony that "it would be particularly oxymoronic to convey the idea of constructionism through a definition since, after all, constructionism boils down to demanding that everything is understood by being constructed" (p.2).

The presence of digital technologies as part of constructionism was further elaborated by Edith Ackermann, as she differentiated constructivism from constructionism by proposing:

1. The increased role of external aids in learning and development;
2. The emphasis on digital and technological aids;
3. The learner's hands-on initiative for a learner takes in the creation of tools, objects, or knowledge (Ackermann, 2001, p. 5).

Digital technologies become important when they go beyond aiding in the production of a product, and, rather, *make explicit the actions that one must carry out during the process of developing an object*. The ability to explain one's actions is very different from what takes place during the production of something using traditional objects. *It is one thing to be able to produce a sand castle or a vase from a clay tusk.*

Another thing is to provide information so that a robot can produce the same sand castle or vase. In the case of the robot, in addition to the product, one must be able to represent the actions the robot must take so that the product can be produced. These actions are described as concepts and strategies created by the learner. This representation can be studied and analyzed in terms of the concepts and strategies used and be improved or debugged for production efficiency. In fact, this representation constitutes a “window into the mind” of the learner, in the sense that it allows for one to understand and to identify the common-sense knowledge that was used during the production process and, with that, help the learner reach a new level of scientifically-based knowledge that is a product of a growing learning spiral (Valente, 2005).

In this sense, to create an educational makerspace, it is important to consider, in addition to traditional objects of construction, digital information and communication technologies such as computers, digital cameras, as well as fabrication technologies such as 3D printers, laser cutters and computerized numerical control milling machines. These technologies should not only be part of the makerspace because they are innovative and part of advanced production processes, but also because of the role they play in making the concepts and strategies learners use to develop the artifacts they produce explicit. For these technologies to function they need to be programmed using concepts such as scale, distance, geometry. Furthermore, the learner must develop different strategies to apply these concepts in the “program.” Lastly, as noted by Riley (2015), technologies add precision, scalability, and reproducibility to the students’ work, as shown in Figure 1.

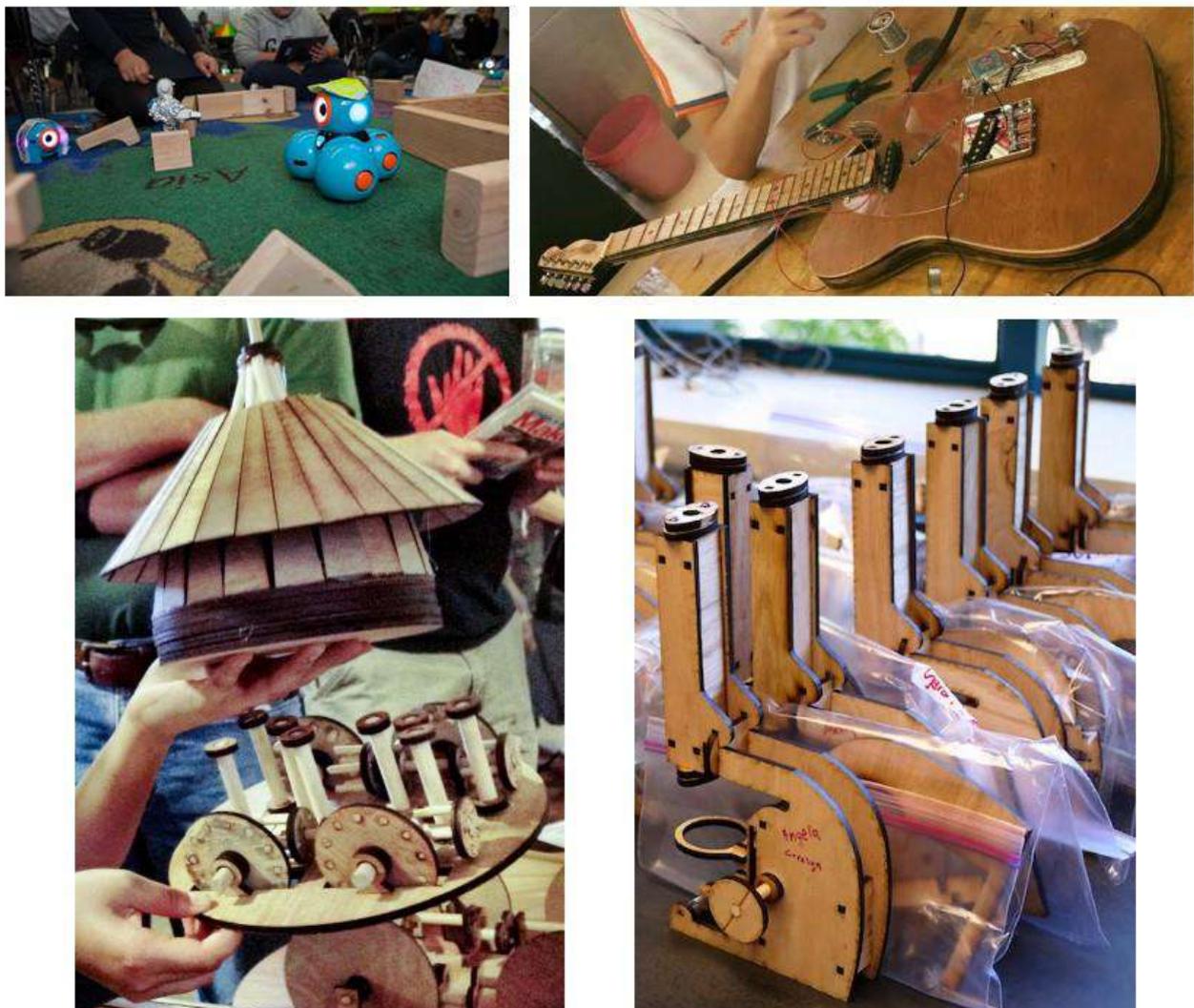


Figure 1. Examples of students’ projects (clockwise): a robot-enacted theater play (top left), a custom-made guitar, a daVinci machine, and a microscope (bottom right)

The tasks that can be performed in makerspaces, particularly using digital technologies, allow learners the possibility of working with concepts from several knowledge areas, such as subjects in standard curricula. Riley (2015), while analyzing students' use of fabrication technologies, identified that students had the opportunity to develop mathematical concepts such as Cartesian coordinates for the transposition of 2D shapes into 3D figures and vice versa, geometric shapes, units of measure, scale, Boolean operations, etc. The production of artifacts using a combination of traditional materials and digital technologies makes it possible for learners to use concepts from other areas such as science, engineering, and technology.

In addition to these concepts, several authors mention that makerspaces promote personal and social development. For example, Clapp et al. (2017) identified the development of agency (a more proactive orientation towards the world) and character building in makerspaces. The learner can take risks, cope with failures to achieve success, and develop a mindset which includes creativity, curiosity, mental openness, persistence, social responsibility, and teamwork.

However, the lack of a deeper understanding of constructionism, of the role digital technologies play in these environments, and of a more precise definition of what constitutes an educational makerspace, have contributed to several misunderstandings. First, makerspaces set up in schools are quite heterogeneous, varying in terms of size, capacity, and cost. Some schools have understood that simply having a room with tables, traditional materials, and glue guns is enough, while other schools offer spaces with the most sophisticated digital fabrication technologies (Blikstein, 2018). It is crucial to understand the role that technologies play in these spaces and to seek to balance the composition of traditional materials and digital technologies.

Second, educational makerspaces in schools should be understood as spaces for knowledge production. In this sense, it is important that they are not seen as environments for the development of isolated activities, but activities that are integrated with curricular disciplines. It is not enough to create makerspaces in which learners can be creative and have agency, while curricular subjects are still introduced in a traditional way. Third, for the learner to construct knowledge in the makerspaces, it is important that a series of issues are observed. The elaboration of a product is fundamental, as Papert emphasized. However, the production process and the analysis of representations, which provide the opportunity for one to understand the concepts and strategies used by the learner, are also important. Thus, the fact of *producing something is not enough to ensure that the learner has constructed knowledge*. The teacher's role is fundamental to mediate processes and product development, to create opportunities for reflection, and to develop the learner's awareness of the concepts and strategies that are used, as observed by Piaget and Vygotsky.

Knowledge construction in the maker activities

In his investigations, Piaget identified three types of knowledge that an individual constructs: physical knowledge (constructed through the direct action of the individual with the object), logical-mathematical knowledge (fruit of a reflection regarding the information collected at a practical level, generating the conceptualization), and social-arbitrary knowledge (constructed through the interaction with other people in society) (Matui, 1995). However, it is the development of logical-mathematical concepts that has received the most attention from teachers and been implemented in the learning process, since these concepts depend on the ability for abstraction and their development must be aided by educators.

Vygotsky makes a similar distinction regarding the construction of concepts. He distinguishes spontaneous concepts from scientific ones. The first is developed based on the individual's experience in the world in which s/he lives, and with the world organization imposed by society; while as scientific concepts are developed from spontaneous experiences, but fundamentally depend on social interaction and on the presence of more experienced people or the school environment (Vygotsky, 1986).

Vygotsky was concerned with the study of how to provide the means for the construction of knowledge. He makes an important distinction between development and learning. *Effective or real development* can be understood as all the knowledge the learner has already constructed. *Potential development* is

what the learner can achieve during the teaching and learning process - understood here to be the literal translation of the Russian term *obuchenie*, which involves the learner, the person who teaches, and the relationship between these pairs that are subjects of the educational process (Matui, 1995). Therefore, learning is what allows for the transition from *real development* to *potential development*. Between these two levels is the area or zone of proximal development (ZPD) where teaching must take place, since “the only good teaching is what advances to development” (Matui, 1995, p. 121).

Papert emphasizes the importance of enriching learning environments by incorporating digital technologies, so that subjects can act and construct concepts and ideas that permeate these environments (Papert, 1980). The use of these technologies requires logical-mathematical concepts and the interaction with these concepts becomes a way to stimulate “Piagetian learning”. However, constructing knowledge about these concepts does not happen without the help of more experienced people, as emphasized by Vygotsky (1986).

From this brief analysis of the ideas proposed by notable socio-interactionist authors, one can see that the development of spontaneous concepts, or even some kind of logical-mathematical or social-arbitrary knowledge, can be achieved through “Piagetian learning”. For learners to be able to develop scientific or logical-mathematical concepts, however, the help of more experienced people who understand the process of how to promote learning and the content being studied is necessary – there is a need for a “true” educator, in the literal sense of the word. However, one cannot assume that simply providing information or completing a task is sufficient for constructing knowledge.

The evaluation of teaching and learning processes is still based on the idea that the student has learned a concept if s/he is able to successfully apply it, or is able to talk about the acquired information. However, the fact that the learner succeeds at performing a task does not necessarily mean that s/he understands what was done. *Piaget noted that there is a difference between doing something successfully and understanding what was done.*

In 1974, Piaget published two books: *La Prise de Conscience* (translated into English as “Grasp of consciousness: action and concept in the young child”, 1976) and *Réussir et Comprendre* (translated to English as “Success and Understanding”, 1978). These described the process by which children and adolescents develop what he called “conceptualized understanding” of the concepts involved in a series of tasks, which Piaget asked the subjects of his research to perform.

In these studies, Piaget noted that *children can use complex actions to achieve premature success*, which represents all the characteristics of *savoir faire*. The child can perform a certain task but not understand how it was performed, nor be mindful of the concepts involved in the task. Piaget also noted that the passage from this practical form of knowledge to understanding is done through the grasp of consciousness, which is not a kind of insight, but a level of conceptualization. This level of thinking is achieved thanks to a process of transforming schemes of action into notions and operations. Thus, through the coordination of more complex concepts, the child can move from the level of premature success to a level of conceptual understanding, which takes place in three phases. In the first, the child neglects all the elements involved in the task; in the second, s/he coordinates some elements, and in the third, s/he coordinates all the elements involved in the task.

Besides this succession of phases, Piaget first observed that it is not the object that leads the child to the comprehension phase. Being able to understand how to topple a sequence of dominoes does not necessarily mean understanding how to make a castle with playing cards. For each situation, the child must transform the action schemas into notions and operations that are involved in a given task. Piaget also noted that understanding is the fruit of the quality of the interaction between the child and the object. If s/he has a chance to play with objects, to reflect on the results obtained, and to be challenged by new situations, the greater is the chance that the child will be attentive to the concepts involved, and, thus, reach the level of conceptualized understanding.

In the case of working with digital or fabrication technologies in the makerspaces, learners can explore, create, and reflect in a very stimulating and innovative environment. However, from the educational point of view, it is impractical to think that they will be able to construct knowledge individually, without being aided by others. First, it would be too costly to construct learning environments involving concepts from all the existing domains so that an individual could act in this environment and construct her/his

knowledge in isolation. Second, as an educational solution this model is not practical, because the time needed to train people with the knowledge already accumulated by humanity would be enormous. In this sense, the idea of knowledge construction can be improved if we have teachers prepared to help students (Piaget, 1998) or, as Vygotsky proposes, through more experienced people who can help formalize concepts that are historically agreed upon (Vygotsky, 1986). Without the presence of an educator it would be necessary for the learner to recreate these conventions.

Conclusion

There is a great interest in introducing maker activities in K-12 education, so the students can have more agency, engage in project-based learning, and be generally more active in the learning process, learning to produce artifacts by using traditional and digital materials. This article argues that it is possible to create learning environments that are based on Papert's constructionist ideas using activities in which the learner can develop objects of their interest and, with this, explore and construct knowledge about various curricular concepts, especially those related to STEM.

The analysis of constructionist ideas indicates that Papert has emphasized the production of objects as a way for learners to express their ideas. However, as proposed by some researchers, production should take place using digital technologies, which besides the product also allows for the representation of the actions provided to these machines. These actions are registered as the concepts and strategies the learner used, which can be analyzed and debugged. These representations constitute a window into the learner's mind allowing teachers or a more experienced person to help the learner construct new knowledge.

The fact that the learner is working with digital technologies in the makerspace allows for the representation of the action s/he is using, or her/his knowledge, in addition to the creation of the product. This means that digital technologies play an important role in makerspaces. Furthermore, since the makerspace is created in the school, it is important to integrate activities students develop to other curricular content.

Finally, according to what was discussed in terms of some ideas proposed by Piaget and Vygotsky, it is important that the implementation of the makerspace consider the presence of teachers or a more experience person whose function is to work with the students so as to challenge them, to create conditions for them to interact with the object being produced, and to help them understand the concepts and strategies used. Through these interactions with the students, teachers can help students construct new knowledge, as well reach a higher level of comprehension about what they are doing.

However, for this type of setting to take place in a school it is necessary to change the relationships taking place in the learning environment and to determine new roles to be assumed by the different professionals who work in the school. This means implementing changes in the relationship between people, and the quality of the students' interactions with the objects and activities performed. As observed by Piaget, the fact that the learner can make an artifact or can successfully arrive at an answer does not necessarily indicate that knowledge was constructed. The learners must also to be able to conceptualize what was produced, which allows for the transformations of their mental schemas.

The solution to education that prioritizes understanding is the use of stimulating objects and activities, so that the student can be involved with what s/he is doing. These objects and the tasks should be rich in opportunities that allow the student to explore, and must create the conditions for the teacher to challenge the student and, thereby, increase the quality of the interaction with the product. Also, as proposed by Vygotsky, the learner needs to get help from more experienced people. Without this type of support the learner must recreate knowledge and conventions that are already available. However, for teachers to be able to support and help the student in the makerspace it is necessary that they receive training not only in terms of how to use technologies, but also regarding how to integrate the activities the students are developing with the disciplines in the curricula. The analysis of literature on makerspaces has shown that this integration has not fully happened yet—we still have a long way to go in the process of creating makerspaces in schools for knowledge construction.

Acknowledgements

This work is supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico, (CNPq), Brazil, grant 306320/2015-0; by Fundação de Amparo à Pesquisa (FAPESP), Brazil, grant 2015/16528-0; and by the Lemann Center for Entrepreneurship and Educational Innovation in Brazil at Stanford University, USA.

References

- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference? *Future of Learning Group Publication*, 5(3), 438.
- Anderson, C. (2012). *Makers: The new industrial revolution*. New York: Crown.
- Blikstein, P. & Worsley, M. (2016). Children are not Hackers: Building a culture of powerful ideas, deep learning, and equity in the Maker Movement. In K. Peppler, E. Halverson, & Y. Kafai (Eds.), *Makeology: Makerspaces as Learning Environments* (Volume 1) (pp. 64-79), New York, NY: Routledge.
- Blikstein P. (2018). Maker Movement in Education: History and Prospects. In: de Vries M. (Ed.) *Handbook of Technology Education*. Springer International Handbooks of Education. Springer, Cham.
- Clapp, E. P., Ross, J., Ryan, J. O. & Tishman, S. (2017). *Maker-Centered Learning: empowering young people to share their worlds*. San Francisco: Jossey Bass.
- Dewey, J. (1916). *Democracy and Education*. New York: The Free Press.
- Donaldson, J. (2014). The Maker Movement and the rebirth of constructionism. *Hybrid Pedagogy*, 1-19. Retrieved from <http://hybridpedagogy.org/constructionism-reborn/>
- Dougherty, D. (2013). The Maker Mindset. In: M. Honey & D. E. Kanter (Eds.). *Design, Make, Play: Growing the Next Generation of STEM Innovators*. London: Routledge.
- Freinet, C. (1998). *Educação pelo trabalho*. São Paulo: Martins Fontes.
- Freire, P. (2008). *Pedagogia da autonomia: saberes necessários à prática educativa*. 37.ed. São Paulo: Paz e Terra.
- Halverson, E. R. & Kimberly, M. S. (2014). The Maker Movement in Education. *Harvard Educational Review*, December 2014, Vol. 84, No. 4, pp. 495-504. Retrieved from: <http://hepgjournals.org/doi/10.17763/haer.84.4.34j1g68140382063>.
- Hatch, M. (2013). *The Maker Movement Manifesto: Rules for Innovation in the New World of Crafters, Hackers, and Tinkerers*. New York: McGraw-Hill Education.
- Holman, W. (2015). Makerspace: Towards a new civic infrastructure. *Places*. Retrieved from: <https://placesjournal.org/article/makerspace-towards-a-new-civic-infrastructure>.
- Kurti, R. S., Kurti, D. I. & Fleming, L. (2014). The Philosophy of Educational Makerspaces: Part 1 of Making an Educational Makerspace. *Teacher Librarian: The Journal for School Library Professionals*, June 2014. Retrieved from: www.teacherlibrarian.com/2014/06/18/educational-makerspaces/.
- Martinez, S. L. & Stager, G. (2013). *Invent to Learn: Making, Tinkering, and Engineering in the Classroom*. Santa Barbara: Constructing Modern Knowledge Press.
- Matui, J. (1995). *Construtivismo: teoria construtivista sócio-histórica aplicada ao ensino*. São Paulo: Editora Moderna.
- Montessori, M. (1965). *Spontaneous activity in education*. New York: Schocken Books.
- Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.
- Papert, S. (1986). *Constructionism: A new opportunity for elementary science education*. A proposal to the National Science Foundation, Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, Massachusetts.

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.) *Constructionism*. Norwood, NJ: Ablex Publishing Corporation. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>.

Piaget, J. (1976). *Grasp of consciousness: action and concept in the young child*. London: Psychology Press.

Piaget, J. (1978). *Success and Understanding*. Cambridge, Mass: Harvard University Press.

Piaget, J. (1998). *Sobre Pedagogia*. São Paulo: Casa do Psicólogo.

Riley, E. (2015). What do people learn from using digital fabrication tool? In P. Blikstein, S. L. Martinez & H. A. Pang (Eds.) *Meaningful Making: projects and inspirations for fab labs and makerspaces*. Torrance, CA: Constructing Modern Knowledge Press.

The White House (2009). Office of the Press Secretary. *Remarks by the president at the national academy of sciences annual meeting*. Retrieved from: <https://www.energy.gov/articles/remarks-president-national-academy-sciences-annual-meeting>.

Valente, J. A. (2005). *A Espiral da Espiral de Aprendizagem: o processo de compreensão do papel das tecnologias de informação e comunicação na educação*. Unpublished thesis. Universidade Estadual de Campinas (UNICAMP), Campinas, São Paulo, Brazil. Retrieved from: <http://www.bibliotecadigital.unicamp.br/document/?code=000857072&opt=4>.

Vygotsky, L.S. (1986). *Thought and Language*. Cambridge: The MIT Press.

Constructionism in Different Cultures: the Case of Brazil

José Armando Valente, jvalente@unicamp.br
State University of Campinas, UNICAMP, Brazil

Paulo Blikstein, paulob@stanford.edu
Stanford University, USA

Abstract

This paper starts with the conjecture that constructionism proposed by Papert is not used in the same way in different countries and cultures. Based on Piaget's ideas it is possible for a concept to be assimilated or accommodated in terms of mental structures that were constructed based on the experiences researchers have by living in their respective cultural contexts. Thus, it is unlikely that the constructionist ideas proposed by Papert are used in the same way as in the USA, Brazil or Thailand. In Brazil, the educational context has been heavily influenced by the ideas of Paulo Freire, considered the Patron of Brazilian Education. We have several examples to show that constructionism in Brazil was influenced by the ideas of Freire, as well other factors which are particularly related to the educational system and the country social and economic situation.

Keywords

International education, contextualized knowledge, Brazilian education, Freire, Constructionism

Introduction

This paper starts with the conjecture that the "construction" of constructionism as an educational philosophy by researchers in education happens according to the assimilation and accommodation processes described by Piaget. In this case, the idea of constructionism (or any other educational idea) might be assimilated in terms of mental structures that were constructed based on the experiences that researchers had by living in their respective cultural contexts. It is likely, thus, that constructionism was assimilated and accommodated according to the influences of the different cultures. The amalgam of constructionism and knowledge constructed in these contexts may constitute a new theoretical basis that support projects and activities related to the use of new technologies in education in different countries or cultures. Thus, it is unlikely that the constructionist ideas proposed by Papert are used in the same way as in the USA, Brazil or Thailand.

In this paper, we intend to examine the specific characteristics of constructionism in Brazil, given its educational culture and academic tradition. *One key element here is that in the case of Brazil, constructionism was heavily influenced by the ideas of the well-known Brazilian educator Paulo Freire.* This can be seen in the research developed at different centers that are related to this area, such as the Nucleus of Informatics Applied to Education (NIED) from the State University of Campinas (UNICAMP), the Teacher Training and New Technologies of the Graduate Program in Education (Catholic University of São Paulo-PUC/SP), from the Laboratory of Cognitive Studies (LEC) at the Federal University of Rio Grande do Sul (UFRGS) and others.

Freire on awareness

According to Freire (1980), the power of transformation is directly related to raising the subject's awareness, understood as a reflective attitude of the subject's own condition and the condition of his/her social practices in the world. Freire believes that the subject may reach three levels of awareness: a primitive level, in the sense of responding automatically without being aware of the situation; the awareness of the external actions when the subject starts perceiving the social context; and the self-conscious level, when the subject clearly associates the social and historical context with his/her own

current reality. Literacy practices that manage to reach the latter level of awareness will bring about the most significant transformations that can cause impact on the cognitive, perceptive realities and on the social and cultural context of the subject.

Other factors

But Paulo Freire was not the only factor that influenced the adoption of constructionism in Brazil, as we will discuss in this article. For example, there was a sense that technology was an expensive investment, and that schools had much more urgent needs such as offering better salaries for teachers, improving the school lunch, fixing leaking roofs or buying new chairs. A second element, which is connected to the last point, was that the high perceived price of technology made teachers and school administrators heavily control the access to computers, computer labs, and other technologies, creating labs that were highly regimented and with a plethora of tacit and explicit rules for usage. A third element was the idea that technology was associated with “imperialistic” exploitation schemes coming from foreign countries, the US in particular.

Thus, we show that the work developed by the authors based on the constructionist approach and implemented in different contexts in Brazil had not only principles from the Freire’s ideas, but were also influenced by a wide variety of factors that were specific to Brazil.

Case studies and examples

“Contextualized constructionism:” a concept developed by NIED

NIED was created under strong influence of Papert and Minsky’s ideas. They visited UNICAMP in two occasions, July of 1975 and 1976. In 1975 they developed several workshops about Logo ideas. This was done using only a blackboard since at UNICAMP at that time there was no program to interpret and execute the Logo commands. A group of researchers from UNICAMP visited the MIT Logo Laboratory in February-March 1976. When this group returned to Brazil, the Logo interpreter was deployed to the PDP 10 computer, which allowed the beginning of the first work on the use of Logo with children. This experiment was performed with children of UNICAMP teachers and used the only cathode ray terminal connected to the PDP 10 computer. Papert and Minsky returned to Brazil in July 1976 to teach seminars and participate in the activities of the research group on the use of Logo in education that had been established. These experiences and studies gave rise to Maria Cecília Calani’s master dissertation (Calani, 1981), and later this research group was consolidated with the creation of NIED in May 1983.

Valente has been associated with NIED since its creation, was its director during the period of 1986 to 2003 and is still part of this research group. Through these years NIED research approach has gone through basically four phases: use of Logo in activities related to public schools, use of online technologies to promote educators’ professional development, use of mobile technology in schools, and more recently exploring the use of ubiquitous technologies in educational settings.

After the creation of NIED, the first research project developed was related to the Educom Project, which took place in five universities: Federal University of Pernambuco (UFPe), Federal University of Minas Gerais (UFMG), Federal University of Rio de Janeiro (UFRJ), UFRGS and UNICAMP. This project considered the diverse uses of the computer in different pedagogical approaches, including educational software development and the use of the computer as resource for problems solving. The objective of the project developed by NIED was to integrate the Logo methodology into the high school curriculum of Mathematics, Sciences and Portuguese in three public schools in the state of São Paulo

In the projects and activities developed by NIED, Papert’s ideas was referred as Logo methodology, Logo aesthetics or Logo environment. Only after the creation of the constructionism term in 1986, activities developed by NIED incorporated the term. It was used to describe the cognitive aspects of the learner’s knowledge construction. However, it was important to consider that the learner was also part of a social and cultural environment built locally by colleagues and teacher, and globally by parents, friends or by the community where s/he lives. S/he could use all these social and cultural elements as a source of ideas and information or a place to find problems to be solved by the computer.

The social aspect was included during the work that was carried out by NIED researchers as part of a project developed by the Secretariat of Education of the Municipality of São Paulo, implementing information technology in their school system. The secretary at that time (1989-1991) was Paulo Freire. One of the pedagogical recommendation that was proposed by his group of educators was to relate the theme of the project the students were developing to their cultural identity or culture context. In this case, the community should be the source of problems to be solved through the computer, and solutions and knowledge constructed by the learner should return to the community in the form of some improvement to be implemented.

Based on this experience NIED researchers created the contextualized constructionism concept in which the contextualized aspect is related to Freire's ideas. The rationale was that if the products the learners are producing are related to their interest and context in which they live, the better the chance of them understanding about the content and getting involved in the educational activities of producing this product (Freire, 1975). The contextualized constructionism concept was used to support various projects developed by NIED such as teacher training, training of factory workers, adult ICT literacy, and one laptop per student project (Valente, 2005).

Teacher training

The process of preparing teachers for the use of ICT in their pedagogical activities was developed by NIED through several online courses and using the contextualized constructionist approach (Valente, 2003). One of these courses was to prepare teachers from special education to be able to use ICT in their classroom activities. One course was part of the Project of ICT in Special Education (*Projeto de Informática na Educação Especial – PROINESP*), sponsored by the Special Education Secretary from the Ministry of Education Office and by the National Federation of APAEs (FENAPAEs). The objective was to prepare special education teachers from APAE (*Associação de Pais e Amigos dos Excepcionais*) a non-government organization of parents of special needs children, and other special education institutions in different regions in Brazil to be able to use ICT in their classroom activities.

The goal of this online course was to promote the development of a reflective teacher, emphasizing reflection-in-action and reflection-on-action and working with contextualized and decontextualized knowledge. These different reflections were elaborated at different levels: about technical aspects of ICT, about the use of ICT to develop their own projects, and about how to use ICT with their students. The contextualized aspect was related to the process of knowledge construction which was grounded on each teacher's reality in terms of previous experience, classroom students and school setting. The decontextualized aspect had to do with each teacher capability to go beyond his/her contextual knowledge and to be able understand and discuss with other course participants about the knowledge they constructed based on their different contexts.

Since the courses emphasized the development of reflective teachers and provided means for them to understand the use of ICT activities in different learning contexts, teachers were acquiring means to interact with their students, discussing the concepts involved in their activities as well as how knowledge could be constructed based on different realities, whether regional or types of disability.

Training of factory workers

In 1991, NIED researchers started to interact with a local factory in the Campinas region with the objective of using computers for the creation of learning environment for training their assembly line workers. The first part of this project was dedicated to understanding the actions that the workers performed in their jobs and the concepts involved in these actions. On the basis of this understanding and with workers that the company designated to be part of the project, were developed various software like the "Target Game," whose objective was to explore the concept of Statistical Process Control (Fernandes, Furquim & Baranauskas, 1996); "Enxuto," which allowed modeling and simulation of manufacturing systems (Borges, Borges & Baranauskas, 1995); and "Jonas," a system whose function was to support training in manufacturing processes (Borges, Borges & Baranauskas, 1995). These software and systems were based on the contextualized constructionist approach. The workers could construct models for a given problem, involving specific concepts related to production, propose

experiments based on the knowledge that perform this experiment on the computer, and observe and analyze the results obtained.

For example, the Target Game was used in a study to understand the effectiveness of the training process, as part of a doctoral thesis (Schlünzen, 2000). Some workers were trained to be facilitators in the training process since they were very knowledgeable about the concepts related to statistical quality control process and about the difficulties their assembly line colleagues had with these concepts. They were responsible for training their colleagues and with this methodology it was possible to work with all employees from different production sectors. The results showed that this training process not only contributed to improved performance of the workers in their respective workplace, as it also resulted in a modest but significant increase⁵⁶ in the productivity of the company in general. During the period of this study the company did not contract other workers or acquired more efficient equipment. The fact that this project was conducted with factory workers, and the ideas of (positive) social and hierarchical disruption within the workplace clearly resonate with values and ideas coming from Paulo Freire.

Adult ICT literacy

This study was associated with the investigations conducted at the Multidisciplinary Research-Action Healthy Community Laboratory – LIPACS a joined effort with the Research Group “Culture, Society and Media” of the Institute of Arts of the State University of Campinas – UNICAMP. The objective was to conduct studies and surveys on the use of ICT applied to the teaching and learning processes and on the impact of these technologies on society, on the population regarded as excluded (Maia, 2011).

The study included 16 senior citizens, among whom 14 were women and 2 were men. Their schooling ranged from 3 and 7 years of Basic Education, and all of them were regarded as digitally excluded. Practical activities were used using sites of interest for the group, emails, websites to access audiovisual files and social network platforms. The actions were based on social and historical concepts and were supported by the strategy of dialog with the learners. The purpose of this research was to understand how senior citizens were empowered with technologies to build new literacies and, with that, become aware of their capacity to function in and to transform the context where they live so to be able to build new realities.

For example, in one of the groups studied, gardening became the generative theme. The activity started with a presentation on gardening delivered by one of the participants. This presentation was an integral part of the ICT activities planned for this learner. In addition, the learner was in the process of acquiring reading and writing skills and did not feel he fit in the group. He became the leader the moment he made the presentation of his own professional activities as a gardener. Figure 1 shows the participants visiting a garden, being guided by this learner who is learning about the digital camera.



Figure 1. Participants visiting a garden and one of them learning about a digital camera

With his self-esteem boosted, he fascinated the whole group with the gardening topic. Mediators took advantage of the moment to encourage debate among learners so that, based on what was observed, discussions were raised on how gardens related to each person’s own life. Digital technologies also contributed to the activities. They could visit websites and blogs, register their work using images, took pictures of the garden they visited and of meaningful objects found in their own home, or objects which are or used to be part of their lives to represent the garden itself. Messages with the images were

⁵⁶ The increase, as measured by the authors at the time, was 5%.

emailed to friends and relatives. Through their work it was possible to see increased awareness of the social use of the ICT to the degree of even transforming the context of their own lives. These changes were noticed both in the intellectual aspect and in relation to ICT, as well as in terms of practices in specific social contexts.

One laptop per student project

One of the projects developed was the Inquiry Based Learning Project (Project ABInv), with the objective to study the implementation of an inquiry-based learning approach, so that teachers and students could be engaged in "doing science", using features of the laptop in a 1-1 situation. The project was developed in three public schools in the state of São Paulo. The methodology used in this study was action research. NIED researchers worked with the teachers and students to define the thematic proposals for the development of the investigations based upon the school curriculum.

The role of the researchers was, primarily, to encourage and assist the teachers' development so that they could adopt a creative and pedagogic view of the appropriation of the laptops according to this new inquiry-based approach. The teachers learned how to work with students to raise questions related to the curriculum; how to decide together with the students what is a good question to be investigated; how to create practical, implementable experiments; how to collect data to respond to the questions under investigation; and how to analyze and present the results from the experiments in order to respond to the questions they raised. The students explored the mobility offered by the laptops, working in alternative special configurations within the classroom, in other internal spaces within the school, as well as spaces outside the school building. They were encouraged to develop collaborative projects, dividing up the tasks so that the experiments could be executed, and the data recorded and analyzed.

The results from the Project ABInv were recorded in the book, *ABInv – Aprendizagem Baseada na Investigação* (Valente, Baranauskas & Martins, 2014). In one of the schools it was possible to involve all the students from each of the first to fifth grades. The themes of the project developed were: 1st grade, Indians in Brazil in which the students investigated how the Indians produced dyes to paint their bodies; 2nd grade, Animals from the Pantanal Ecosystem, and the question investigated was why the knees of the of the Tuiuiú bird, symbol of the Pantanal region, bend backwards when he walks; 3rd grade, Astronomy, primarily the planets and the students' interest was to investigate the environmental conditions necessary to maintain an organism alive; 4th grade, Garbage, and the questions that arose were in relation to organic and inorganic garbage; how garbage decomposes over time and if the conditions in which it is kept influences its decomposition; and 5th grade, Plant Cultivation, and the theme studied was the growth of plants in different soil conditions, fertilized earth, normal earth, or in cotton, as shown in Figure 2.



Figure 2a. Testing dye fixation on skin



Figure 2b. The Tuiuiú bird



Figure 2c. Chrysanthemums plant studied



Figure 2d. Garbage decomposition



Figure 2e. Growth of beans

Figure 2. Students' projects

"The City that We Want" project in São Paulo

In the early 2000s, a team led by the Massachusetts Institute of Technology (USA) started a project with the Municipal Secretary of Education in the city of São Paulo, Brazil. The goal was to show what could be accomplished in a typical public school using technologies such as programming, robotics, and physical computing. The project encompassed as many as 30 schools throughout São Paulo (Cavallo et al., 2004). Students were to begin by identifying generative themes that would motivate their

projects. Due to rain shortage and lack of infrastructure investment, Brazil was experiencing a massive crisis in the electric energy system, and the population was resorting to all sorts of creative solutions to save energy. The crisis, being an everyday concern for all the population, appeared to be a good generative theme, which could generate projects such as building galvanometers, timer devices, water heaters, energy generators, and robots to control lights. The theme eventually changed based on the interactions with the students, and the change underscores the meaning of negotiating in real time and in locus for truly authentic Freirean generative themes. Since the majority of the households in the Brazilian favelas had illegal energy connections, and therefore neither energy meters nor bills, students were more interested in projects about safety and raising awareness about the danger of illegal connections, in order to train them to make safe, yet illegal, energy connections.

The second authentically Freirean moment in the project had to do with the use of materials. As students acquainted themselves with the new resources and planned their projects, they were initially enthralled by the Lego parts, digital cameras, and video cameras. But even though students seemed excited, some were afraid to use the equipment, and these anxieties were only further stoked by some teachers. There were historical reasons for such behavior. Access to computers in schools is often regarded as an administrative issue, addressed with strict usage rules and constant supervision. The high cost of the equipment and maintenance (especially in developing countries) amplifies the concern of damaging these machines. In many schools the computer room was even more regimented than regular classrooms. The solution was to break with all established rules. The researcher's computers were left on the floor unattended and available for students, cameras were made available without signup sheets. During some of the wrap-up interviews for the project, results revealed that 70% of students mentioned "trust" as the most important element of the workshops. Students explained that they felt trusted by researchers because they were allowed to freely use the equipment. For an attentive reader of Freire, this should not come as a surprise: the manifestations of oppression and power are not necessarily overt. Similarly, manifestations of trust are not always explicit. The unrestricted access to equipment was a design decision heavily inspired by Freirean ideas.

Finally, a third element in this project was very specific to Brazil (and likely other developing countries). As researchers started to get acquainted with the community and students' lived experiences, by visiting houses, small stores, and car repair garages, they identified a technological culture of repurposing and recycling. Car mechanics would use all sorts of improvised solutions to keep cars running at a minimum cost. In their homes, people would never discard a broken appliance without trying to fix it in all possible ways. If fixing was impossible, they would repurpose the broken device in creative ways. The community radio station was put together with equipment from different sources, many of which were broken or incompatible and had to be fixed. The perception of this culture led researchers to change, again, the design of the project, and replace Lego materials with the repurposing of electronics, appliances and other devices.

Conclusion

The goal of this paper was to show how, in the history of constructionist implementations in Brazil, some particular characteristics of the Brazilian context were determinant. We focused on a few of those characteristics. The first and most prominent is the influence of Paulo Freire and his ideas. In particular, there has been a considerable focus on generative themes, community-relevant projects, personal relevance, connection with broader societal problems, and sensitivity to local culture (including perception of cost of technology). Even though Papert and his colleagues were also very concerned with many of those issues, the work of Freire in Brazil preceded them, so the perception was that there was a combination of both approaches.

But there were order elements of the implementations that were not only theoretical considerations but consequences of Brazil being a developing country, with public schools being not well resourced and most of them being located in low-income areas. For example, computers were perceived as expensive, foreign and threatening devices, and therefore the reaction to projects that were technology-heavy was quite different from the reaction in the US or Europe. Relative to the salaries of teachers and the budgets of schools, computers were so much more expensive than in the developed world, so they were more likely to end up locked in a room with many rules and security measures. At the same time, because

public schools in Brazil have so many shortcomings, bringing technology into those schools was seen with much more suspicion, and seen as much less of a priority. All of those factors influenced the reception of constructionism in Brazil in ways that perhaps were very different than what took place in developed countries.

Seymour Papert purposely refused to define constructionism in precise ways, and advocated for researchers and practitioners to define it by themselves through rich, contextualized learning narratives. It seems that in the Brazilian case, that process took place with mixed results. On one hand, it was possible to combine constructionism with existing theoretical approaches such as the work of Paulo Freire. On the other hand, the difficulties stemming from the fact that technology was perceived much differently in Brazil were never fully resolved in many of the projects.

References

- Borges, E.L.; Borges, M.A.F.; Baranauskas, M.C.C. (1995). Da simulação à criação de modelos: Um contexto para a aprendizagem na empresa. *Proceedings of the VI SBIE - Simpósio Brasileiro de Informática na Educação*, Florianópolis, SC, Brasil, 1995.
- Calani, M.C. (1981). *Conceitos Geométricos Através da Linguagem Logo*. Master dissertation. Department of Computer Science, State University of Campinas (Unicamp), Campinas, Brazil.
- Fernandes, L. D.; Furquim, A. A.; Baranauskas, M. C. C. (1996). Jogos no computador e a formação de recursos humanos na indústria. *Proceedings of 3rd. Congresso Iberoamericano de Informática Educativa. R BIE Barranquilla – Colômbia*.
- Freire, P. (1975). *Pedagogy of the Oppressed*. New York: The Seabury Press.
- Freire, P. (1980). *Conscientização. Teoria e prática da libertação: uma introdução ao pensamento de Paulo Freire*. São Paulo: Moraes.
- Maia, I, F. (2011) *No jardim dos letramentos: tomadas de consciência e poéticas em rede e na cultura da convergência*. Ph.D. thesis, State University of Campinas (Unicamp), Campinas, Brazil. Retrieved from repositorio.unicamp.br/jspui/handle/REPOSIP/284430.
- Papert, S. (1986). *Constructionism: A new opportunity for elementary science education*. A proposal to the National Science Foundation, Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, Massachusetts.
- Schlünzen, K., Jr. Construindo conhecimento nas empresas usando software construcionista. In Lethelie , E., Bortolozzi, F., Weber, K. C. Pereira, H. (Eds.), *Anais do International Symposium on Knowledge Management/Document Management – ISKM/DM2000* (pp 197 - 215), Curitiba: Pontifícia Universidade Católica do Paraná, PUCPR.
- Valente, J.A. (2003) Teacher training via Internet: Creating a virtual environment for contextualized learning. In *Proceedings: 9th European Logo Conference, Eurologo 2003*, Porto, Portugal, August 27-30, 2003, p. 38-47.
- Valente, J.A. (2005). *A Espiral da Espiral de Aprendizagem: o processo de compreensão do papel das tecnologias de informação e comunicação na educação*. Thesis, State University of Campinas, (Unicamp), Campinas, Brazil. Retrieved from <http://www.bibliotecadigital.unicamp.br/document/?code=000857072&opt=4>.
- Valente, J. A., Baranauskas, M. C. C. & Martins, M. C. (2014). *ABInv – Aprendizagem baseada na investigação*. Campinas, SP: Unicamp/NIED. Retrieved from www.nied.unicamp.br/?q=livros.

Concept-building Oriented Programming Education

Jiří Vaníček, vanicek@pf.jcu.cz
University of South Bohemia, Czech Republic

Abstract

In this methodological paper we present an approach to teaching programming to 12 to 14-year-old pupils based on concept building. The method of presenting pupils with a set of similar tasks of increasing difficulty with the same concept at the background is conducted through construction of didactical environments. The world of Scratch is simplified in these environments in a way to enable pupils to focus on the given problem. A pupil is guided through various situations in which the concept is at play and builds a generic model of the concept. The paper presents concretized principles for creation of educational materials for teaching programming. The principles are fulfilled in teacher and pupil's materials which were developed within the project PRIM bringing computational thinking into compulsory primary and secondary education in Czechia. Some experience from a pilot research study of teaching conducted is presented.

Keywords

programming education; lower secondary; concept-building; Scratch

Introduction

If Informatics is, in accordance with Gander (Gander, 2014, p. 7), regarded as a part of general education, computational thinking becomes a substantial new skill. If it becomes a component of compulsory curriculum, programming, perceived as a “playground” where various components of computational thinking such as abstraction, algorithmization, decomposition, evaluation or generalization (Selby, 2014) and other cognitive functions can be developed, grows in importance. In consequence we have to ask what approaches and methods to use when teaching programming. These approaches and methods must respect the above stated goals as well as the decreasing age at which the teaching of Informatics starts. Many education authorities believe that IT can help children develop their competencies already in their early years (Kalaš, 2010, p. 9), which is in line with Bruner's proposition that the foundations of any subject may be taught to anybody at any age in some form (Bruner, 1960, p. 12).

The Czech Republic, like other countries, can expect programming to become compulsory already at primary school (see both the government programme Strategy for digital education until 2020 (MŠMT, 2014) and the new edition of the Framework Educational Programme for Informatics). Yet very few teachers of the compulsory subject Informatics have training comparable to teachers of other subjects. A research survey conducted in the Czech Republic shows that only 18 % of respondents in a research survey among ICT teachers at lower secondary schools show teaching qualification in informatics or some closely related discipline (Rambousek, 2013, p. 13). The same researches show that pupils would appreciate if they could work with computers much more and in a more interesting way (p. 159). Currently attention in lessons of informatics is paid to the user approach to technologies. Teachers are not ready to teach the foundations of informatics and do not realize they will need this skill. Most teachers who teach Informatics in Czech schools cannot code a program. This means that what is needed at this time is a curriculum and teaching materials that can be used by an insufficiently trained teacher with little experience in the area of programming and Informatics and with a lot of fears about not being able to cope with the new demands and content. The case is probably much the same around the world.

Rather than focusing separately on educating and training professional IT specialists, it may be more practical to focus on how teachers can learn by doing: how learning and teaching of programming can develop teachers' computational thinking as well as students'. Many approaches to teaching programming favour pupils' activity, active learning, learning by doing, and construction of knowledge

as a result of active creative work. All this with respect to the fact that knowledge and knowing are not transferable. We believe that knowledge is actively constructed by the learner in interaction with the world, so we are inclined (following Ackermann) to offer opportunities for children to engage in hands-on explorations that fuel the constructive process. This approach is in line with Piaget's view that "knowledge is an experience" and Papert's opinion that "knowledge is formed and transformed within specific contexts, shaped and expressed through different media" (Ackermann, 2001, p. 3, 8).

Some might expect that this approach within the Scratch environment can guarantee that a pupil will learn to programme in an innovative way. We doubt that this is sufficient. A very important factor in the education process is the quality of the teachers, their experience and prior training and education as well as their beliefs, whether they are able and willing to depart from a traditional teaching focus on having pupils' reproduce and imitate and giving them little opportunity for creative activity.

Orienting teaching on building of concepts

It is a challenge to concretize and implement this theoretical basis into a particular way of teaching programming. One of the possible ways is to use the framework (Explore, Explain, Envisage, Exchange, bridgE) used in the ScratchMaths project for primary programming education in England (Benton et al, 2016, p. 29). When looking for how to conceive teaching materials for programming in Scratch on lower secondary school level we turned our attention to theories from didactics of mathematics, which has been interested for several decades in how concepts are built. School mathematics has sought new ways of teaching since the 1960s. The experience from the so called New Math (Kline, 1973) made mathematics educators realize that "understanding mathematics is not given by the content but rather by the method of teaching" (Hejný, 2012, p. 43).

The concept is the key term in procept theory (Gray, Tall, 1994, p. 117), scheme-oriented education theory (Hejný, 2012) as well as APOS theory (Dubinsky, 2001). Block-oriented programming environments such as Scratch allow a pupil to see all the available commands of the language but it is difficult for pupils to see which of the commands represent important programming concepts and whether their mastery is essential for development of programming skill. It is up to teachers to lead their pupils through this battlefield; orientation to building concepts will help them.

Results of research analysing the frequency of the use of various commands in Scratch projects shared by users worldwide point at this issue. The category of More blocks, essential for development of the skill to decompose, was the least frequent among 10 categories of the language; only 1 % of blocks used fell into this category (Hudičák, 2017). This corresponds to other studies (Scratch, 2017). It seems pupils have to be led to some concepts because their lack doesn't prevent pupils from programming but can impede their development of computational thinking. This may occur even if teaching is oriented on creation of products; in some cases formal knowledge is required.

Theory of generic mental models (Hejný, 1987) works with the mechanism of cognitive process and helps to analyse pupils' thinking processes and detect sources of pupils' mistakes. The aim of the approach is to decrease the emphasis on formality of knowledge without deeper understanding. According to this theory the process of constructing a piece of knowledge starts from initial motivation, moves to the construction of isolated models and results in creation of the so called generic models. Having been motivated, a person first observes phenomena in which the new concept is present and creates isolated models (Hejný, 1987, p. 59) that are tested in new situations. With enough time and opportunity to get a sufficient number of isolated models in different situations, person is able to build a generic model, which is universal and should work in all known situations.

A concept is not a data point, a single piece of information. It is an abstraction. To grasp a concept a pupil needs as many isolated mental models, experience with specific cases in which the concept is at play to allow making connections in the pupil's mind, looking for relationships, structuring the information, constructing a generic model of the intended concept. This implies a pupil must go through a lot of number of situations in which the concept appears through various prisms and from different points of view. The pupil in these situations creates a number of models including what seem like models, but are partial and/or incomplete, as well as models that are surprising (Hejný, 2013a). The

pupil must come across situations in which the concept behaves in a strange, specific, unexpected way to see it plastically.

Let us illustrate the importance of understanding concepts on several examples. If we want pupils to understand programming, to build concepts correctly, it is not enough that they be able to develop a programme, create an algorithm, or propose a solution to the problem. They must be able to look at the problem from a distance; they need the practical programming skills that they will apply when solving the situation. So called “beaver tasks” from the Bebras challenge contest (Dagienė, 2017) are situational informatics tasks. In such situational tasks, pupils plunge into a described situation which they must grasp, get to understand the concepts and terms that are used, find an informatics principle the task is based on and solve the problem using cognitive and thinking skills. Tasks of this type are unsolvable without deep understanding of the concepts related to programming.

Hejný claims that understanding is more important than skill. A well-constructed mental model helps one grasp a concept correctly and find relationships between this concept and others. An example of diametrically different behaviour of various commands in Scratch is the pen state or sprite visibility preserving the given state and is valid until the state is changed, which is in contrast to executive commands “stamp” or “change costume”. Pupils have problems with understanding these differences. Other example comes from programming robots. It may happen that the “programme is standing and the robot is moving” e.g. having carried out the command that makes the engine work the programme waits for fulfilling the condition at the input from sensors before carrying out the command to stop the engine. Children suppose that when the programme is stopping, the robot must not move. These situations are difficult for pupils if they do not understand the mechanism of the given concept well, do not have the generic model built.

Two types of programming tasks and building of concepts

Observations of in-service teachers (well-experienced as teachers but without training in informatics) as well as pre-service teachers (with informatics background but without experience) are a rich source of information about how these teachers approach their teaching. We compared approaches to teaching programming used in Czech and other textbooks and also methodological manuals for teaching programming available on the internet. Predominant in these textbooks and manuals are two basic types of programming tasks.

One uses a series of several-minute long programming etudes (as in Kalaš, 2017), always targeting the acquisition of one specific skill or focused on one specific item of knowledge or one concept. Pupils can easily check their result and it is easy to organize such lessons especially for non-well-experienced teachers. But motivation in these activities is weaker than in larger projects.

Teaching through larger programming units – the so called “projects” (as in LEAD, 2013), e.g. creating games or stories — often has the form of a sequence of activities organized as a tutorial or a problem task. It is hard to combine the goal to teach a specific concept, procedure or method with open-ended activities. These activities cannot target the development of one concept. Programming projects can result in creation of long multiline codes in which it is harder for the pupil to orient and for the teacher to find the pupil's mistake.

In the two ways of selecting types of problems, two different approaches to teaching goals can be observed: in case of shorter programming etudes the approach is intensive, focused on competences in the area of concepts, and the other is holistic, more open, emphasising creativity, focusing on product (Vaniček, 2015).

Didactical environments

One of the disadvantages of block oriented programming environments is that users can see all the blocks of the language. However, pupils cannot master all these commands, especially if they are to understand them in depth. Then it is difficult to give pupils a problem because if they don't immediately know how to solve the task they may, instead of thinking, waste time by looking for an appropriate tool or command that would solve the problem. A tutorial is not a solution because following fully specified

procedures does not help a student learn to generalize. Especially in introductory activities for teaching programming, we find it most efficient to create didactical environments. A didactical mathematics environment as understood by Wittmann (2001) is a set of linked situations that provide problems that let a pupil discover important ideas. We add to this concept three more conditions: the motivating power, long-term commitment, and flexibility in the level of difficulty. (Hejný et al, 2013b).

Mathematics educators use ways of creating learning environments based on one task. Wollring mentions the relation “Task \subset Task format \subset Learning environment” (Wollring, 2008). In his scheme-oriented approach to mathematics education Hejný (2011, 2013b) uses learning environments that are close to children’s everyday experience (e.g. Spider web, Snakes, Staircases, Stepping, Seat in the bus) to create learning environments, microworlds in which it is possible to grasp the rules quickly and to solve a given set of similar or progressive tasks. Similar environments have been used in programming education for years (e.g., turtle graphics, robot Karel). Scratch allows one to create learning environments through sprites, their costumes and scripts, by setting the backdrop of a stage, and by preparing blocks and scripts on the script building area. In fact it enables one to create environments didactically as strong as Hejný’s which depart from the world of mathematics and are viable as Scratch projects that a pupil opens and works in.

Paraphrasing Hejný’s characteristics of effective teaching mathematics (Hejný, 2012), we characterise effective teaching of programming by three cognitive goals:

- pupils understand programming, their knowledge is not mechanical;
- pupils are intrinsically motivated for work, they are not frustrated by programming;
- pupils develop intellectually, which primarily means development of the ability to: 1. communicate both orally and in writing, 2. cooperate in a group or even lead a group to solve problems, 3. analyse a problem situation, 4. effectively solve problems and 5. correct one’s own mistakes. (Hejný, 2012).

Properties of curriculum based on building concepts

The theoretical background described above allows us to state principles for creating a concept building programming education curriculum.

- It is made of a lot of very short activities, which are related to the same concept. The aim is for pupils to get an idea of the behaviour of the given concept in various situations. The activities help the teacher; unlike broader activities in which a teacher can get lost, in these short activities, the teacher is able to find a pupil’s mistake in a script, and there is less chance that the pupil will programme a situation which the teacher is not able to test, fails to correct or cannot guide the pupil to discovery.
- Several didactical environments must be used for teaching one concept. These environments must be based on different activities of sprites, e.g. drawing lines, stamping shapes, moving on the scene, changing costumes. In these environments the intended concept is shown in various thought-out situations, allowing the pupil to abstract away the incidental or irrelevant properties and attach to the concept properties that are substantial and independent of a particular environment. The pieces of knowledge gained by the pupil do not remain isolated.
- A mistake is not unwelcome; on the contrary it is one of the main sources of knowledge. A pupil must be presented with various possible mistakes that can be the result of misunderstanding and that can help the pupil grasp the concept more deeply. Pupils find it motivating assignments of such activities are worded as if somebody has made a mistake in their programme and is facing a situation in which they desperately need help. The pupil’s task is to figure out what situations expose the mistake, what it is caused by, if it really is a mistake (some solutions may be erroneous only from certain points of view) and whether the mistake is covert (e.g. the solution is not general enough or the mistakes shows only after change of input parameters or when run again).
- Activities should aim at various programming competences. Pupils are asked not only to create and run a script but also to read it, interpret it, predict the output, test its correctness, set the situation in a way that would make the mistake manifest or to adapt a working script to solve a similar problem. Then the pupil’s pieces of knowledge do not remain isolated.

chapter about assembling blocks poses the question “Which script did the cat use to draw letter U?” to teach pupils to read scenarios.

Rocket track

1. The rocket moves towards the mouse pointer and when [T] is pressed, it stamps a picture of itself. Create a script.

2. Tom created this script:

When launched, the rocket never stamped itself. Can you find why?

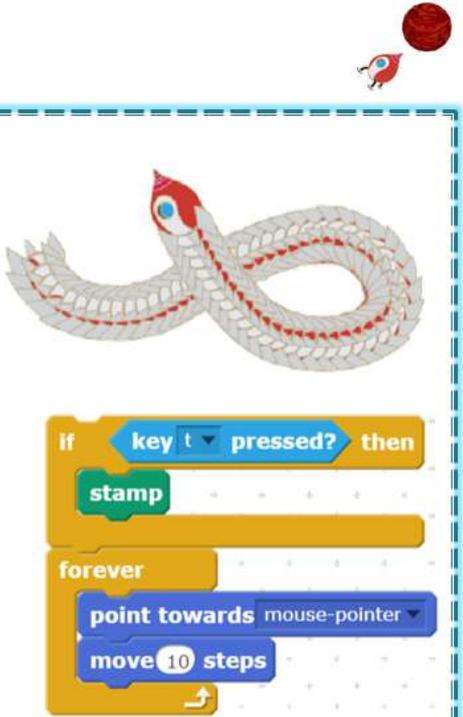


Figure 2.

Figure 2 shows a task in which students are to find a mistake in the script. Students know that, **pen down** sets the sprite’s state to stay down through subsequent moves. For students who imagine that **stamp** acts in the same way, setting a sprite-state that makes it continue stamping through subsequent moves, the sprite behaves contrary to expectations. Pupils are meant to reason and test hypotheses. If more tasks of this type with different kinds of mistakes with different sources are used, pupils are guided to deeper understanding of the given concept.

Figure 3 shows the didactical environment Words and Letters. Letters are “printed” by stamping a sprite of the appropriate costume. This is an advanced task where pupil creates a procedure with a text parameter.

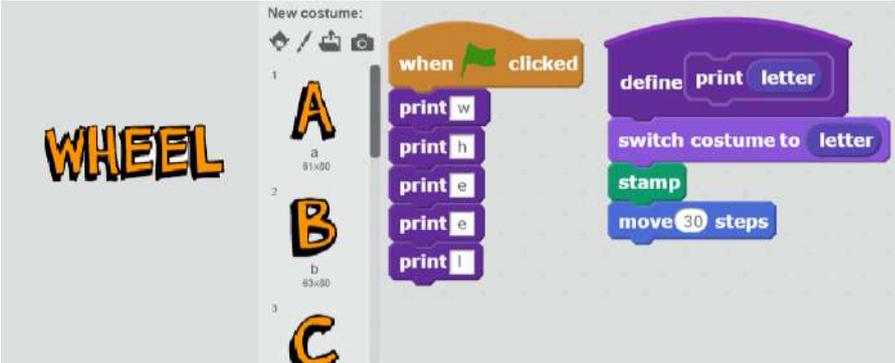


Figure 3.

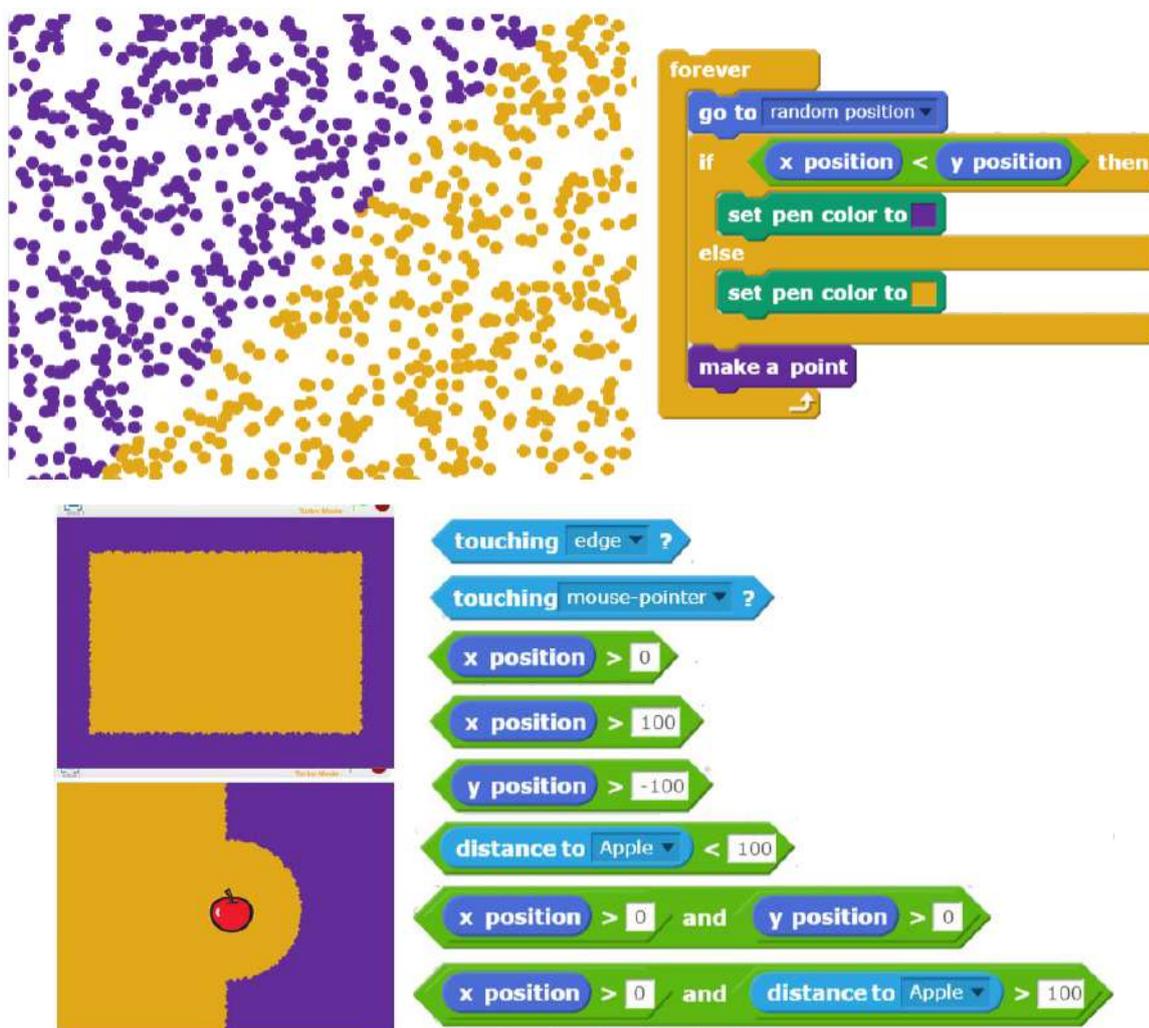


Figure 4.

Figure 4 shows the didactical environment Coordinates for understanding conditions, conditionals and coordinates. A randomly jumping sprite makes points whose colour depends on whether the given condition is fulfilled. In the bottom part, conditions of growing complexity are shown. By experimenting with various conditions a pupil may get to understand coordinates.

Conclusion

The first version of the textbook is finished and is being piloted at schools. Preliminary research was conducted with experienced teachers. In later stages, the textbook will be tried out with teachers who lack education in informatics and had no prior experience teaching programming. So far the piloting has shown that

- even experienced teachers find it difficult to adapt teaching in which they do not talk too much to pupils and let them work independently;
- teachers skip those tasks they find too similar to previous tasks and fail to see a different teaching goal in them, to see a different situation that offers the pupils another isolated model;
- teachers tend to evaluate each pupil's answer, to state whether it is right or wrong instead of allowing the computer to provide the feedback, allowing the computer to show whether the pupil's idea was correct or not;
- if a pupil asks for help when they make a mistake, teachers immediately show the mistake instead of giving encouragement or advice on how to look for the mistake;

- but teachers find pupils' attitudes to programming positive, which is motivating for them as teachers.

We believe that teaching programming can contribute significantly to development of each pupil. To achieve this goal, it is important that students not simply adopt ready-made knowledge but learn to argue, discuss and evaluate. Critical thinking as a way to work out what is disinformation is an important skill of a citizen of a democratic society. Especially in our geographical area we believe the school should put emphasis in education on pupils' ability not to allow anybody to manipulate them. This will allow us all to fulfil the thesis of the famous Czech educator Comenius, who believed education would help to solve problems of the whole world.

Acknowledgement

We would like to thank the project "PRIM" – "Support for Development of Computational Thinking" (CZ.02.3.68/0.0/0.0/16_036/0005322) for funding this work.

References

- Ackermann E. (2010) Constructivism(s): Shared roots, crossed paths, multiple legacies. In Clayson, J. E., Kalaš I. (eds.) *Constructionist approaches to creative learning, thinking and education: lessons for the 21st century, proceedings Constructionism 2010*, Paris 16.-20. 8. 2010. Bratislava: Library and publishing centre Comenius University.
- Ackermann, E. (2001) Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication*, Vol. 5 No. 3, p. 438.
- Benton, L., Hoyles, C., Noss, R., & Kalas, I. (2016) Building mathematical knowledge with programming: insights from the ScratchMaths project. In: *Proceedings of Constructionism 2016*, Thailand: Bangkok, February, pp. 25-32.
- Bruner, J. S. (1960) *The process of education*. Cambridge: Harvard University Press.
- Dagienė, V., Sentance, S., V. and Stupurienė, G. (2017) Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, Vol. 28, No 1, p. 23-44.
- Dubinski, E., & McDonald, M. (2001) APOS: A constructivist theory of learning in undergraduate mathematics education research. In D. Holton (Ed.): *The teaching and learning of mathematics at university level: An ICMI Study*, pp. 275–282. Dodrecht: Kluwer Academic Publisher.
- Gander, W. (2014) Informatics and General Education. In Gülbahar, Y., Erinc, K. (Eds.) *Informatics in Schools, Teaching and Learning Perspectives*. Heidelberg: Springer LNCS, p. 1-7.
- Gray, E., Tall, D. (1994) Duality, ambiguity and flexibility: A proceptual view of simple arithmetic. *Journal for Research in Mathematics Education*, Vol. 25 No. 2, pp. 116–141.
- Hejný, M. (1987) *Teória vyučovania matematiky 2* (Theory of mathematics education 2). Bratislava: Slovenské pedagogické nakladateľstvo, 553 p.
- Hejný, M. (2011) Scheme-oriented mathematics education: Spider web mathematical environment. In M. Kaldrimidou, X. Vamvakousi (Eds.), *Proceedings of 4th Conference of Greek Association of Researchers of Mathematics Education "The classroom as field of development of mathematical activity"*, Ioannina: University of Ioannina & GARME, pp. 3–24.
- Hejný, M. (2012) Exploring the Cognitive Dimension of Teaching Mathematics through Scheme-oriented Approach to Education. *Orbis scholae*, Vol. 6 No. 2, pp. 41-55, ISSN 1802-4637.
- Hejný, M. (2013) *Vyučování matematice na 1. stupni ZŠ orientované na budování schémat: Aritmetika*. (Scheme-oriented mathematics primary education: Arithmetics). Praha: PedF UK.

Hejný, M., Slezáková, J., Jirotková, D. (2013) Understanding equations in schema-oriented education. *Procedia - Social and Behavioral Sciences*, Vol. 93, October, pp. 995-999.

Hudičák, M. (2017) *Co vlastně děti programují ve Scratch? (What do kids actually program in Scratch?)* Theses. České Budějovice: Jihočeská univerzita.

Kalaš, I. (2010) *Recognizing the potential of ICT in early childhood education*. UNESCO IITE, Moscow, 148 p.

Kalaš, I. (2017) *UCL Scratchmaths curriculum*. UCL, Institute of Education. Available at <http://www.ucl.ac.uk/ioe/research/projects/scratchmaths/curriculum-materials>

Kline, M. (1973) *Why Johnny Can't Add: The Failure of the New Mathematics*. New York: St. Martin's Press.

MŠMT (2015) *Strategie digitálního vzdělávání (Strategy of digital education)*. Praha: MŠMT, 2014, 49 p.

Rambousek, V. et al. (2013) *Rozvoj informačně technologických kompetencí na základních školách (Development of information technology competencies at primary and lower secondary schools)*. Praha: Česká technika.

Scratch statistics (2017) Scratch Block Usage (random sample). Scratch [online]. [cit. 2018-01-04]. Available from: <https://scratch.mit.edu/statistics/>, <https://scratch.mit.edu/projects/99177947>

Selby, C., Woollard, J. (2014) *Computational Thinking: The Developing Definition*. SIGCSE, March, Georgia: Atlanta.

The LEAD Project (2016) *Programování pro děti (Programming for children)*. Brno: Computer Press, 160 p. EAN 9788025138090.

Vaníček, J. (2015) Programming in Scratch using inquiry-based approach. In Brodник, A. (ed.) *Informatics in schools. Curricula, competencies, and Competitions*. Heidelberg: Springer Lecture Notes in Computer Science, Vol. 9378, pp. 82 - 93.

Wittmann, E. Ch. (2001) Developing mathematics education in a systemic process. *Educational Studies in Mathematics*, Vol. 48 1 20.

Wollring, B. (2008) Kennzeichnung von Lernumgebungen für den Mathematikunterricht in der Grundschule. In Kasseler Forschergruppe (ed.): *Lernumgebungen auf dem Prüfstand. Bericht 2 der Kasseler Forschergruppe Empirische Bildungsforschung Lehren – Lernen – Literacy*. Kassel: Kassel university press GmbH, pp. 9-26.

Practice papers

Constructive and Collaborative Digital Storytelling for Enhancing Creativity and Cooperation In and Out of School

Francesca Agatolio, *francesca.agatolio@phd.unipd.it*
University of Padua, Dpt. of Psychology, Italy

Alfredo Asiain, *alfredo.asiain@unavarra.es*
Public University of Navarra, Dpt. of Philology & Language didactics, Spain

Alfredo Pina, *pina@unavarra.es*
Public University of Navarra, Dpt. of Math & Computer Engineering, Spain

Gabriel Rubio, *gabrielmaria.rubio@unavarra.es*
Public University of Navarra, Dpt. of Philology & Language didactics, Spain

Michele Moro, *michele.moro@dei.unipd.it*
University of Padua, Dpt. of Information Engineering, Italy

Abstract

An effective integration of technologies in the didactical context means much more than providing students with an easy access to computers. Technological tools can be used to provide students a personalised and active learning environment in accordance with the constructionist principles but, for this purpose, teachers should be aware of which type of knowledge and skill they aim to transmit and which technologies can help with that. In this paper we describe two digital storytelling experiences, held respectively in Spain and in Italy using Scratch, which provide an example of how teachers can offer an engaging learning experience without much effort. In both cases, Scratch is seen as a catalyst for different knowledge, skills and disciplines.

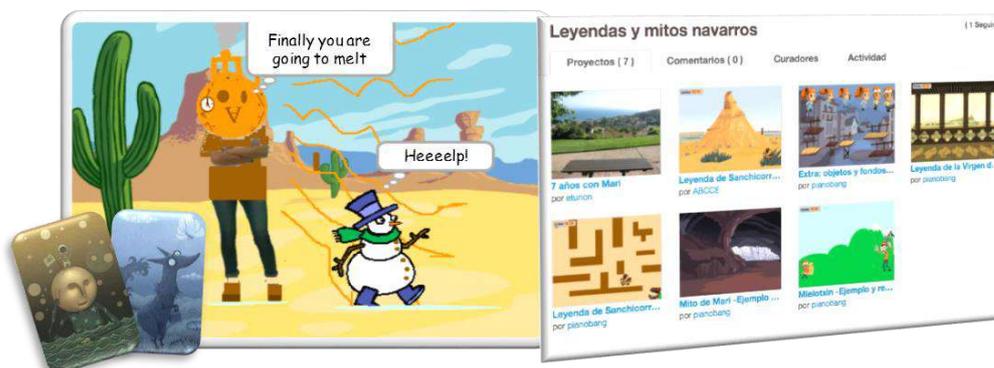


Figure. Two digital storytelling experiences using Scratch

The activity made in Spain, in particular, focused on the multidisciplinary aspect by the creation of stories related to the intangible cultural heritage. The Italian experience, on the other hand, was aimed at supporting creative and collaborative learning for gifted children.

Keywords

constructionism; digital storytelling; personalised learning; multidisciplinary learning; gifted children

Introduction

In the interesting paper “Digital Storytelling: A Powerful Technology Tool for the 21st Century Classroom” (Robin, 2008), Robin started his reflection about the potential of digital storytelling reporting the results emerged in 2007 by the survey of the U.S. Department of Education. The report states that no significant differences are detected in the performance of students that use technologies at school compared to others. In 2015, the results of the PISA survey (OECD, 2015), not only confirm this fact but also suggest that the students’ performance seems to be inversely related to the quantity of time spent using computer in school. Indeed, students from countries where schools provide easier access to computers, show worse performance in ‘digital reading’ and mathematics (fig. 1). The report concludes that “On average, in the past 10 years there has been no appreciable improvement in student achievement in reading, mathematics or science in the countries that have invested heavily in information and communication technologies for education”. This should not come as a surprise. The report states that the main activities mentioned by the students are: chatting on line; sending e-mails; browsing the Internet; downloading, uploading or browsing material from the school’s website; posting work on the school’s website; playing simulations at school; practicing and repeating lessons; doing individual homework on a school computer; using school computers for group works and to communicate with other students. If we consider the usage of computers during mathematics classes, things are not much better. More frequently tasks are: drawing the graph of a function; calculating with numbers; constructing geometric figures; entering data in a spreadsheet; rewriting algebraic expressions and solving equations; drawing histograms; finding out how the graph of a function changes. All this has little to do with the computer’s role described by Papert; all this is still the “computer that programming the children” (Papert, 1980).

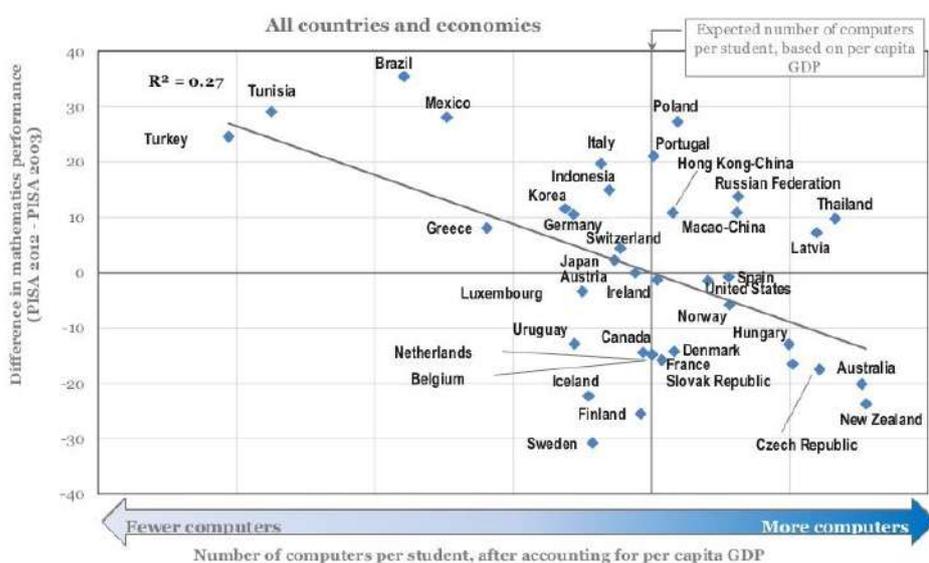


Figure 1. Trends in mathematics performance and increase in computers in schools (OECD source)

We are among those educators and researchers that think that “Integrating technology is much, much more than putting a piece of software into a classroom” as Robin states citing Mary Ann Wolf, executive director of the State Educational Technology Directors Association. To invest in technologies for education does not only mean ensuring that schools must have computers and that students have to use them a certain number of hours per week; and neither does only mean to provide internet connection. This is only a step - maybe not the most urgent – in the process of integrating technologies in education. Technologies are not a panacea for all the problems of education, but some technologies lend themselves to help teachers in “engaging students in experiences liable to encourage their own personal construction of something in some sense like it” (Papert et al., 1991). Integrating technologies in education means 1) to identify the added value provided by using a specific technological tool to convey a certain knowledge and/or skill (Mishra et al., 2006), and 2) to use this tool to deliver a personalized “discovery experience” to the students. It is a “magic” that requires many interconnected different kinds of knowledge, among them the technological knowledge is not the main one. However,

teachers should not get discouraged because it is a gradual path and there is no need for them to have all the answers. In this paper, we present two experiences of digital storytelling held in Spain and in Italy: both of them are easily replicable, do not require complex technological knowledge and provide an example of how technologies can be readily used to prepare lessons where different disciplines and skills converge and students are actively engaged. Though the two experiences were organized separately, the past collaborations of the authors inspired by common pedagogical principles render these experience comparable and deserving a unique presentation and discussion.

Digital storytelling as a catalyst for different knowledge, skills and disciplines

Digital storytelling concerns the storytelling process supported by the use of digital tools such as Scratch or Alice. The digital component allows integrating text with animations, sounds, narrative voices and also interactive elements. The possibility to combine this variety of resources contributes to making storytelling a problem solving experience, particularly engaging for the students (Sadik, 2008) and able to bring out their creativity (Malita et al., 2010). During a digital storytelling activity, students are involved in many different processes such as the research on a topic, the organization of information, the construction of meaningful narratives and often team working (Robin, 2008). The use of digital storytelling goes back to the 1980s with the creation of the Center for Digital Storytelling in California and now it is worldwide diffuse, in part due to the spreading of low-cost technologies (Meadows, 2008). In (Yuksel et al., 2011) the authors report the results of an investigation about the use of this tool around the world. It emerges that the main reasons for teachers to choose digital storytelling are: to allow students to construct their own understanding in a content area, to facilitate team working, to promote discussion, to introduce students to new contents, to help them in facing problem solving issues, critical thinking and complex ideas. Moreover, since the software involved in this context are in most cases programming environments, digital storytelling is often used to introduce the basis of programming and computational thinking (Kelleher et al., 2007). On the contrary, the use of digital storytelling focused on the improvement of writing skills is less frequent (Burke et al., 2010). The two digital storytelling experiences that we are going to describe have been carried out separately in Spain and in Italy: activities in the Spanish case as regular classroom activities while in the Italian case they were developed as extra-curricular. The main aspect they have in common is the use of the technological tool (Scratch) seen as a catalyst for diverse skills, knowledge and disciplines. During the activities, children are not asked to solve a little isolated task but they are actively involved in the construction of a complex product (a story), acting like in a laboratory experience, applying contents from many subjects, sharing ideas and competences. Moreover, they have time to shape and refine their product. In defining the principles of constructionism, Papert was inspired by students who were working on soap sculptures (Papert et al., 1991). He reported that what had struck his imagination was the fact that “the project was not done and dropped but continued for many weeks.” During these weeks, students have “time to think, to dream, to gaze, to get a new idea and try it and drop it or persist, time to talk, to see other people's work and their reaction to yours”. We think “time” is one of the key point that teachers should take into account: if we want that our students become really involved in the learning experience, we should allowed them the time to embrace their project. Another aspect that should be considered is the difference of each student from the other and they should be allowed to express themselves and their own talent. Most technological tools (Peirce et al., 2008) are particularly suited to personalize the learning process: the two experiences we report in the paper, for instance, show how Scratch can be used both with standard groups of children and students with special needs. In all this, the technological tool never steals the ‘center of stage’ but it is only an instrument at the service of the students, just as a ruler or a protractor. In every class of each school there is at least a ruler, and teachers know when it can be useful for students and to what end; we'd like to be the same with a computer.

Digital storytelling to enhance creativity and to bring secondary students closer to our intangible cultural heritage: a Spanish experience

The intangible Cultural Inheritance of Navarra has several graphical, textual or sound elements as a multimedia database for the research and knowledge of the cultural richness of Navarre and Basse Navarre. However, this database is not known neither used by some of his target users: secondary and primary level students. Nevertheless, its contents are really suitable for exploring, handling and even gaming.

The choice of the task

The article 2 of the Convention for the safeguarding of the intangible cultural heritage (UNESCO 2003) says:

The “intangible cultural heritage” means the practices, representations, expressions, knowledge, skills – as well as the instruments, objects, artefacts and cultural spaces associated therewith – that communities, groups and, in some cases, individuals recognize as part of their cultural heritage. This intangible cultural heritage, transmitted from generation to generation, is constantly recreated by communities and groups in response to their environment, their interaction with nature and their history, and provides them with a sense of identity and continuity, thus promoting respect for cultural diversity and human creativity.

Resnick (Resnick 2007) says:

“To succeed in today’s Creative Society, students must learn to think creatively, plan systematically, analyze critically, work collaboratively, communicate clearly, design iteratively, and learn continuously. Unfortunately, most uses of technologies in schools today do not support these 21st-century learning skills. In many cases, new technologies are simply reinforcing old ways of teaching and learning”

Our hypothesis is that if we work with the contents of the intangible heritage in the right way using digital objects and an adapted narrative for young people, this could provide an attractive and creative engine to access and recreate intangible heritage. For thus we define a didactic strategy and appropriate tools suitable to our scholar context and able to deal with digital objects and to create digital stories.

Description of the activity

We used 4 narrative engines based on 4 Navarre legends (Mito de Mari, Mielotxin, Sanchicorrota and the Virgen of Uxue). The text of all of them has been adapted with a didactic simplification and the intangible heritage produced some digital images. The teacher provided to the students with 4 Scratch projects as possible examples of how to tell digital stories based on the 4 legends. Their task was to tell their own story of one of the legends, working in 3-4 persons teams. This has been done with 46 students (25 boys and 21 girls) of 2 classes of the first year of the secondary school (13 years old). The Stages/Sessions were the following and combined regular lectures of language literacy with computing sessions of technology/digital competency:

1. Pre Questionnaire. Reading and Analysis of the narratives
2. Examples that show how to transform a story into s videogame/digital story
3. Choosing the story and storyboards design
4. Storyboards presentations
6. Programming the game/interactive digital story (2 sessions)
7. Results presentation and post questionnaire

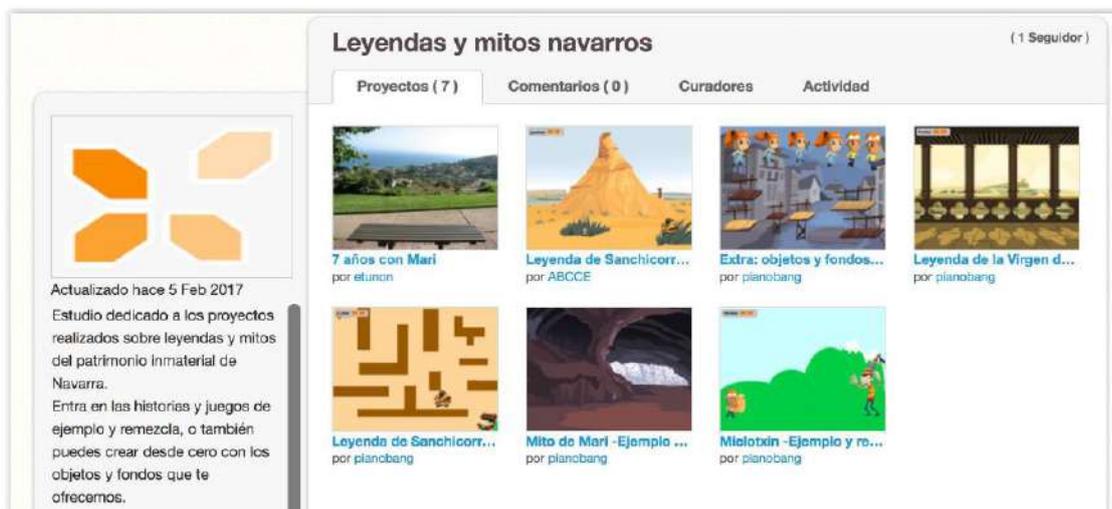


Figure 2. Digital objects for the activity (<https://scratch.mit.edu/studios/3737471/projects/>)

Results and considerations

We used two questionnaires in order to check the knowledge on programming and on the Navarre legends before and after the activity.

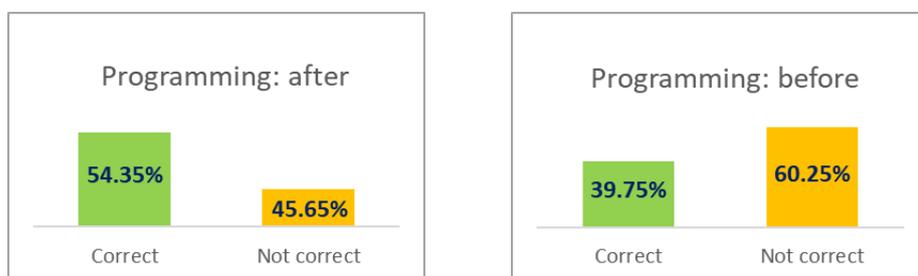


Figure 3. Programming skills knowledge before/after the activity.



Figure 4. Legends and Myths knowledge before/after the activity.

Figures 3 & 4 show the improvement after the activity; other feedback from the students is that they liked creating videogames, the work team was fine and all of them enjoyed the activity. Mixing programming and language literacy as we did, gives lots of possibilities both for technology and for literature.

Digital storytelling to enhance cooperation between gifted children: an Italian experience

From 2016 to 2017, we taught a programming course dedicated to 7-10-years-old gifted children. The course was based on Scratch; it lasted nine months and took place outside of school. The lessons were both face-to-face (three hours, once per month on Saturday) and online (one hour per week). In the last part of the course, we decided to introduce a digital storytelling activity. At that point, the children's programming skills had reached a good level and we sought to focus their attention on the cooperative aspect through a digital storytelling experience.

The choice of the task

When we started to design the mentioned activity, we had to take into account some limitations related to the course's organization. Indeed, the course aimed at offering mathematically talented students the possibility to explore challenging contents. Because children came from different parts of the region, many classes were held online, through an e-learning platform. Of course, this fact restricts the occasion to have team working experiences but it does not prevent the possibility to make a reflection on the importance of collaboration in some situations. Gifted children are often loath to work cooperatively with peers (Robinson, 2007). They appear frustrated when arguing their ideas and giving explanations, ending up as lonely dominators or silent onlookers (Diezmann et al., 2001). Moreover, we noticed they are not prepared to listen to each other and to modify their own ideas in relation to those of peers. This prompted us to propose a digital storytelling activity in which students, after defining the plot of the story as a group, make separately their scene while taking care that it was coherent with the ones made by the workmates. What we expected was that, after an initial "failure", they would understand the necessity to improve communication among their group to have a better result. Regarding the choice of the topic, we sought students to create their stories starting from the images of "Dixit" (fig.5), the famous board game created by the paediatric psychiatrist Jean-Louis Roubira. The peculiarity of the "Dixit" images is that they are metaphorical and open to different and creative interpretations. The frequent use of metaphors and their comprehension is a typical trait of talented students to the point that it can contribute to the identification of giftedness (Tan et al., 2013).



Figure 5. Some images from the "Dixit" board game

During our teaching experience with gifted children we often confirm this fact; we like to report here the example of a 10-years-old pupil that, about teamwork, said us he has learnt that teamwork is like drinking a hot drink: membership has to happen gently, otherwise you afford to burn your tongue. These considerations lead us to structure the activity as follows.

Description of the activity

In the face-to-face lesson, we introduced the activity giving children the "Dixit" cards and proposed them to play a simple game: we gave them four cards each and asked them to create a simple story using the images. Taking turns, they had to invent the beginning of the tale taking inspiration from a card; then, the next student had to continue the tale using one of his cards, and so on. In this way they explored the metaphorical potential of the images. After that, we divided the 12 students in 4 groups of 3 members each and we gave every group six cards. We asked students to select three cards, one for

each member of each group and to draft a storyboard drawing inspiration from them. During the next week, they made their sketches using Scratch and uploaded them in the e-learning platform; workmates could communicate through mail. In the first online lesson, we commented together with the students their works, reflecting on how could be improved. The next week children worked to fix and refine the sketches and, finally, they joined them to create the final story. In this way, we aimed at giving them the time both to express their creativity individually and to refine their work taking into account the ideas of the others.

Results and considerations

The groups of children approached the activity in different ways: only the members of one group felt spontaneously the need to coordinate from the beginning using email. The other children made the first version of their scenes without facing with the workmates, but almost all recognized the necessity to take in account the others' work when we asked them to join the sequences of the story. In general, they have proved to be reluctant to communicate with the others, probably also because using email is not suitable for children of this age. Despite this, they were willing to modify their own project adjusting the timing, changing the images and adding explanation to accommodate the workmates. Regarding this, we think that using a technological tool like Scratch makes the difference: indeed, it allows making changes in a quick way so that children can easily refine their work. Moreover, Scratch enhance the students' creativity providing different ways to express themselves: many students integrated the program with their personal drawings and voices inspiring the other children. The final products resulted quite different but they have in common the sense of humor and some surreal traits (fig. 6), suggesting the importance for a teacher to provide tasks open as much as possible so that also students with special needs can express themselves. Finally, even if the online mode proved to be a barrier to communication, it enables students to take the time they need following their inspiration.

Figure 6. Frames of a story. In quotes the children's voiceover



Conclusion and Discussion

An effective integration of technologies in the didactical context means much more than providing students with an easy access to computers: it requires an informed choice of the tools by which it is possible to create an active learning environment. Tools like Scratch make it easier to offer an experience in which different skills, contents and disciplines converge. In the paper we reported two examples of how, without much effort, teachers can use the digital storytelling as a multidisciplinary engaging tool (the Spanish experience) or as an instrument to promote personalised and collaborative learning without limiting creativity (the Italian experience). More specifically, the two experiences show that either disciplinary-oriented or motivationally-oriented projects are different faces of the same medal and that they can coexist when carefully adapted to the context, to the group of kids involved, to the expected outcomes. On the other hand, it must be observed again that the Spanish experience was designed and evaluated in a formal setting, whereas the Italian project was necessarily adapted to an informal setting, particularly

challenging for the exploitation of communication means. In taking inspiration from the two experiences, these characteristics should be taken into account.

References

- Burke, Q., & Kafai, Y. B. (2010, June). Programming & storytelling: opportunities for learning about coding & composition. In Proceedings of the 9th international conference on interaction design and children (pp. 348-351). ACM.
- Diezmann, C. M., & Watters, J. J. (2001). The collaboration of mathematically gifted students on challenging tasks. *Journal for the Education of the Gifted*, 25(1), 7-31.
- Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, 50(7), 58-64.
- Malita, L., & Martin, C. (2010). Digital storytelling as web passport to success in the 21st century. *Procedia-Social and Behavioral Sciences*, 2(2), 3060-3064.
- Meadows, D. (2003). Digital storytelling: Research-based practice in new media. *Visual Communication*, 2(2), 189-193.
- Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record*, 108(6), 1017.
- OECD (2015), *Students, Computers and Learning: Making the Connection*. OECD Publishing, Paris. <http://dx.doi.org/10.1787/9789264239555-en>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Peirce, N., Conlan, O., & Wade, V. (2008, November). Adaptive educational games: Providing non-invasive personalised learning experiences. In *Digital Games and Intelligent Toys Based Education, 2008 Second IEEE International Conference on* (pp. 28-35). IEEE.
- Resnick, M. (2007): «Sowing the Seeds for a More Creative Society», *Learning and leading with technology*, 35(4), pp. 18-22
- Robin, B. R. (2008). Digital storytelling: A powerful technology tool for the 21st century classroom. *Theory into practice*, 47(3), 220-228.
- Robinson, A. (1997). Cooperative learning for talented students: Emergent issues and implications. *Handbook of gifted education*, 243-252.
- Sadik, A. (2008). Digital storytelling: A meaningful technology-integrated approach for engaged student learning. *Educational technology research and development*, 56(4), 487-506.
- Tan, M., Barbot, B., Mourgues, C., & Grigorenko, E. L. (2013). Measuring metaphors: Concreteness and similarity in metaphor comprehension and gifted identification. *Educational & Child Psychology*, 30(2), 89-100.
- UNESCO 2003. *Convention for the safeguarding of the intangible cultural heritage* http://portal.unesco.org/en/ev.php-URL_ID=17716&URL_DO=DO_TOPIC&URL_SECTION=201.html
- Yuksel, P., Robin, B., & McNeil, S. (2011, March). Educational uses of digital storytelling all around the world. In *Society for Information Technology & Teacher Education International Conference* (pp. 1264-1271). Association for the Advancement of Computing in Education (AACE).

Towards a Framework for Educational Robotics

Julian M. Angel-Fernandez, *Angel-Fernandez@acin.tuwien.ac.at*

Vienna University of Technology, Vienna, Austria

Nikoleta Yiannoutsou, *nyiannoutsou@ppp.uoa.gr*

University of Athens, Athens, Greece

Chronis Kynigos, *kynigos@ppp.uoa.gr*

University of Athens, Athens, Greece

Carina Girvan, *girvanc@cardiff.ac.uk*

Cardiff University, Cardiff, Wales, UK

Markus Vincze, *Vincze@acin.tuwien.ac.at*

Vienna University Technology, Vienna, Austria

Abstract

Educational robotics has been considered as a field with a good potential to teach difficult concepts (e.g. friction) in appealing way. As a consequence, the interest in educational robotics has grown in the last decade, which is reflected in increasing number of robotic platforms, kits, and programming interfaces now available. Nevertheless, researches still fail on describe activities that could be used by teachers and other people with no technological fluency, who are scared by the overwhelm amount of information that made them avoid the use of robotics to teach. Moreover, most of the activities developed until now do not consider pedagogical methodologies to inform the design and implementation of them. As a direct consequence of the misinformation about the correct use of pedagogical methodologies and robotics' multidisciplinary, the number of people who master the use of robotics in education is still scant. This paper presents ongoing work on the development of a framework in the European project Educational Robotics for Science, Technology, Engineer, and Mathematics (ER4STEM). The framework aims to make evident the connection between 21st century skills, robotics and pedagogical methodologies to support the creation of pedagogical activities, which is defined in ER4STEM as an activity that has clear learning outcomes and evidence of learning, use of one or more pedagogic methodologies during the activity, and detail description of the activity. This is achieve through the critical use of tools and examples of activities developed ER4STEM.

Keywords

educational robotics; framework for educational robotics; pedagogical activities; educational activities; educational robotics for STEM; constructionism

Introduction

Robotics is a field where mechanical, electronic and computer engineering converge but it also involves other fields such as mathematics, psychology (e.g. human behaviour and attitudes), biology, arts, and sciences. Therefore, it has been recognized as a technology that could highly impact education (Papert, 1980). Nevertheless, the broad connection with different fields and the constant evolution of technology can make people to focus on the technology aspect without fully consider how pedagogical methodologies should be included, such as best approach or specifying learning outcomes clearly. Despite all these, robotics has been already used to teach diverse topics, such as Geography (Serholt, et al., 2014), Geometry (Walker & Bureson, 2012), Maths (Hussain, Lindh, & Shukur, 2006), Physics (Church, Ford, & Rogers, 2010) among others, with a high predominance in physics and programming. Despite all of these works, most of them are not well documented, which reduce their availability and therefore replication in other educational context.

This does not mean that researchers are not aware of were not aware of these and other weaknesses. For example, the project TERECoP (Alimisis, et al., 2012) presented a constructivist methodology for

teacher training in the use of robotics in education. Several training sessions were available across Europe. Nevertheless, this approach focuses on face-to-face training to teachers and it is linked with Lego Mindstorms. Others have come with frameworks to establish precise procedures that have to be followed to create an activity with robotics. This is the case of Roberta initiative (Bredenfel & Leimbach, 2010), which established specific criteria for the activities that could have the brand Roberta and more important the teachers. Although these approaches are beneficial in the long term, it is still required materials that could increase the use of robotics in a critical way that considers benefits of the technology and pedagogical methodologies.

Educational Robotics for Science, Technology, Engineer and Education (ER4STEM) is a European project that aims to realize a creative and critical use of Educational Robotics (ER) to maintain children's curiosity in the world. ER4STEM has adopted constructionism as a foundational approach to designing workshops, robotic solutions and in the development of an integrated framework for inclusive learning and engagement with STEM. The project partners have found fundamental value in designing a variety of approaches, thus each workshop implements activities that foster students to discuss, argue and communicate their ideas about STEM concepts in a meaningful context for them. Consequently, the framework created in ER4STEM aims to make the explicit connection among pedagogical methodologies, knowledge in robotics, and 21st century skills.

Frameworks in Educational Robotics (ER)

There is a limited number of works that offer a clear guideline on the correct use of robotics in education, especially on the connection between technology and pedagogy. Roberta initiative (Bredenfel & Leimbach, 2010) aims to create a gender-balance didactic material and course concept. It specifies several characteristics that teachers and activities must have to be considered as Roberta teacher and activity, respectively. These characteristics could be cluster in four main areas: activity and teacher characteristics, design ideas, and quality criteria. The design ideas for an activity are: selection of interesting topics, provide examples, allow rapid achievements, and strength participants' self-confidence. Once the activity is created, it has to fulfil the following requirements: last from 2 to more than 40 hours, be suitable for mixed groups, be connected to real problems, and be certified by the initiative.

Another framework is created by (Chiou, Lye, Lai, & Wong, 2011), called EARLY. Their framework is based on the work done by (Carroll, 2002), that identifies four critical components in activities that involve technology. These components are: people, activities, context and technology. As a consequence, the EARLY framework describes three basic components: participants (i.e. teachers, learners, developers and experimenters), environment (i.e. computer, material, software and robot) and arena (e.g. problem based arena and soccer). A final element called scope embraces all of them to describe a specific situation or activity. Although the authors present five different case studies, the framework lacks literature support and formal evaluation.

The Educational Robotic Applications (ERA) is a framework created by (Catlin & Blamires, 2010) that postulates ten principles for the correct use of robotics in education. They grouped these principles in three categories. (1) The Technology category where the principles are intelligence, interaction and embodiment. They are related to expect features that robots have and could improve the educational experience. (2) The Student aspect that focuses on engagement, sustainable learning and personalisation. (3) The Teacher category that covers pedagogy, curriculum and assessment, equity and practical. Although these are important aspects to be considered in any educational activity, the authors neither offer information in how they should be implemented nor consider difficulties that may arise in their use. Nevertheless, there is a clear direction on how to use these principles and, as the authors suggest, require supportive testing and evaluation.

ER4STEM Framework

The manner that ER is being presented lacks that guidance that can help people design, develop and implement activities in ER that uses pedagogical methodologies to inform any decision. However, ER involves a huge group of stakeholders. Therefore, the first task in ER4STEM was to determine who the

stakeholders in ER are. The stakeholders identified were (Angel-Fernandez, y otros, 2016): young people, young people parents, teachers, school boards, organizations offering educational robotics, educational researchers, robotics researchers, human computer interaction researchers and industry. This group of stakeholders is still too big, if it is to consider that each one of them has a different needs, requirements and objectives. This variety makes it difficult to address all at once. Therefore, it was decided to focus on those stakeholders who have a direct impact on the quality of the activities. This was decided because those stakeholders would provide information that could inform other interested parties to implement ER. Teachers, researchers, organizers of educational activities and industry have been identified as those stakeholders (Angel-Fernandez, y otros, 2017).

Based on their requirements and needs, and ER4STEM's aims, ER4STEM's researchers suggested that workshops and lessons must be treated as similar because the place where the activity is implemented should be transparent for the final users. In order to achieve this, any activity should have a clear learning outcomes and evidence of learning, which could be formal or informal. This has several benefits: (1) the activities designed and implemented as a workshop are easily implemented as lessons. The description of objectives and proof of learning makes it easier for teachers to link the activity with any school's curriculum. (2) The evidence of learning allows people to verify if the activity is reaching the expected results or not. Also it could be used to measure the real impact of ER, which has not been quantified yet (Fabiane & Barreto, 2012) and it would generate arguments towards the use of ER in formal settings. As a consequence all activities done under ER4STEM, and hopefully in all ER, must be *pedagogical activities*, which have the following characteristics: (1) Clear learning outcomes and evidence of learning, which could be formal (e.g. assessment) or informal (e.g. write to a friend about what you have done today). (2) Use of one or more pedagogical methodologies during the activity, which has to be thought during the design of the activity and refine after the implementation of it. (3) Description of the activity using the activity template created in the project (Yiannoutsou, Nikitopoulou, Kynigos, Gueorguiev, & Angel-Fernandez, 2016). This will help other stakeholders to have a clear idea of all considerations taken into account and the assumptions done by the designer.

As a consequence of these all elements already presented, weaknesses of current approaches and industry requirements, ER4STEM's framework is a work on progress that aims to guide any ER's stakeholder on the design or adaptation, implementation and evaluation of pedagogical activities. This is achieved through the explicit connection among pedagogical methodologies, knowledge in robotics, and 21st century skills. To achieve this, the ER4STEM's framework provides four components, such as it is depicted in Figure 1. (1) An ontology of ER. The concept ontology in this case must be understood as it is done in Computer Science. This ontology provides specific definition of each word used in the field and the connection between them. (2) Tools created specifically to be used in ER, such as a web-repository, activity template and activity blocks. The last is a piece of activities that have been proven to be useful to foster specific skills and could be connected with other blocks to create a pedagogical activity. (3) Values or pillar of ER4STEM were selected from the industrial's needs, literature review and project's objectives. These values are: creativity, collaboration, communication, critical thinking, evidence of learning, mixed gender teams, multiple entry points, changing and sustaining attitudes to STEM, and differentiation. (4) Processes for workshops and conferences for young people.

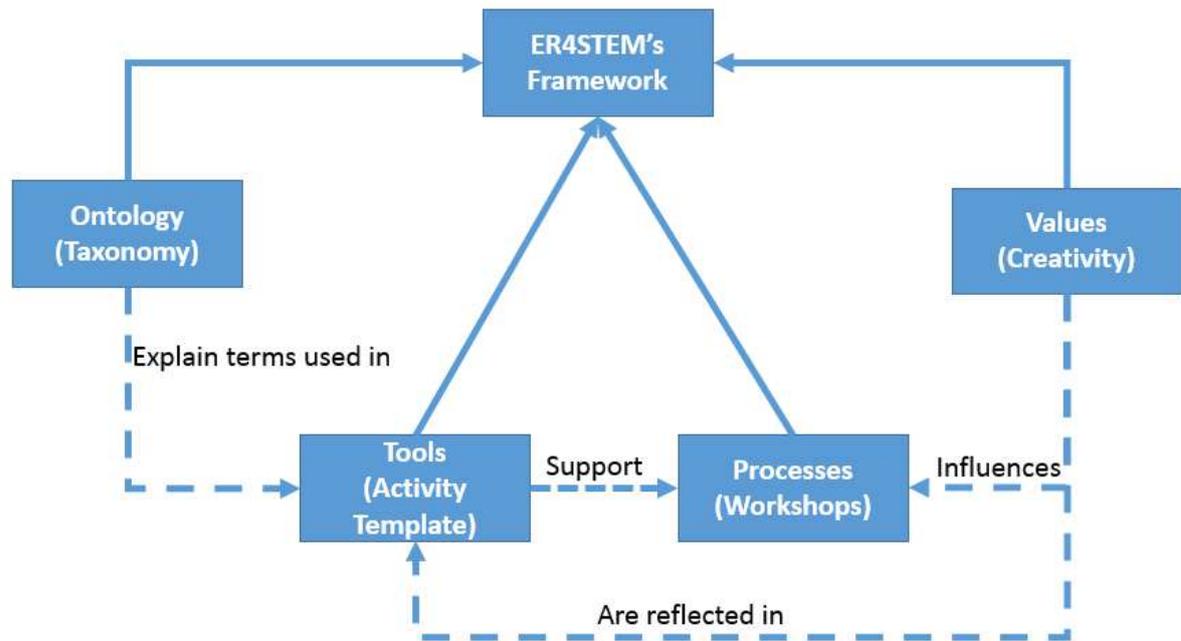


Figure 1. Graphical representation of the elements that compound ER4STEM's framework. The dash arrows represent the connections between elements that constitute the framework. The other lines come from different elements to the framework because they constitute the framework.

Values of ER4STEM

The values of ER4STEM were selected after doing a literature review, analyzing current industry's needs and project objectives. From the *literature review*, several weaknesses on how works in ER were identified (Angel-Fernandez, y otros, 2016). Thus: (1) There is not a clear evidence how pedagogical theories were considered during the design of the activity. (2) Activities reported in many cases are not fully described and therefore limiting their replication. (3) Some of the studies lack rigorous and systematic analysis of the data, which would make it become anecdotal. On the other hand, the *analysis of the industry* revealed that there is a common agreement that STEM is critical to the future economic growth. However, there are different views on whether the supply of STEM-skilled labour will be sufficient or not in the near future. According to Business Europe the lack of STEM-skilled labour will be one of the main obstacles to economic growth in the coming years (Europe, 2011). Therefore, the project objectives are four. (1) The provision of multiple entry-points to ER and STEAM. (2) Empowering children to solve real world problem and address all young children. (3) Provide a continuous STEM schedule. (4) Develop an open and conceptual framework. As a result, the values are: creativity, collaboration, communication, critical thinking, evidence of learning, mixed gender teams, multiple entry points, changing and sustaining attitudes to STEM, and differentiation. For each one of these values a literature review is been done to provide stakeholders with suggestions that have been already studied by other researchers.

An example of a value: Creativity

This is one of the skills that most of the people talk but it is difficult to explain in words. An important aspect to foster creativity is to avoid tell children that they are no creative just because the person does not consider that they are doing something new or innovative. Regarding this, it is important to remember that there are diverse level of creativity, for example (Kaufman & Beghetto, 2009) proposed four types of creativity: little-c, big-c, mini-c and pro-c. Little-c is the creative that involves novelty beyond individuals. Pro-c could be positioned between little-c and big-c, and it embedded ideas that are considered with significant valuable in their field but their contribution has not been recognized as big-c. Little-c, which occurs when individuals comes ideas that are new for them and for others but without a significant relevance to their field; and big-c, which occurs when individuals come with ideas that revolutionize their fields. Other important facts to remember are:

- The creation of environments, that promotes creativity, is also possible by
 - Defining clear goals in the activity (Csikszentmihalyi, 1996)
 - Balancing knowledge and challenge (Lewis, 2015) (Csikszentmihalyi, 1996). Too difficult or easy will not contribute in the development of creativity.
 - Create a climate where students are not concerned that they may fail (Lewis, 2015) (Csikszentmihalyi, 1996) (Sefertzi, 2000) (Vassileva, et al., 2012)
 - No creating competitions or providing rewards after finishing the activity (Lewis, 2015)
 - Motivating students to be creative (DeHan, 2009)
- Elements proposed by (Nelson, 2012) to foster creativity in robotics are:
 - Ability to visualize solutions, for example sketching or building prototypes of robots.
 - Thorough knowledge base in the domain, for example building on previous robotic projects
 - Ability to decompose and manipulate partial solutions
 - Ability to take informed risks, which include tasks with no right or wrong answers
 - Flexibility to try alternative techniques
 - Creativity friendly environment
 - Practice
- Failure most not be penalized (Sefertzi, 2000) (Lewis, 2015)
- Use of diverse tools to motivate creativity (Sefertzi, 2000), such us brainstorming, story boarding, lotus blossom, checklist, morphological analysis, and excursion technique.

Educational Robotics Ontology

An ontology as is presented by (Grimm, Abecker, Volker, & Studer, 2011) is a formal explicit specification of a domain of interest that could be executed by a machine and understood by humans. This representation is helpful in two ways. (1) It provides a specific definition of the concepts in the domain of interest. This will avoid misinterpretation of a concept that has different meaning depending of the field. For example, in ER4STEM when the idea of creating an ontology came, there were a misunderstanding between engineers and educational researchers because each one had a different definition of it. Also it will help stakeholders without knowing the concept to understand it. (2) It is the base of a semantic search on the repository, which would let it to provide better results to a query.

Thus in context, the ER4STEM's ontology was created in two steps. (1) Determining requirements and possible queries that should be answers and (2) Describing and formalizing the ontology. During the first step, it was decided to use the activity template as a base to determine concepts that must be in the ontology. On the other hand, the queries were created from diverse meetings between all partners in ER4STEM. This allow the discussion between researchers, practitioners and industry, which contribute to have different perspectives. The final questions are:

- What kind of activity I can use to for participants between x and y?
- Which activity I can use to improve an X skill?
- Which activities I can implement with an X robotic platform?
- What platforms I can use with Y programming language?
- What type of activities I uses an X pedagogical methodology?
- Which activities I can use for participants with X, Y and Z characteristics?

Based on requirement analysis, it was firstly decided to focus on concepts that are intrinsically embedded in ER, and avoid concepts and terms that unequivocally do be described in other ontologies that could not add any additional value to the base of knowledge. For example robotics or technology ontologies. The second step was initiated with the creation of a beta version of the ontology. This version was discussed with educational experts from University of Athens and Cardiff University, who provided corrections to the educational concepts. Taking into account their comments, a new version of the ontology was created and shared with all partners to have a feedback from them. This feedback lead to the first stable version of the ontology. The taxonomy and its relations are presented in the web-repository.

Tools

In ER4STEM three tools have been created to support stakeholders in ER. (1) *The activity template* is a generic design instrument that identifies critical elements of teaching and learning with robotics based in theory and practice (Yiannoutsou, Nikitopoulou, Kynigos, Gueorguiev, & Angel-Fernandez, 2016). It was designed to be a mediating artefact between pedagogical experts and the ER4STEM partners interested in design activity plans for ER. The template addresses the following aspects: a) the description of the activity, with explicit reference to the domains involved, objectives, duration and necessary materials; b) a level of detail that will demonstrate the influence of a specific approach. (2) *Activity blocks* were designed the outcomes of the first year of the project. They focus on the practical aspect of the activity plan. The activity blocks are adjustable short activities that were selected as good activities that could be used to foster one of the ER4STEM's values. (3) *Repository* is the digital representation of *activity template*, *activity blocks* and *ontology*. The repository's main objective is to support other people in the creation of new activities and inspire them ideas that other users have shared. Figure 2 presents the front page of the repository, which has the option to login in case the user wants to share their activity. Also there is the possibility to visualize diverse activities that already exist in the repository. Also it is also possible to search for specific key words or features, such as age.



Figure 2. Front page of the ER4STEM's repository.

Processes

A macro process was created base on research cycle and the professional teaching and learning cycle (Laboratory, 2008). The main aim was to conceive a suitable structure that could be used in activities that involves the use of robots. The final result is depicted in Figure 3. As it could be seen this process is composed of four main macro phases: design or adaptation of an activity plan, implementation in real

settings, activity's evaluation or assessment, and improvement of the activity plan. The first macro phase is divided in two possible steps, which represents the possibility to design an activity from scratch or adapt one from other existing activities. The second macro phase is implementation, which mainly focuses on considerations involving the settings and the context in which the activity is going to take place. The third phase provides instruments and procedures for evaluating the implementation. The fourth and last macro phase focuses on possible improvements of the activity plan based on information derived from the implementation in real settings, on reflections from the teachers, the students and the designers. Once the activity has been improved, there is the possibility to being implemented again as an activity for future groups.

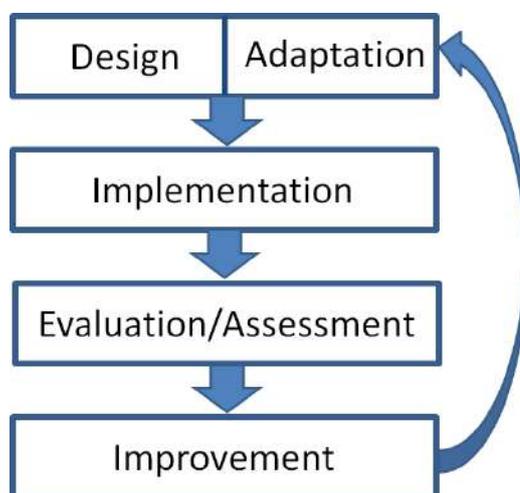


Figure 3. Framework's macro process definition

Using this macro process as reference there has been identified two processes can be created from the project experience: conferences and competitions, and pedagogical activities.

Acknowledgements

This work is funded by the European Commission through the Horizon 2020 Program (H2020, Grant agreement no: 665972). Project Educational Robotics for STEM: ER4STEM. Also the authors would like to thank all researchers, who have participated in all conversations that allow the specification of the framework.

References

- Alimisis, D., Arlegui, J., Fava, N., Frangou, S., Lonita, S., Menegatti, E., . . . Pina, A. (2012). Introducing Robotics to Teachers School: experiences from the TERECoP project. Constructionism. Paris.
- Angel-Fernandez, J. M., Kynigos, C., Lepuschitz, W., Pullicino, J., Grizioti, M., Girvan, C., & Todorova, C. (2017). Towards an Extended Definition of ER4STEM Framework.
- Angel-Fernandez, J. M., Lammer, L., Kynigos, C., Gueorguiev, I., Varbanov, P., Lepuschitz, W., . . . Vrba, P. (2016). Best Practice and Requirements.
- Bredenfel, A., & Leimbach, T. (2010). The Roberta Initiative. Conference on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS (SIMPACT 2010). Darmstadt.
- Carroll, J. M. (2002). Human-Computer Interaction in the New Millennium. Addison-Wesley Professional.
- Catlin, D., & Blamires, M. (2010). The Principles of Educational Robotic Applications (ERA). Constructionism, (pp. 1-17). Paris.

- Chiou, A., Lye, N. C., Lai, R., & Wong, K. W. (2011). Framework for robotics in education: Some experiences and case studies in test arena based projects. IEEE International Conference on e-Learning in Industrial Electronics (ICELIE). Melbourne.
- Church, W. J., Ford, T., & Rogers, C. (2010). Physics with robotics Using LEGE MINDSTORM in High School Education. Educational Robotics and Beyond. Stanford.
- Csikszentmihalyi, M. (1996). Flow and the psychology of discovery and invention. In *Creativity* (pp. 107-126). New York: Harper/Collins.
- DeHan, R. L. (2009). Teaching Creativity and Inventive Problem Solving in Science. *CBE Life Science Education*, 172-181.
- Europe, B. (2011). Plugging the Skills Gap – The clock is ticking (science, technology and maths). Brussels: Business Europe.
- Fabiane, B., & Barreto, V. (2012). Exploring the Educational Potential of Robotics in Schools. *Computers and Education*, 58(3), 978-988.
- Grimm, S., Abecker, A., Volker, J., & Studer, R. (2011). Ontologies and the Semantic Web. In *Handbook of Semantic Web Technologies* (pp. 509-579). Berlin: Springer.
- Hussain, S., Lindh, J., & Shukur, G. (2006). The Effect of LEGO Training on Pupils' School Performance in Mathematics, Problem Solving Ability and Attitude: Swedish Data. *Educational Technology & Society*, 9(3), 182-194.
- Kaufman, J. C., & Beghetto, R. A. (2009). Beyond Big and Little: The Four C model of Creativity. *Review of General Psychology*, 1-12.
- Laboratory, S. E. (2008). *The Professional Teaching and Learning Cycle*. Austin: Southwest Educational Development Laboratory.
- Lewis, T. (2015). Creativity - A Framework for the Design/Problem Solving Discourse in Technology Education. *Journal of Technology Education*, 35-52.
- Nelson, C. A. (2012). Generating Transferable Skills in STEM through Educational Robotics Robots. In *K-12 Education: A New Technology for Learning* (pp. 54-65). Hershey: IGI Global.
- Papert, S. (1980). *MINDSTORMS: Children, Computers, and Powerful Ideas*. New York: Basic Books, Inc.
- Sefertzi, E. (2000). Creativity. Retrieved from http://www.adi.pt/docs/innoregio_creativity-en.pdf
- Serholt, S., Barendregt, W., Leite, L., Hastie, H., Jones, A., Paiva, A., . . . Castellano, G. (2014). Teachers' Views on the Use of Empathic Robotic Tutors in the Classroom. IEEE International Symposium on Robot and Human Interactive Communication. Edinburgh.
- Vassileva, M., Sharkova, A., Laister, J., Zörwer, B., Arrizabalaga, E., Uriarte, X., . . . Beatty, E. (2012). *Creativity Development and innovation: Hadbook for SMEs*.
- Walker, E., & Burlison, W. (2012). User-Centered Design of a Teachable Robot. International Conference on Intelligent Tutoring Systems. Crete.
- Yiannoutsou, N., Nikitopoulou, S., Kynigos, C., Gueorguiev, I., & Angel-Fernandez, J. (2016). Activity Plan Template: a mediating tool for supporting learning desig with robotics. International Conference on Robotics in Education 2016. Vienna, Austria.

Think, Create and Program: Evolving to a K-9 Nationwide Computational Thinking Curriculum in Costa Rica

Carol Angulo, carol.angulo@fod.ac.cr

Alberto J. Cañas, acanas@gmail.com

Ana Gabriela Castro, ana.castro@fod.ac.cr

Leda Muñoz, leda.munoz@fod.ac.cr

Natalia Zamora, natalia.zamora@fod.ac.cr

Omar Dengo Foundation, Costa Rica

Abstract

As early as 1988, the National Program of Educational Informatics (PRONIE MEP-FOD) led by the Omar Dengo Foundation in partnership with the Ministry of Public Education of Costa Rica, has implemented Papert's constructionism ideas and computer programming as part of the public schools curriculum, as a means for students to learn through constructing. This large scale program currently benefits 87.6% of the country's K-9 students, who receive two weekly lessons.

In an effort to update and revise the Program to respond to the challenges brought about by the Fourth Industrial Revolution, and to stimulate the development of the skills our students need in order to fulfill future jobs that we cannot imagine today, we are enhancing the Program into a richer curriculum that includes the understanding of computational thinking where computer programming is a fundamental methodology to achieve and exercise higher-order thinking skills, while at the same time students learn and comprehend key concepts in 'computing', all within a constructionist learning environment. Computational thinking is addressed from a broad but also deep perspective, based on the definition of key concepts, skills and attitudes from preschool to ninth grade. After the definition of the learning outcomes and the designing processes, we have been piloting this new curriculum in 107 schools and have started extending it to 1300 in 2018.



Sixth grade student using Arduino

In this paper we present advances, including methodological and conceptual foundations of the updated educational proposal and the implementation of pilot studies.

Keywords

computational thinking; programming skills; computing concepts; educational informatics; competences; problem solving; higher order thinking skills; K-9 curriculum

Introduction

The National Program of Educational Informatics of the Omar Dengo Foundation in partnership with the Ministry of Public Education of Costa Rica (PRONIE MEP-FOD), constitutes a public-private collaborative effort which has been designated since 1988 with the task of introducing technology in the education system of the country with an emphasis in promoting higher order thinking skills. With a pathway of thirty years, the Program has been established as an important component of the educational policy of the country, driving the national public education forward, offering innovative education models that are based on technology, contributing to solve the digital gap, and preparing future generations to actively participate in the new digital society.

PRONIE MEP-FOD departs from the constructivist epistemological framework proposed by Jean Piaget, and the constructionism of Seymour Papert, which guides the pedagogical practices to use technology as a way to facilitate learning, and to promote cognitive development in children, teenagers, and teachers all over the country. The Program considers that the main potential of computers is their use as tools to increase people's capabilities to think, create and share. It aims to facilitate and promote students to become active users of technology and responsible creators.

As of December 2017, the Program had reached a total coverage of 87.6% of the K-9 students enrolled nationwide, benefiting 610,883 students from 3,174 schools. From these, 1,199 have a computer laboratory where Papert's pedagogical ideas are implemented, benefiting 476,565 students annually (356,617 elementary and 119,948 high school students). The work in the labs consists of a project-based strategy that guides the programming tasks, under the facilitation of an educational informatics educator that works with groups of 30 students approximately. The PRONIE MEP-FOD aims to reach 100% of schools in the next few years.

The incremental presence of digital technologies in all activities of the new society, opens the challenge of introducing these resources and especially their logic, foundations, and full appropriation to the new generations. This implies transforming programming, coding, and all skills associated with programming into a new literacy, a literacy as important as learning how to read and write. "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability." (Wing, 2006, p.33).

According to Costa Rica's Chamber of Information and Communication Technologies (CAMTIC) the visionary idea of the Program founders 30 years ago has been an important element favoring the evolution of the country's economy to one with a stronger and dynamic sector associated with technology. As a consequence, there is an estimated lack of 8000 computer scientists nationwide (El Financiero, 2017). Between June and August 2016, there was a supply of 5000 new jobs in companies related to technology in Costa Rica (La República, 2016), a condition similar to what is found in other countries around the world where the labor demand in informatics is wide and in constant growth. Besides looking for human talent in technological areas, companies also want their employees to have differentiated skills for new jobs, according to OECD (2016).

The accumulated experience of the PRONIE MEP-FOD in promoting skills like programming, problem solving, critical thinking and collaboration, has provided a solid basis from which to build a revised curriculum to address this new context. In this framework, PRONIE MEP-FOD designed didactic practices according to new demands and technologies and their incorporation to Costa Rican society through the public education system, ensuring equity and mutual benefits. Consequently, since 2014 the Program has evaluated and updated the educational informatics computer lab proposal, in order to encourage students to develop the skills required to become 21st century citizens.

Educational Informatics Laboratory: Think, create, program

After studying international trends and the evolution of educational proposals that incorporate technology to solve problems with programming, and revising the experience gained by the PRONIE MEP-FOD itself through three decades, four major competences linked to the expected learning outcomes were established. These are:

- **Problem-solving with programming:** to understand and solve problematic situations (for which the solutions are not immediately obvious) through strategies that imply computer programming.
- **Modeling and representation of data:** abstraction, modeling, and representation of data as the basis for communication and information-building through programming.
- **Understanding of concepts, operations, and components of computational and computer systems:** to understand (and manipulate) the main components of computer systems in an effective and safe way, through the understanding of its most basic theoretical foundations, vital aspects of the surroundings, and ethical and social implications.
- **Manufacturing of physical devices and robots:** to design and create tangible and interactive objects, made with different materials and technological resources, which can be programmed and controlled from a computer or mobile device.

Inspired by the constructionist didactics of *Creative Computing* (ScratchEd team at the Harvard Graduate School of Education, 2014), each of the activities designed for the ten grades consists of a designing process, personalizing process, sharing and collaborating process and reflection process (Bujanda, 2016), aiming at promoting meaningful learning, the development of computational thinking, and learning of computer science concepts, and following a competencies-based methodology. These competences became evident after the construction of a computational thinking concept map (Figure 1) that identified the computing concepts, programming skills and attitudinal elements (tolerance to frustration, preference for precision, learn from mistakes) aimed to pursue.

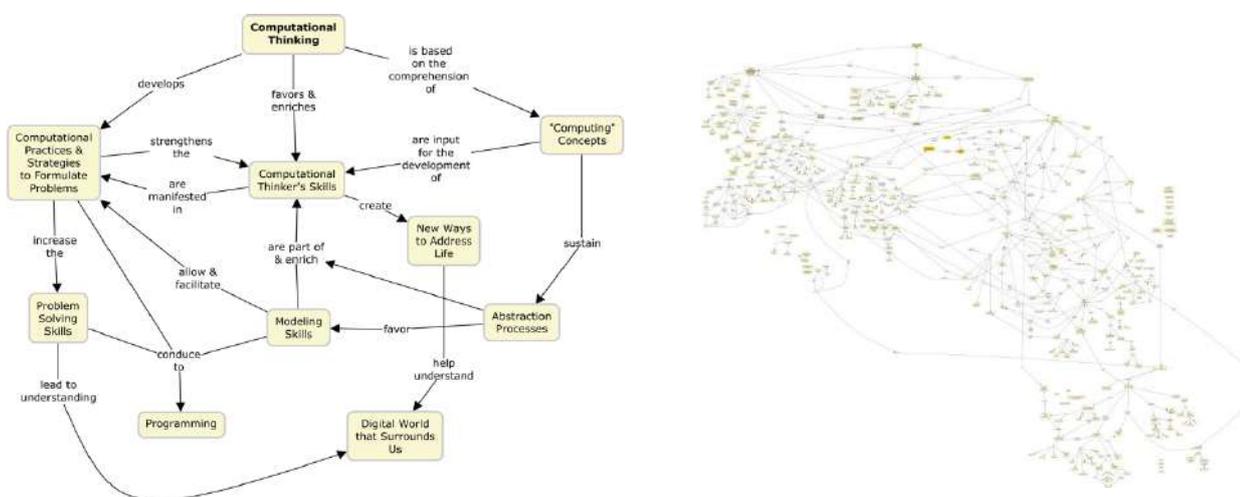


Figure 1. On the left, a high-level summary concept map on Computational Thinking; on the right, a purposefully unreadable image of the (work in progress) concept map used to identify the big-ideas that would guide the K-9 curriculum and the concepts associated with them gives an idea of the scope of the number of concepts considered.

As the competencies went beyond simple coding to include big ideas such as data processing, in-depth programming, abstractions and models, machines and programs, in addition to the attitudes and practices of the computational thinker, we found ourselves that our 30 years of experience provided us a good guideline in terms of the progression in programming we could expect from students through K-9, but with little to rely upon in terms of the identification of big ideas and associated concepts in computing, let alone the progression of the understanding of those big ideas and concepts throughout 10 years. To identify the big ideas and associated concepts with constructed the concept map shown in Figure 1. These conceptual big ideas enable the understanding of how current technologies work from their foundations, and it creates the basis to comprehend future technologies.

The big ideas and the associated concepts provide a framework that facilitates the work of the educational informatics teacher in guiding an integrated and coherent learning process; they provide a way for the teacher to help the student better understand what is behind the programming that the student is realizing. These ideas influenced the activities and units organized, identifying not only which key concepts should be learned but also their connection with other elements of the curriculum. For each big idea, a progression of that idea was built from kindergarten to 9th grade, and together conform, as a result, a progression for Computational Thinking, as defined by the proposal.

The concepts are to be made evident by the teacher as the student encounters them through programming, or as is appropriate through the advancement in programming skills, instead of being associated with a fixed curriculum. For this, the progression of concepts through each big idea is carried out through a block of years (grades) we called levels⁵⁷, generating a more flexible and fluid learning process, giving students opportunities to create links among the concepts covered in different levels, and to teachers and students as well, to easily detect the progress in the achievement of expected results. Teachers are made aware that certain level of understanding of a concept is expected at a particular block, but it is up to the teacher to determine when to make the concept evident to the student(s) or class.

We established a group of procedural, conceptual, and attitudinal indicators to obtain a description of the aimed competence and how it will be reflected in the students. These indicators guided the construction of didactic designs with two or three units and several activities for each year.

Additionally, a plan to train and update educational informatics teachers has been developed, and it is mainly delivered through virtual courses for scalability reasons.

The design of this new curriculum takes specific initiatives into consideration, such as the Maker Movement, which have a close relationship with the way we believe students build knowledge. In this regard, it promotes that people in the new culture should: make, share, give, learn, be equipped, play, participate, support, and change (Hatch, 2014). Accordingly, three grades in elementary school have been selected to incorporate resources associated with physical computing, such as Makey Makey, PicoBoard, Arduino Uno, and Circuit Playground, and preschool and eighth grade were selected to incorporate robotics, so that students can develop computational thinking while creating and programming robots to solve problems. In grades that do not incorporate physical computing, projects that involve game design or computer networking are proposed.

Starting with 4th, 5th and 6th grades in 400 schools in 2018, the project will be gradually implemented until all Educational Informatics labs in the country are covered by 2021.

Implementation and monitoring of pilots

We have been piloting this project since 2015, reaching 57 elementary schools and 50 high schools around the country. The pilot plan has been accompanied by a monitoring process that registers the reactions, constraints, likes and dislikes from students and teachers, which has been used to improve and adjust the designs.

The distribution of designs, monitoring and piloting per year, is detailed below:

2014: Design of the first and fourth grade proposals

2015: First and fourth grade proposals were piloted and monitored with 20 teachers. Second and fifth grade proposals were designed.

2016: Second and fifth grade proposal were piloted and monitored with 50 teachers. Third, sixth and seventh grade proposals were designed.

⁵⁷ Level 1: kindergarten, first, and second grades.

Level 2: third and fourth grades.

Level 3: fifth and sixth grades.

Level 4: seventh, eighth and ninth grades.

2017: Third, sixth and seventh grade proposals were piloted and monitored with 50 third and sixth grade teachers, and 20 seventh grade teachers. Preschool, eighth and ninth grade proposals were designed.

2018: Preschool, eighth and ninth grade proposals are being piloted and monitored, with 20 teachers in preschool and 40 teachers in eighth and ninth grades.

For each of the monitoring stages, we defined the main indicators to be evaluated, methods of data collection, and people in charge of the process to comply with the following monitoring objectives:

- General objective: Compile inputs that allow critical and timely evaluation of the updated proposal, for decision making and adjustments to the designs.
- Specific objectives:
 - To identify the relationship between the activities carried out in classroom and the recommended curricular proposal.
 - To review teachers' understanding of the main components of the proposal after training is received.
 - To assess didactic resources of proposal by teachers.
 - To identify the main constraints that affect the implementation of the proposal.
 - To assess likes, dislikes and understanding of the proposed activities by the students.

Data collection has been done through three specific channels: teacher survey, student survey and non-participatory observation.

Preliminary conclusions

Both the piloting phase and the recently initiated scaling up to the national level, will allow us to continue to learn about the strengths and weaknesses of the proposal, specially to evaluate how big ideas associated with computational thinking evolve in students, and their understanding of concepts and strengthening of programming skills for problem solving, in order to continue the necessary adjustments. However the preliminary results are stimulating so far, and signal of a robust and pertinent proposal to address the new digital society with all its challenges and opportunities.

After 30 years of learning through programming based on Papert's ideas and facing the fourth revolution, we evolved towards a new curriculum that further deepens programming, and is enriched with elements of physical computing and computational thinking concepts, all within a constructionist framework.

Papert's fundamental ideas are as pertinent and provocative, if not more, than 30 years ago, and the educational systems need to explore new ways, even new pedagogies such as this, to strengthen the leading role that they are called to have in this knowledge society.

References

- Angeli, C., Voogt, J., Webb, M., Cox, M., Malyn-Smith, J., and Zagami, J. (2016) A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Educational Technology & Society*, 19 (3), 47-45.
- Angulo, C. (2017) Design of the monitoring of the Update of the Educational Computing proposal. PRONIE MEP-FOD. San José, Costa Rica.
- Brennan, K., Balch, C. & Chung, M. (2011) Creative Computing. Harvard Graduate School of Education.
- Bujanda, M. E. (2016). Framework of expected learning results in students. PRONIE MEP-FOD. San José, Costa Rica.
- Bujanda, M. E., Pérez, Ó., Otárola, A., López, E., Matarrita, D., Angulo, C., Hernández, A. V. (2016) Review learning objectives. Update of the LIE proposal. PRONIE MEP-FOD. San José, Costa Rica.
- Cañas, A., Castro, A. G., Acuña, A. L., Rodríguez, A., Angulo, C., Matarrita, D., Zamora, N. (2017) Progression of powerful ideas associated with computational thinking. PRONIE MEP-FOD. San José, Costa Rica.

- Castro, A., & J. Lopez, F. O. (2017) Technology review report for use in eighth grade. PRONIE MEP-FOD. San José, Costa Rica.
- El Financiero (2017) Increase offer of computer scientists will require an additional effort in the academy. Retrieved from: http://www.elfinancierocr.com/tecnologia/Camtic-UCR-UNA-ITCR-Cenfotec-Ulacit-empleo_0_893310680.html.
- Fonseca, E. (2017) Observation report and surveys to Secondary Teachers, in training of the New Proposal LIE 7th level. PRONIE MEP-FOD. San José, Costa Rica.
- Fundación Omar Dengo. (2016) Digital technologies and capacities to build the future: contributions of the Program. PRONIE MEP-FOD. San José, Costa Rica.
- Fundación Omar Dengo (2016) Contribution of PRONIE MEP - FOD to the learning of students who graduate from II and III cycles of Basic General Education. PRONIE MEP-FOD. San José, Costa Rica.
- Hatch, M. (2014) The Maker Movement Manifesto. McGraw-Hill Education.
- Harel, I & Papert, S. (1991) Constructionism. Ablex publishing corporation norwood, New Jersey. USA.
- La República.net (2016) Engineering, IT and languages lead labor demand. Retrieved from: https://www.larepublica.net/noticia/ingenieria_informatica_e_idiomas_lideran_demanda_laboral
- OECD. (2016) Skills and Work. Retrieved from: <https://oecdskillsandwork.wordpress.com/2016/03/23/what-skills-do-employers-want/>
- Otárola, A., & Castro, A. G. (2017) Proposed entry and exit profile for the seventh level. San José, Costa Rica: Programa Nacional de Informática Educativa MEP-FOD.
- Otárola, A., Castro, A. G., & Pérez, Ó. (2017) Seventh Year Educational Proposal: "Imagine and design worlds in 3D". San José, Costa Rica: Programa Nacional de Informática Educativa MEP-FOD.
- Papert, S. (1980) MINDSTORMS: Children, Computers & Powerful ideas. Basic Books, Gálapago. Buenos Aires.
- Papert, S. (1995) The children's machine: Rethinking school in the age of the computer. Paidós, Barcelona.
- Quesada Solano, M. E., Cedeño Suárez, M. A., & Zamora Calvo, J. M. (2001) The Curricular design in the curricula: theoretical aspects and methodological guide. Heredia, C.R.: EUNA.
- Roanes, L. & Roanes, E. (2015) Turtle Geometry. Faculty of Education. Universidad Complutense de Madrid.
- Russell, S., & Norvig, P. (2003) Artificial Intelligence: A Modern Approach (2da edición). Upper Saddle River, Prentice Hall, New Jersey.
- Schwab, K. (2017) The Fourth Industrial Revolution. Crown Business. World Economic Forum. USA.
- ScratchEd team at the Harvard Graduate of Education. (2014) Creative Computing. Retrieved from: <http://scratched.gse.harvard.edu/guide/files/CreativeComputing20141015.pdf>
- Shannon, R., & Johannes, J. (1976) Systems simulation: the art and science. IEEE Transactions on Systems, Man and Cybernetics, 6(10), 723-724.
- Skills and Work, OECD (2016) Skills and Work, Understanding skills needed at the workplace. Retrieved from: <https://oecdskillsandwork.wordpress.com/about/>
- SparkFun. (2013) Analog vs. Digital. Retrieved from: <https://learn.sparkfun.com/tutorials/analog-vs-digital/digital-signals>
- Wing, J. (2006) Computational Thinking. Retrieved from: <https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>
- Wing, J. (2008) Computational thinking and thinking about computing. Computer Science Department, Carnegie Mellon University. USA.

Computational Thinking and Music Learning

Judith Bell, jbelle@chisnallwood.school.nz

Chisnallwood Intermediate School, Christchurch, New Zealand

Tim Bell, tim.bell@canterbury.ac.nz

University of Canterbury, New Zealand

Abstract

Computational thinking (CT) can be integrated with subjects outside computer science, and music is no exception. CT ideas such as decomposition, patterns, abstraction and algorithms can all be exercised in a meaningful way while at the same time engaging students with key concepts from music. This paper explores connections between computational thinking and music, and gives vignettes of creative ways to connect the two subjects in a manner that explores each subject in an authentic way.

The first example is sorting musical values (such as pitches and note lengths), which explores the ideas around sorting while exercising music reading and aural skills. We then explore the idea of giving commands to robots to position them on a giant staff to represent music; and data representation is explored by coding binary values using high and low pitches. Finally, students write computer programs to play scales and tunes, which forces them to think about both the rules around these musical concepts, as well as having to exercise programming discipline to make it sound correct and be adaptable to play in different keys.

These ideas provide students with ways to genuinely engage with both computational thinking *and* music. As well as the practical benefit of teaching two topics at once, it shows students how school subjects don't exist in isolation, and how there are aspects of thinking in common between the subjects.

Keywords

computational thinking; music theory; curriculum integration

Introduction

Computational Thinking (CT) is becoming part of school curricula in many countries under titles such as "Computing", "Digital Technologies", and "Computer Science" (Hubwieser, Giannakos, Berges, et al. 2015), which brings the challenge of fitting it into what is usually an already full school curriculum. This raises the concern that STEM subjects will overshadow traditional subjects that are important for a rounded education. However, the arts have an important role in STEM subjects because they encourage creativity, communication and teamwork.

In this paper we provide several vignettes of how these issues can be addressed in music education by integrating aspects of music education with Computational Thinking. Integration means that both subjects are being taught at the same time, which not only provides efficiency in the use of class time, but also helps students to avoid seeing subjects in isolation, and can potentially engage students who are more attracted to one of the elements than the other. The contrast may seem particularly strong when music and computer science are juxtaposed, particularly based on stereotypes of music as a creative art and computer science appearing to be a machine-centric science. In fact, they have more in common than might be expected from the stereotypes, both in terms of skills needed to be effective (creativity, teamwork, communication, working with notation), and also the opportunities to use one subject to engage and even enhance learning in the other. Furthermore, in practical situations both music and computer science involve constructing something that is intended for an audience; in computing the audience might be the users of an app or web page, or an organisation collecting data, while in music the audience might be at a concert, watching a film, or listening to an advertisement. Because both involve thinking about the needs of an audience and constructing something to meet those needs, when teaching them there are strong constructionist opportunities for student projects developing something with a real purpose.

The challenge is to design activities that include genuine music learning as well as genuine computational thinking. Simply using computers in a music class (such as making digital recordings or using online resources) is unlikely to help students engage at a deep level with computational thinking. Similarly, using music in a computer class (such as playing background music in a game or learning how to compress audio files) is unlikely to be teaching many key elements of music. There is a well-established body of work that achieves a strong link between CT and music by using computer programming to generate music, such as “Media computation” (e.g. Guzdial, 2003; Greher & Heines, 2014) and using systems such as EarSketch (Engelman, Magerko, McKlin, et al., 2017) and Sonic Pi (Aaron, Blackwell, & Burnard 2016), but we will explore other approaches here.

The elements of CT and music

In order to be sure that an exercise is teaching both CT and music, we need to be aware of the kinds of concepts that are covered by these two areas of learning.

Curricula based on CT generally include learning to create new artefacts on digital devices, as well as using the devices. This often includes working with and for others, which has been argued to be an important part of this discipline (Kafai, 2016). There are many definitions of CT, and even some debate about what should be included (Tedre and Denning, 2016), but there are common elements that appear in most definitions. For example, Selby and Woollard (2013) use the following list: algorithmic thinking; abstraction; decomposition; generalization and evaluation. Computer programming isn't listed explicitly here, although learning to program does cover these concepts well, and some would argue that it defines the scope of CT (Denning, 2017). In this paper we evaluate the relevance of each activity against these CT criteria, including programming, since this gives concrete evidence that they are likely to genuinely support CT in a computing curriculum. Our examples also provide broader contexts in which students can engage with CT.

A music curriculum, at a high level, will typically cover creating, performing, responding and connecting (Kaschub and Smith, 2016), and the elements of music that define the scope of curriculum are often articulated as a list such as pitch, timbre, texture, dynamics, duration, tempo, and structure (Burton, 2015). As with CT, the elements do not provide a curriculum, but they help us to identify if something belongs in a music curriculum. For a general classroom course these are likely to be made accessible through the use of basic notation, accessible instruments, and meaningful contexts such as popular music, film music, and local culturally relevant music.

Again, we will evaluate activities against these criteria to ensure they match the concepts that are likely to appear in school music curricula. We note that these are very atomic components of music, and are likely to be covered by broader descriptions. For example, pitch might come up in relation to understanding the range of a musical instrument, which in turn might be in the context of the instruments of an orchestra or other ensemble.

There are common elements between music and CT. In practical situations, both rely on notations in formal languages (for music that could include Common Music Notation on 5-line staves, tablature, solfège notation or graphic notation; for computing it includes programming languages as well as markup languages and protocols); and both use concepts around sequence (the order in which notes appear in time; and the order of statements in a computer program) and repetition (in music this includes repeats, as well as forms such as rondo or the structure of popular songs; in computing loops and recursion provide this).

However, we are not advocating that these related ideas should be used directly as common examples, as sometimes analogies and differences might add to confusion, but we do note that there are already related forms of thinking in both subjects.

In the remainder of this paper we give some vignettes of ways that music and CT can be combined in a way that genuinely engages students with both subjects. The first three exercises are built on activities from CS Unplugged (csunplugged.org), and the fourth is based on programming in a simple block-based language. The exercises mainly cover music theory, but also some composition.

Parallel sorting network

A parallel sorting network is an abstract concept for parallelising the task of ordering data into a sequence, usually in increasing order of value. It is a quintessential activity from the CS Unplugged activities (Bell, Rosamond, & Casey, 2012), where the network is drawn on the ground (or floor), and students traverse the network and come out in sorted order.

Figure 1 shows the layout of a 6-way sorting network. Six students enter the network at the left, each holding a card with a numeric value on it. When they meet at a node, they compare numbers, with the smaller value leaving to the left, and the larger one to the right. This simple rule is repeated each time they meet, and at the end the students emerge with their cards in ascending order.

Since sorting can be applied to any values that have a binary relation that is a total order, sorting networks can be used for keys other than numbers, including putting words or names into alphabetical order, or putting elements of a well-known story into the sequence in which they should occur. This opens up a number of possibilities in music.

One valuable approach is to have students compare written notes in Common Music Notation (Figure 2, left). Initially this can be simple notes that are on the same clef and only use lines and spaces (so the simple rule is that the note that is further up the staff should go to the right). This can be extended by adding accidentals, so that two notes on the same line might be distinguished by being sharp, natural, or flat. If the students have mixed levels of understanding, the "harder" notes (with accidentals) can be given to a more advanced student, so that if such a comparison is needed, the student holding the card will be able to explain it to the student that they meet each time. The next step up is to add ledger lines and different clefs (bass, alto, tenor). This approach could be used with a variety of notations, such as solfège, tablature or numbered music notation, and even mixing up different notations!

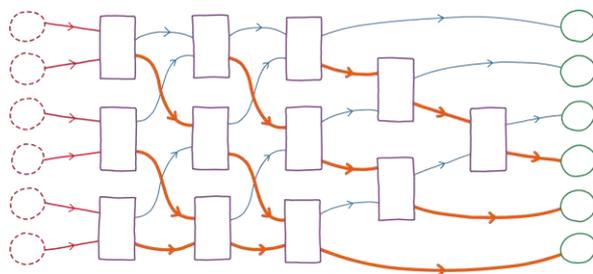


Figure 1. A 6-way parallel sorting network

The next extension is to compare note and rest lengths (students are given quavers/crotchets/minims and so on, with tied and dotted notes used as an extension). It can also apply to dynamics (*ff*, *f*, *mf*, *mp* etc.) and tempo indications (Allegro, Andante, Presto etc.). Aural skills can be exercised if the students are given instruments that make a sound; we have found tuned bells to be particularly effective, as it is possible to get bells that are the same size but have different pitches (Figure 2, right). This forces students to make pitch comparisons over and over, and success is measured if the bells come out in ascending order.

While a fixed sorting network itself is fairly easy to master, we have been surprised that students enjoy using it regularly. They can try out different things to compare, and reason about the meaning of the order for different musical elements. The activity has a built in gamification in that students want to complete the task as quickly as possible, but if they are inaccurate because of speed then the team doesn't achieve a successful outcome.



Figure 2. Comparing musical notes (left) and bell pitches (right) in a sorting network

From a computational point of view, students can encounter the factorial number of orders that the input can start in, and the idea that two identical values end up being sorted together. It touches on every aspect of CT, but is particularly strong in abstraction (the algorithm works regardless of what is being sorted), decomposition (a sophisticated outcome is broken into very simple comparison steps), and logic (trying to explain, for example, why the smallest value will always find its way to the correct node).

It is also an effective music education tool - students are repeatedly gaining experience with the notation for pitch and rhythm, and exercising their aural skills in a motivating context; and as they become comfortable with a notation, it can be extended by adding more difficult variations. When using this we have repeatedly found that students are able to meaningfully grasp music concepts beyond their current music theory level, which results in much deeper understanding of their current knowledge.

Positioning robots

An activity based on simple turtle-style robots is to draw a large staff on the ground (Figure 2), and have students program simple robots to go to the location of a specified note on the staff. The robots in Figure 3 are “BeeBots”, a simple device that can be programmed using only four commands (forward, back, left, right) that can be entered as a sequence. These commands are then followed when the “go” button is pressed, which is equivalent to running a program. The BeeBot moves 15cm for each forward/back command, so with the stavelines 30cm apart, each movement corresponds to one space or line on the staff.

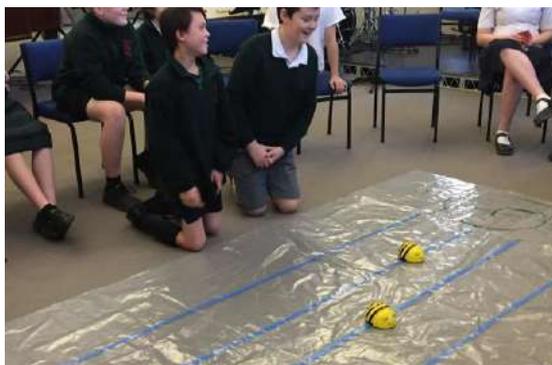


Figure 3. Students using BeeBots to show a musical interval

Initially the BeeBot only needs to move forward to the given note (e.g. “get the BeeBot to go to D”), but this can be extended into notating a short tune either with the BeeBot pausing on each note of the tune and spinning 360 degrees, or having multiple BeeBots. This now requires programming the position in two dimensions based on the simple movements.

In our experience, students initially make mistakes with the programming, but are determined to get to the right note. After a while they become adept at the “language” of the BeeBots, and some have even taken the risk of sending the BeeBot on longer paths than needed to introduce some humour to the exercise. This approach is easiest using tunes based on a diatonic C major scale, although representing sharps and flats can be done by the direction the BeeBot is pointing (left for flat, right for sharp).

Another variation is to use two BeeBots, and work with intervals, either programming them to show an interval (possibly given aurally), or having other students name the interval that has been displayed. This can be extended further by having one group of students program a chord with three or four BeeBots either vertically aligned (harmonic) or in a horizontal pattern (melodic), which also provides the challenge of avoiding collisions. We then have another group play the chord on a piano, and/or name the chord (starting with simple triads, and extending to other types of chords, including inversions, 7ths, sus chords and so on). A further extension is to change the clef.

A challenge that exercises *evaluation* from CT is to try to make all the BeeBots arrive at their final destination at the same time. This would require students to accurately determine the number of steps in the “program” before it is run, and pad out the steps for shorter routes to match the longest one. This exposes students to the idea that we could analyse the running time of an algorithm without running it.

This exercise motivates students to repeatedly use the fundamental idea in programming of sequences, with the concomitant CT elements, particularly algorithmic thinking, and decomposition of the path into steps. Musically, students are working with rudiments of music theory, particularly pitch notation, accidentals, clefs, intervals, and triads/chords. They also need to demonstrate teamwork to create “chords”.

Another variation of this is to create ukulele or guitar diagram grids for students to program the BeeBots to land on chord shapes. In this case the grid lines are 15cm apart, representing the strings and frets.

Codes based on sound

The idea that all data on computers (text, sound, images, numbers etc.) is represented as binary numbers is fundamental to the idea of a *digital* device. The CS Unplugged activities include a popular way to introduce binary numbers with minimal mathematical background required (in fact, the main prerequisite is being able to count up to 31). This is done by having students work with 5 cards, with 16, 8, 4, 2 and 1 dots on one side of them respectively. The Unplugged website gives a lesson plan for a constructivist approach to show students the relationship between decimal numbers and binary representation by flipping over the cards as a binary representation.

This segues to representing letters of the alphabet using these numbers; for the 26-letter English alphabet students will usually suggest using 1 for A, 2 for B, 3 for C, and so on. For example, the binary number 11010 represents the number 26, or the letter Z. Since the 0 and 1 digits are abstract representations, this transitions easily to the idea of using other representations that have two values, including sound. A variety of alternatives can be explored (loud/soft, long/short, timbre, melodic patterns, low/high), with the latter (low/high) being effectively how a telephone modem works. Once students realise that letters of the alphabet can be transmitted using only high and low notes, a tune can be used to represent some text. As a simple example, the first exercise in the CS Unplugged “Modems” activity has a recording of a jazz singer singing this sequence, which students can decode (<https://www.youtube.com/watch?v=MOMXxRbpkjM>).

The concept of encoding messages into an apparently unrelated format is called “steganography”. The idea that this is even possible is an important opportunity to show students how innovative algorithms can achieve something that might not have been thought possible, and can motivate students to create their own hidden messages.

This leads to a constrained exercise in composition for students - they should first work out the binary representation that they would like to hide in a composition, and then write a meaningful piece of music that encodes the message. This is typically done using notes that are clearly high and low, but any other two-state representation could be used, such as whether each note is higher or lower than the previous one, a percussion part that has two sounds (such as a kick and snare drum), or a backing part that is ascending or descending. The composition could be performed live on acoustic instruments, entered into recording or notation software, or played using websites that allow students to create music using a range of graphical input formats.

As an intriguing variation, the hidden message could be a melody itself, represented with diatonic music numbers or MIDI! If MIDI is being used, the notes can be represented in 7 bits, and this could map on

to rhythms with 8 quavers (eighth notes) in a 4/4 bar (notes or rests for 1 or 0 respectively), with either the first or last quaver being chosen by the composer, and the other 7 based on the bits in the MIDI note. In this case, a melody is being represented by a rhythm.

This activity exercises a variety of CT concepts, including algorithms for converting between decimal and binary, abstraction where even representations may have different representations(!), and evaluation of the number of bits needed to represent a range of symbols (e.g. for alphabets of more than 26 characters).

Musically, it provides motivation for composition as well as some constraints. Students listening to steganographic recordings will be exercising their aural skills to recognise the binary digits that make up the message.

Programming note sequences

Music theory contains a lot of rules around sequences of notes, particularly scales, which are fundamental to many genres of music. These scales are based on repeating patterns, so they are amenable to being programmed. Most programming languages are able to play MIDI notes, which opens up the possibility of writing programs to play scales. Parameters could be added to determine aspects like the key, range and speed.

For example, Figure 4 shows a simple one-octave chromatic scale programmed in the Scratch language, starting on a C (midi note 60). Each note is 1 semitone higher than the previous one (the “change degree by 1” command). Students can be given this as an example, and asked to identify the scale. They can then “remix” the program (edit it) to make the degree change by 2 each time, creating a whole-tone scale, which has a distinct sound to it. Other step sizes produce various arpeggios (for example, steps of 3 produce diminished 7th arpeggios), and students could be challenged to make the notes ascend in octaves (12 steps, as there are 12 semitones in an octave), to write descending scales (negative steps), and identify them aurally.

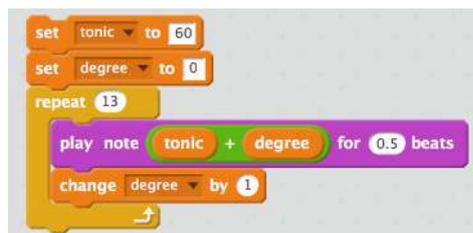


Figure 4. A first attempt at a Scratch program for a scale

Figure 4 shows how the choice of variable names provides a valuable form of integrated learning. The terminology (tonic and degree) for how the note is calculated articulates how the logic of the program works. In this example, the term “degree” is not quite musically accurate as it is the number of semitones above the tonic, rather than the degree. While such discussions might seem pedantic, they give the opportunity to work with the precise terminology in music, and to recognise the importance of accurately-named variables to make a program understandable.

The next challenge is to have students adapt the program for other scales (such as major, minor, and modes). These will involve a mixture of intervals between notes for each octave (for example, a major scale increases the pitch of each note by a tone, tone, semitone, tone, tone, tone, semitone respectively) to give 8 notes. Programming this forces students to think about each degree of the scale, and they can debug their program by listening to it and checking that the scale sounds like a conventional major scale. They could also write a program to play a sequence of scales (e.g. C major, then C# major, D major, and so on), which provides opportunities to work with nested loops. For more advanced programmers, this also raises the possibility of storing the degrees of the scale in a list, and looking up the degree in the list, which makes it easy to move up and down a scale.

This exercise engages students with loops and variables in a way that there are precise desired outcomes (such as playing a particular scale). The need for accuracy playing scales can transfer from musical training to the need for the program to do precisely what is specified. As well as working with algorithms, students are using abstract representations of notes, and are also using decomposition to break scales into their elements.

Another valuable activity is to have students write programs to play a tune, and use musical phrases that can be represented as functions (“More Blocks” in Scratch), so that they are both aware of the structure of the music and using the CT concept of decomposition.

Musically, students are developing their knowledge of music theory (such as the intervals between notes in each scale and use of musical phrases) as well as their aural skills.

Conclusion

In addition to the natural connections between music and CT, we have given four examples of exercises that can be used to engage students meaningfully with both subjects in an integrated manner. This provides ways to involve students who might be more interested in one aspect than the other, and also means that teaching time is being used to cover two subjects at once. They may also help teachers who feel confident in one subject to explore the other subject with students. In practice, it is likely that this would benefit from a collaboration between a computing teacher and a music teacher if each subject is to be explored in depth.

These examples only touch on aspects of each curriculum. We recognise there are other ways that CT and music can be combined, particularly using programming languages to create compositions, either offline or as a live performance. Computational thinking is a relatively new subject in schools, and we look forward to new ideas appearing as teaching collaborations and teachers with cross-curricula interests create new activities that meaningfully combine different subjects.

References

- Aaron, S., Blackwell, A. F., & Burnard, P. (2016). The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music, Technology & Education*, 9(1), 75-94.
- Bell, T., Rosamond, F., & Casey, N. (2012). Computer Science Unplugged and related projects in math and computer science popularization. In H. L. Bodlaender, R. Downey, F. V Fomin, & D. Marx (Eds.), *The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the occasion of his 60th birthday* (Vol. LNCS 7370, pp. 398–456). Heidelberg: Springer-Verlag, Berlin, Heidelberg.
- Burton, R. L. (2015). The elements of music: what are they, and who cares? In *The Australian Society for Music Education National XXth Conference Proceedings* (pp. 22–28).
- Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33–39.
- Engelman, S., Magerko, B., McKlin, T., Miller, M., Edwards, D., & Freeman, J. (2017, March). Creativity in Authentic STEAM Education with EarSketch. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 183-188). ACM.
- Greher, G. R., & Heines, J. M. (2014). *Computational thinking in sound: Teaching the art and science of music and technology*. Oxford University Press.
- Guzdial, M. (2003, June). A media computation course for non-majors. In *ACM SIGCSE Bulletin* (Vol. 35, No. 3, pp. 104-108). ACM.
- Hubwieser, Peter, Michail N. Giannakos, Marc Berges, Torsten Brinda, Ira Diethelm, Johannes Magenheimer, Yogendra Pal, Jana Jackova, and Egle Jasute.

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., & Jasute, E. (2015). A Global Snapshot of Computer Science Education in K-12 Schools. In *Proceedings of the 2015 ITiCSE on Working Group Reports - ITiCSE-WGR '15* (pp. 65–83). New York, New York, USA: ACM Press.

Kafai, Y. B. (2016). From Computational Thinking to Computational Participation in K–12 Education. *Commun. ACM*, 59(8), 26–27.

Kaschub, M., & Smith, J. P. (2016). The Big Picture: Developing Musical Capacities. *Music Educators Journal*, 102(3), 33-40.

Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition. Available from <http://eprints.soton.ac.uk/356481>.

Tedre, M., & Denning, P. J. (2016). The Long Quest for Computational Thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research* (pp. 120–129).

Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics

Chantal Buteau, *cbuteau@brocku.ca*
Brock University, Canada

Ana Isabel Sacristán, *asacrist@cinvestav.mx*
Cinvestav, Mexico

Eric Muller, *emuller@brocku.ca*
Brock University, Canada

Abstract

In this practice report, we reflect and discuss the roles of, and demands on university instructors in three undergraduate mathematics computer-based courses implemented since 2001 at Brock University, Canada. These are the *Mathematics Integrated with Computers and Applications (MICA) I, II, III* courses, in which instructors create an environment that supports students' constructionist learning experiences as they design, program, and use interactive environments (i.e., microworlds) to learn and do mathematics. Using Ruthven's (2009) model on the professional adaptation of classroom practice with technology, we feature constructionist characteristics of the course design highlighting the shift from traditional, instructionist pedagogy towards one of empowering students. Since there seem to be relatively few sustained implementations of microworlds in mathematics instruction (Healy & Kynigos, 2010), this report, grounded on a continuous practice of over 15 years, contributes to our understanding of roles and demands of "ordinary" instructors in the "real" classroom, who have aimed at creating an environment for supporting students' constructionist learning experiences. In particular, this report highlights the instructor's demanding role in these student-centred courses, more so since students select their own topics for their last project in lieu of final exam, thereby having the opportunity of it being meaningful to them.

Keywords

computational thinking; constructionism; mathematics; programming; microworlds; university; teachers

Introduction

In this practice report, we reflect on the roles of, and demands on university instructors in undergraduate mathematics computer-based courses – the *Mathematics Integrated with Computers and Applications (MICA) I-II-III* courses – carried out at Brock University, Canada (Ralph, 2001; Buteau, Muller, & Ralph, 2015). These courses follow the constructionist paradigm (Papert & Harel, 1991), requiring students to design and program computational objects for mathematical learning, and have had sustained implementation for over 15 years (Buteau, Muller & Marshall, 2015).

More specifically, at Brock University, mathematics majors and future mathematics teachers learn to design, program, and use interactive environments –called *Exploratory Objects*– for the investigation of mathematical concepts, conjectures, and theorems or real-world situations (Muller, Buteau, Ralph, & Mgombelo, 2009). During their first undergraduate years, students may enrol in a sequence of three MICA courses and create in total 14 Exploratory Objects as part of their course load (Buteau, Muller, Marshall, Sacristán, & Mgombelo, 2016). At the end of each term, students, individually or in groups of two or three, create an original Exploratory Object for which they select the topic (see MICA, 2018). As a result of a literature review study (Marshall & Buteau, 2014), we classified the development of Exploratory Objects as mathematical microworlds: they constitute open-ended exploratory computer activities (Edwards, 1995) where MICA students engage in computational thinking for mathematics (Buteau et al., 2016).

This paper provides a discussion, based on insightful reflections, on the “real” MICA classroom by particularly focusing on how the “ordinary” MICA instructors have created an environment that supports the students’ constructionist learning experiences. We are interested in discussing the constructionist experiences involved in the teaching of MICA courses. In the following, we first briefly lay out aspects of Constructionism, Computational Thinking and Microworlds that frame the pedagogical approach in the MICA classroom. Using Ruthven’s (2009) model of five key structuring features of school classroom practice, we then discuss demands on, and roles of instructors in these courses. This discussion focuses, first, on the overall courses and, second, on students’ final individual projects which are used in lieu of final exams. We end with a few concluding remarks.

Constructionism, Computational Thinking and Microworlds: A Theoretical Framework for MICA’s Pedagogical Approach in the Mathematics Classroom

As stated above, our MICA courses follow the constructionist paradigm; constructionism is defined by Papert and Harel (1991, p.1) in the following way:

Constructionism--the N word as opposed to the V word--shares constructivism's connotation of learning as 'building knowledge structures' irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe.

Thus, the basic principles of the constructionist paradigm involve learning situations or environments that are student-centred, where students build or construct shareable objects that are somehow “tangible”. These situations usually involve open projects, often computer-based ones, in which learners engage with meaningful “powerful ideas” (Papert, 1980) within a social context of collaboration, discussion or interaction among peers.

One concept that is linked to constructionism, is that of exploratory computational environments known as microworlds. A microworld is defined by diSessa (2000) as

a type of computational document aimed at embedding important ideas in a form that students can readily explore. The best microworlds have an easy-to-understand set of operations that students can use to engage tasks of value to them, and in doing so, they come to understanding powerful underlying principles. (p. 47)

For Papert (1980), microworlds involve objects “to think with” and “allow a human learner to exercise particular powerful ideas or intellectual skills” (p. 204) through exploration and discovery in a knowledge domain.

Two ways of describing microworlds are given by Edwards (1995): a “structural” definition which focuses on the design elements (e.g. collections of computational objects to model mathematical or physical properties of a domain; representations; and activities or challenges for students to explore in the domain); and a “functional” definition which highlights how students learn with microworlds, such as through the interaction between the student, the software, and the setting in which it is used.

Related to the latter, an important aspect for us is that related to the role of the teacher in constructionism: Papert (1980) stated that in a microworld situation “the relationship of the teacher to learner is very different: the teacher introduces the learner to the microworld in which discoveries will be made, rather than to the discovery itself” (p. 209); that is, the situation presented by the teacher is one to facilitate students to think like mathematicians rather than to teach them about mathematics.

Weintrop et al. (2016) remind us that “Papert (1996) was the first to use the term computational thinking to refer to the affordances of computational representations for expressing powerful ideas.” (p.130). These authors discuss extensively on how mathematicians and scientists have come to engage in computational thinking in their professional (including research) work, and argue that “the varied and

applied use of computational thinking by experts in the field provides a roadmap for what computational thinking instruction should include in the classroom” (p. 128).

In the next sections we present the structure and functioning of our MICA courses, highlighting the constructionist aspects and role of the instructors.

University Instructors in MICA Courses: Roles & Demands

To provide a structure for our reflection and discussion on demands on, and roles of instructors in MICA courses, we use Ruthven’s (2009) model of five key structuring features of school classroom practice to “*illuminate the professional adaptation which technology integration into classroom practice depends*” (p. 131).

Working environment

For MICA courses, instructors teach in regular lecture rooms where they mostly elaborate the mathematical content. They lead mathematics programming-based activities in computer laboratories (one computer per student). The course format is two hours per week for lectures and two hours per week for lab sessions. The class size is capped at 35 students per course section.

Resource system

Ruthven (2009) describes: “[t]he concept of ‘resource system’ focuses... on the combined operation of the mathematical tools and curriculum materials in classroom use, particularly on their compatibility and coherence of use, and on factors influencing this” (p.136). For their mathematics lectures, MICA I instructors use lecture notes which to date have been shared among instructors. In MICA II, instructors use lecture notes and possibly a mathematical modelling textbook depending on the instructor. For the lab sessions, programming software is used. Since MICA I students need to learn programming, the instructor uses a friendly textbook for the introduction of this technology (currently VB.NET in Visual Studio environment), in addition to lab activity guidelines (Ralph, 2017) that emphasize the connection of mathematics and programming (Buteau & Muller, 2014). In MICA II-III, instructors use only lab activity guidelines since students are then knowledgeable programmers. MICA instructors actually use programming in their own mathematical research, and therefore consider programming as an integral part in doing mathematics (i.e. they are engaging in computational-thinking-based mathematics research as described by Weintrop et al., 2016). In other words, because of their research practice, instructors naturally merge the mathematics and programming resources as a system. Individual instructors might need to learn the specific programming language. However, this is a not overwhelming particularly since within the lab sessions they can count on knowledgeable teaching assistants.

Activity format

Ruthven (2009) introduces the key structuring feature of activity format where “Classroom activity is organised around formats for action and interaction which frame the contributions of teacher and students to particular lesson segments (Burns & Anderson, 1987; Burns & Lash, 1986).” (p. 137).

Overall the MICA instructor needs to provide a learning environment in which students:

- design and program mathematics experiments/modeling/simulations with an appropriate interface in order to conduct an investigation;
- reflect on mathematical results and data in a written report.

(Buteau & Muller, 2010)

This constitutes what we call in this paper ‘a microworld approach’ to learn and do mathematics. In the first-year MICA I course, students learn the microworld approach, and in the upper-year MICA II-III courses, they apply it to broader and more sophisticated mathematics contexts (Buteau et al., 2016). The instructor in any MICA I – II - III provides an environment for students to experience constructionist learning: students learn by making and using these microworld projects. This includes a pedagogical approach where instructors guide students to carefully develop their visual interfaces to support their investigations (the construction is shareable). In fact, this overall aim for students to learn and apply the

microworld approach grounds the MICA course design and pedagogy (Buteau et al., 2015), and leads to empowering students to work as mathematicians (Broley, Buteau, & Muller, 2017; Buteau et al., 2016).

In MICA I lectures, the instructor elaborates on the mathematical content, although it is not presented quite in a traditional manner. For example, the instructor thinks out loud as a 'working mathematician' to make transparent the messy development of mathematical ideas. Or the content may be presented through selected examples in order to prompt students to ask mathematics questions and state conjectures. This is a shift from a traditional, instructionist approach towards one of empowering students. For the instructor, it requires acceptance of a slower pace in terms of mathematics content covered and surrendering some control (i.e., s/he acts as facilitator, not lecturer). In MICA II-III, the instructor follows a more regular lecture format. However, due to the smaller size of classes and the inquiry component of the course, lectures are often more interactive including aspects of an inquiry-based mathematics classroom (Rasmussen & Kwon, 2007); e.g. with interaction between instructor-students or among students, at structured times.

In MICA lab sessions, the instructor usually gives overall guidelines to the whole class for a short period of time (2-15 minutes), then leads students to their individual work, and when necessary, provides opportunities for the whole class to discuss a situation encountered by one or more students. Teaching assistants assist the instructor with students' individual work. This stresses the student-centred activity format of the lab sessions. Overall, and particularly in MICA I, instructors need sensitivity to individual students' development of their instrumental genesis (Trouche, 2004) during their programming and in their involvement in the mathematics microworlds (Buteau & Muller, 2014), stressing again the student-centred characteristic of MICA courses.

Curriculum script

When preparing or teaching lessons on a topic, instructors use their professional knowledge that includes, in particular, "a loosely ordered model of relevant goals and actions which serves to guide their teaching of the topic" (Ruthven, 2009, p.138).

MICA courses are not mathematics content-driven: the microworld approach to learn and do mathematics defines the courses (Buteau et al., 2015). The overall content is not 'traditional mathematics', and the overall script had to be rethought since the technology impacts 'what' mathematics is covered in MICA courses (Buteau et al., 2016). In MICA I, the course content is driven by students needing to learn programming and the microworld approach: the mathematics content provides context in which students learn those (Buteau & Muller, 2014). In particular, the learning of programming technology for mathematics follows a back-and-forth instrumental orchestration (Trouche, 2004) model (Buteau & Muller, 2014). The instructors' script also includes the component of debugging one's own program and for students to appreciate the value of debugging. In MICA II-III, each instructor involves mathematics content relevant to the microworld approach (i.e., a computational thinking approach) according to his/her own, evolving mathematics interests. As such, the instructors integrate authentic programming-based mathematics tasks (Buteau, Mgombelo, Muller, Rafiepour & Sacristán, submitted) which, in particular, includes inquiry practices (i.e., involving *using* the microworld). We associate this with Wagh, Cook-Whitt and Wilensky's (2017) position that "the constructionist approach of interacting with and manipulating program code of computational models can facilitate productive forms of [students'] engagement with inquiry-based science" (p.615).

Time economy

Because of the student-centered characteristic of the microworld approach defining the MICA courses, individual student guidance is required, and this means 'time'. To respond to this characteristic, the department limits enrolment to 35 students per section and provides two teaching assistants for the lab sessions to assist the instructor.

As mentioned before, the instructors need to accept and adapt, due to the nature of the MICA courses, to a slower pace in terms of mathematics content covered in lectures. Also, instructors use programming in their own research and do not require additional time to learn the technology (except if there is a change in the programming language). However, when teaching the MICA courses for the first time,

instructors need to prepare in terms of the change in the pedagogical paradigm required for the MICA implementation of the microworld approach (Papert, 1980; Muller et al., 2009). This is a shift in philosophy that is new to university mathematics instructors who normally mainly, if not strictly, concentrate on content. This shift requires time.

University instructors have a dual role for their departments as teachers and policy makers to create, delete, and modify courses in their curriculum (Barker et al., 2004). Because of the innovative nature at the time of the proposed course design, a professor in the department was awarded a course release to dedicate his time to design what became MICA courses (Buteau et al., 2015).

In the next section, we elaborate on the roles of and demands on the instructors required in the last part of the MICA courses for which the course format is significantly different.

Instructors Creating an Environment for Individual Projects Meaningful to Students: Roles & Demands

Each of the three MICA course terms culminates to a complete constructionist learning experience opportunity for every student. This is a very demanding and engaging exercise for both students and instructors.

Students, individually or in groups of two or three, create original microworld open projects for which they select their own topics. The MICA instructor encourages and insists that students select a topic of interest to them, i.e., something related to them, either an application of mathematics or a mathematical topic of interest. In other words, the instructor motivates students to engage in a project that is meaningful to them. For example, Matthew and Kylie wondered if it is better to walk or run in the rain (Figure 1, left) while Adam investigated the bounded area, as the exponent increases, of the iterative complex function defining the Mandelbrot set (Figure 1, right). See MICA (2018) for these and other examples. Since these original microworlds are meaningful to students and shareable constructions, we have observed that students show pride of, engagement in, and ownership of their own projects (Muller et al., 2009).

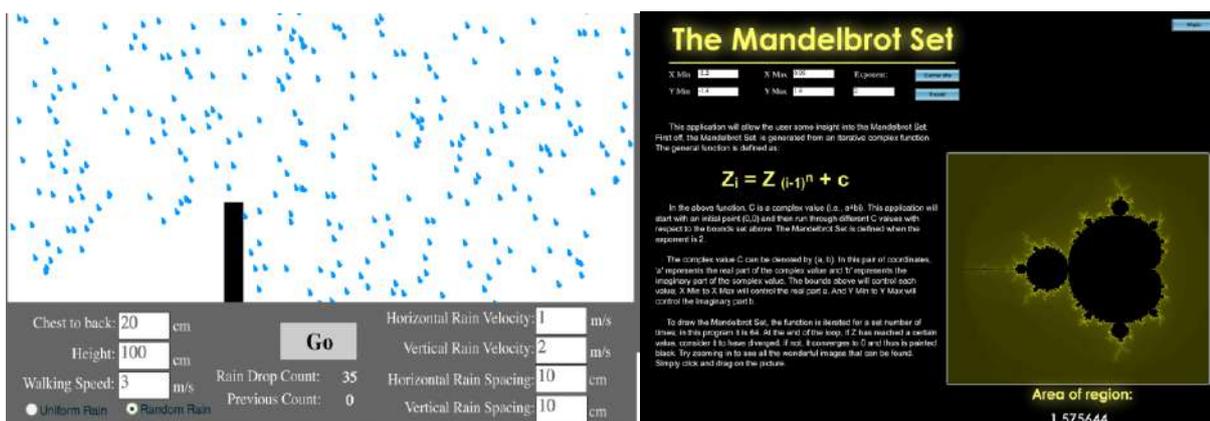


Figure 1. To the left, Matthew and Kylie’s real-world situation microworld: “is it better to walk or run in the rain?”; to the right, Adam’s pure mathematics microworld about the bounded area of the iterative complex function defining the Mandelbrot set as the exponent increases.

This individual project-based activity requires a completely different teaching format. Because students select the topic of their individual microworld projects, this is a demanding, ‘risky’ approach for the instructor, in particular in regards to the instructor’s curriculum script, resource system, and time economy features (Ruthven, 2009). The change in the instructor’s activity format and working environment is not challenging: the instructor acts ‘on-demand’ for students, and this individual guidance takes place in labs or at unscheduled time.

During the original project period, details of the instructor’s curriculum script could not have been elaborated ahead of time as it is driven by the students’ selection of topics. The instructor may need to

engage in research for some student projects: mostly to evaluate the feasibility of the project, which possibly involves a simplification of a proposed topic or problem and to guide students to adequately research the topic. The instructor may also be called upon to guide and to assist in the designing and running of the investigation. This is demanding as it is different and possibly new for each project and for each time the course is offered, and is part of the “at risk” aspect for the instructor. In terms of resource system, the instructor mostly relies on his/her research skills. This involves for each student project to help identify ‘on the spot’ good resources at the right level for students. As for instructor’s time economy, this part of the MICA courses is very much time demanding because of the individual guidance and research component required by the instructor for each student project (one of the reasons why MICA course sections are capped to 35 students). The evaluation of these projects is no less demanding.

Concluding Remarks

Throughout this practice report paper, we described how the teaching at Brock University of the MICA courses embraces a constructionist paradigm based on using and developing computational thinking for mathematics. By course design, MICA students learn by making through individual projects in the form of designing, programming, and using a mathematics microworld (i.e., a tangible computer object that is shareable) to learn and do mathematics. This is what we have called the ‘microworld approach’ in this paper. The student-centred constructionist learning experience in MICA courses culminate to individual original projects for which students select their own topics, thereby have the opportunity of it being meaningful to them. As such, the design aims to empower students to work as mathematicians. In terms of mathematics content, the microworld approach outlining the MICA courses has impacted the ‘what’ of mathematics (Buteau et al., 2016), a key characteristic distinguishing constructivism from constructionism (Noss & Clayson, 2015).

In order to reflect the design of the course, instructors have to change their pedagogy in a significant way, which highlights many characteristics of constructionist teaching. For example, MICA instructors tend to act more as facilitators than lecturers, and to use a slower pace to cover mathematics content. When appropriate, they make transparent the messy development of mathematical ideas. This is a shift from a traditional instructionist approach towards one of empowering students, thereby aligning with the aims of the courses (Buteau et al., 2015). In Buteau et al. (2016), we examined the learning experience of a student, named Ramona, through her three MICA courses. We concluded that “students such as Ramona engage in constructionist experiences of mathematics learning...,” adding that “[s]tudents also progressively develop proficiency in the third pillar of scientific inquiry mentioned by the European Mathematical Society (2011),” namely: “Together with theory and experimentation, a third pillar of scientific inquiry of complex systems has emerged in the form of a combination of modeling, simulation, optimization and visualization” (European Mathematical Society, 2011, p.2).

The present paper discussed the roles of and demands on the university instructors teaching the MICA courses. The application of Ruthven’s (2009) model on the professional adaptation of school classroom practice with technology proved to be very insightful as to identify key components of instructors’ roles and demands specific to the constructionist approach in the MICA courses. However, we suggest that, at the university level, a ‘Teaching Assistant’ structuring feature be added to Ruthven’s (2009) model.

There seem to be relatively few sustained implementations of microworlds in mathematics instruction (Healy & Kynigos, 2010). In this paper, we identified in the mathematics MICA courses, implemented at Brock University since 2001, that the ‘meaningful to students’ constructionist principle impacted, in a significant and challenging way, three of the five key structuring features of the instructor’s classroom practice as given in Ruthven’s (2009) model, namely curriculum script, resource system, and time economy. This could highlight why the implementation of a constructionist approach by (university) instructors is so challenging.

References

Barker, W.D., Bressoud, D., Epp, S., Ganter, S. et al. (Eds.), *Undergraduate programs and courses in the mathematical sciences: CUPM curriculum guide 2004*. Washington, DC: Mathematical Association

of America.

Broley, L., Buteau, C., & Muller, E. (2017, February). (Legitimate peripheral) computational thinking in mathematics. In T. Dooley & G. Gueudet (Eds.), *Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education (CERME10)*, (pp. 2515-2523). Dublin, Ireland: DCU Institute of Education & ERME.

Burns, R. B., & Anderson, L. W. (1987). The activity structure of lesson segments. *Curriculum Inquiry*, 17(1), 31-53.

Burns, R. B., & Lash, A. A. (1986). A comparison of activity structures during basic skills and problem-solving instruction in seventh-grade mathematics. *American Educational Research Journal*, 23(3), 393-414.

Buteau, C., Mgombelo, J., Muller, E., Rafiepour, A., & Sacristán, A. (submitted). Authentic Task Features for Computational Thinking in Mathematics. Submitted to Mathematics Education in Digital Age Conference, Copenhagen, September 2018.

Buteau, C. & E. Muller (2010): Student Development Process of Designing and Implementing Exploratory and Learning Objects. In V. Durand-Guerrier, S. Soury-Lavergne & F. Arzarello (Eds.), *Proceedings of the Sixth Congress of the European Society for Research in Mathematics Education (CERME 6)*, (pp. 1111-1120). Lyon, France: INRP and ERME.

Buteau, C., & Muller, E. (2014). Teaching Roles in a Technology Intensive Core Undergraduate Mathematics Course. In A. Clark-Wilson, O. Robutti, N. Sinclair (eds), *The Mathematics Teacher in the Digital Era* (pp. 163-185). Springer Netherlands.

Buteau, C., Muller, E., & Marshall, N. (2015). When a university mathematics department adopted core mathematics courses of an unintentionally constructionist nature: really? *Digital Experiences in Mathematics Education*, 1(2-3), 133-155. doi: 10.1007/s40751-015-0009-x

Buteau, C., Muller, E., Marshall, N., Sacristán, A. I., & Mgombelo, J. (2016). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Digital Experience in Mathematics Education*, 2(2), 142-156. doi:10.1007/s40751-016-0017-5

Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate mathematics program at Brock University. In Online Proceedings of Math+Coding Symposium, London, ON. <http://researchideas.ca/coding/proceedings.html>

diSessa, A.A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

Edwards, L. D. (1995). Microworlds as representations. In A.A. diSessa, C. Hoyles, R. Noss, & L. D. Edwards (Eds.), *Computers and exploratory learning* (pp. 127-154). New York: Springer.

European Mathematical Society (2011). Position paper of the European Mathematical Society on the European Commission's contributions to European research [online]. http://ec.europa.eu/research/csfr/pdf/contributions/post/european_organisations/european_mathematical_society.pdf. Accessed 20 Jul 2015.

Healy, L., & Kynigos, C. (2010). Charting the microworld territory over time: design and construction in mathematics education. *ZDM*, 42(1), 63-76.

Marshall, N. & C. Buteau (2014). Learning by designing and experimenting with interactive, dynamic mathematics exploratory objects. *International Journal for Technology in Mathematics Education*, 21 (2), 49-64.

MICA (2018). MICA – Mathematics Integrated with Computers and Applications. <https://brocku.ca/mathematics-science/mathematics/mica-mathematics-integrated-with-computers-and-applications/>

Muller, E., Buteau, C., Ralph, B., Mgombelo, J. (2009): Learning mathematics through the design and implementation of Exploratory and Learning Objects. *International Journal for Technology in*

Mathematics Education, 16 (2), 63-74.

Noss R. & Clayson J. (2015) Reconstructing constructionism. *Constructivist Foundations*, 10(3): 285–288. <http://constructivist.info/10/3/285>

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books, Inc.

Papert, S., & Harel, I. (1991). Situating constructionism. In I. Harel and S. Papert (Eds), *Constructionism*. NY: Ablex Publishing Corporation, 1-11. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>

Papert S (1996) An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1), 138–142. doi: 10.1007/BF00191473

Ralph, B. (2001). Mathematics takes an exciting new direction with MICA program. *Brock Teaching*, 1(1), 1. Retrieved from http://www.brocku.ca/webfm_send/18483.

Ralph, B. (2017). *Two-Hour Weekly Labs: Guidelines*. Retrieved from: <http://ctuniversitymath.ca/2017/11/29/introductory-course-lab-guidelines/>

Rasmussen, C., & Kwon, O. N. (2007). An inquiry-oriented approach to undergraduate mathematics. *The Journal of Mathematical Behavior*, 26(3), 189-194.

Ruthven, K. (2009). Towards a naturalistic conceptualisation of technology integration in classroom practice: The example of school mathematics. *Education & didactique*, 3(1), 131-159.

Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments: guiding students' command process through instrumental orchestrations. *International Journal of Computers for Mathematical Learning*, 9, 281-307.

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*, 54(5), 615-641.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal for Science Education and Technology*, 25, 127-147.

Design Curriculum for Educational Robotics: Constructionist Pedagogical Experience in Formal Education

Flavio Campos, *flavio.rcampos@sp.senac.br*
University of Sao Paulo – USP, Brazil

Abstract

This paper presents a pedagogical constructionist experience in designing curriculum for educational robotics for k-12 formal education. The discussion is based on 15 years' experience of a school in Brazil, detailing a timeline of an educational robotics curriculum project development. Beginning as an afterschool program back in 2003, running as a technological resource used by math and science teachers and later integrated into curriculum from kindergarten through high school. The analyses emphasize the historical scale of a constructionist curriculum project with robotics, describing the aspects of each step and the challenges faced by the school, teachers and the administration. Therefore, the paper intends to address a curriculum design approach that drives educational robotics activities, highlighting the constructionism perspective. Pedagogy and its dimensions such as methodology, teaching and learning related to educational robotics are presented within theoretical and educational foundations. Beyond the use of robotics, deepens the discussion of curriculum design, engineering design process during robotics activities and how the integration of robotics education can boost constructionist approach in formal education.

Keywords

design curriculum; educational robotics; constructionism; engineering design process.

Chapter 1

Introduction

In the last decade, robotics has sharpened the interest of teachers and researchers as an important resource for cognitive development and social skills of students from kindergarten to high school and in the learning background of science, mathematics, technology, computing and other knowledge.

As a technological resource used by educators, robotics is one of the most upgraded and integrated into school's activities around the world. From STEM projects through political pedagogical standards, formal education is allowing computational thinking and constructionist approach to emerged in teaching and learning process using robotics and its technology. Many countries, including Brazil, have been benefited by a growing number of educational robotics technology such as LEGO Mindstorms, fischertechnik, Kibo, K'nex, GoGo board and others.

Robotics projects in k-12 education in many schools represent as an isolated practice in different development projects, because these projects have been identified as a specific subject in the curriculum, which means it has been using in professional education in high school or College. Robotics have been seen by educators and the population as a sophisticated toy, in which people that love robotics find themselves in championships and conferences around the world.

The research in robotics has been reaching the university context – engineering and mechanics – and industries. The interest for the subject is growing, and we can see the investments from the government in educational technology. Even with all the investments, only a few k12 schools in Brazil integrate educational technology subjects (such as robotics) in curriculum. Projects that are more significant are limited in professional education and College.

Despite all that, It's not unusual to find educators interested in exploring robotics, STEM curriculum and constructionist concepts in their practice. Influenced by researchers and projects using robotics in

schools, by cinema and media, or by simply amused by technology, teachers and students mobilize themselves to create and conduct their projects. Making the design, build, program and analyse the results of robotics become a motivated activity in learning process and STEM curriculum, helping cognitive process, as well as provides creative activities.

Phase 1 – Afterschool program and science fairs

Beginning in 2003, the school acquired initially knex and LEGO Mindstorms(RCX) educational sets to start its afterschool program. The goal was to develop a program to give students an opportunity to learn and engage in engineering and robotics activities.

The activities were based on engineering and robotics concepts and were developed considering students from 6th grade to high school (11 to 17 years old). Once a week for three hours, students interested in those activities participated in this program offered only after school and they could design artifacts with either knex or mindstorms.

Usually, the projects were presented by the teacher, with challenges prepared based on science concepts and mathematical problems. One aspect of this approach was that only students with a personal interest in robotics and engineering looked for to participate, which represented less than 5% of total students at the school.

Although a small percentage of the students participated at the beginning, it was significant to create a motivational atmosphere and to initiate a discussion about robotics integration into curriculum activities.

During the afterschool program phase, students started to talk more about different possibilities with this technology in the classroom, asking teachers and the administration about using robotics sets to learn different types of subjects such as science, math and even humanities.

After only two years of afterschool programs, the administration decided to add another step into educational robotics and constructionism perspective allowing science and math teachers to integrate educational robotics in science fairs projects.

The 2005 science fair integrated robotics with 9th grade and high school students. Teachers decided to use robotics with a group of 35 students, divided into small groups of 5 and let them with the choice of their own projects. Therefore, students worked for two months in their projects, designing, building and programming devices to present at the science fair.



Figure 2. Knex Eiffel Tower project and RCX spider project – (Science fair 2005)

First, students gathered to decide a theme to work on, and then they structured a schedule to develop the project. Research and prototype design were the main tasks during the project development, and students had time between school program and afterschool program to finish the project.

The teachers were mentors during the process, helping students in everything they need, from scientific concepts to engineering design. At the science fair, students had to present their project to the community, demonstrating not only the “product” but all the process.

By that time, robotics were only used at the afterschool program and during the science fair schedule.

Phase 2 – Mathematics and Science curriculum with educational robotics

The first science fair with robotics sparked a desire to use educational robotics beyond afterschool programs and science fairs. Teachers and school administration decided to use robotics in math and science classes.

In 2005, science and math 9th grade teachers specifically started to plan their classes and projects. The expectations at the beginning were to create room for more and more activities with robotics, letting teachers of science and math experience the learning process with robotics.

One of the math classroom projects designed was a car created by the students to study equations, which students engineered a door to calculate the data. The teacher planned the activity and students had the chance to create the object, although it had to be a car.



Figure 3. Knex car project equations – (Math classroom activity 2006)

The scenario in 2005 was different from what the administration expected. While students were excited to use robotics to learn school content such as equations, simple machines and so forth, teachers actually only used educational robotics in six or seven classroom activities during the school year.

Teachers had a professional development to comprehend theory and practice of educational robotics, as well as time to prepare and discuss possibilities of curriculum projects with robotics. Unfortunately, that seemed not to be enough, as students had just a few opportunities to experience a constructionist approach to robotics.

Phase 3 – Participation First LEGO League competition

After two years of educational robotics, the school administration invited some of the students that had been participating in the afterschool program to be at the First Lego League competition. They had group meetings every week (2 hours) and dedicated time to deep understanding of programming and engineering, using at first Lego mindstorms RCX.

The project was not only to participate in the competition but also to present the challenges to other students at the school and even promote a “school competition” in order to engage students with educational robotics. Therefore, during 2006 competition, they had a waiting list of students wanting to participate, and after three seasons, the school usually takes 3-4 to teams to the competition.

This experience was significant because, with only two years, teachers and administration motivated with students participation decided to think about integrating educational robotics and constructionist approach into curriculum.

Phase 4 – Educational robotics integrated into curriculum

The year of 2008 represented an important change in educational robotics at the school. Teachers and administrators decided to include a subject in the curriculum called “Engineering and Robotics”, with classes once a week for an hour and forty minutes. Such as math, science, history, geography and so forth, this subject was integrated in the curriculum from 1st grade to high school students.

Since 2008, educational robotics and constructionist approach have been part of the curriculum, with weekly activities using different resources (Lego mindstorms, knex, GoGo board, Arduino), programming with a diverse type of programming languages (robolab, nxt, Labview, scratch).

At the beginning, teachers organized classroom activities based on science and math concepts and used educational robotics to achieve the goals, as an example, an activity was to calculate distance, time and speed using a car built with Lego mindstorms.

For two years, teachers were planning activities using educational robotics to understand some subject concepts, but not too deeply engage into technology and robotics itself. The initial experience was to use robotics as a resource, helping teachers to present complex concepts.



Figure 4. Engineering and Robotics curriculum development with NXT and makey makey board– (from left to right: classroom activity 2011 and 2016)

Combining the experience with the desire to expand robotics activities, teachers decided to integrate robotics and technology with complex concepts in different subjects such as science, math and humanities.

Thereby, educational robotics classroom activities started to be planned based on computing, robotics and subjects content. At the same time, teachers prepared a storytelling approach with 1st to 5th grade to integrate educational robotics.



Figure 5. Storytelling, from left to right – (Story with a challenge; an engineering design and Keywords to study during the project)

After reading a story, which contains a challenge, students always in groups of three or four start to plan the solution, picking up an engineering design prototype to build. Actually, students begin the activity a week before, investigating the problem and keywords teachers give them in order to maximize the time between every project.

An example of this storytelling approach is the activity called “submarine” with students in 5th grade in 2017. After students had read the story with two characters talking about submarines, they did research about it; then, students designed their prototypes and built the submarine based on engineering design

process and programmed it to do what the students planned. At the end, students showed their projects to the class and shared ideas and choices, registering the most important parts of the physical construction and how the mechanism created worked. A quote from the students about the most important parts of the submarine created and how it works:

“The motor, engineering basis, gears and propellers. The motor makes the propellers spin and the submarine moves”. Student (2017)

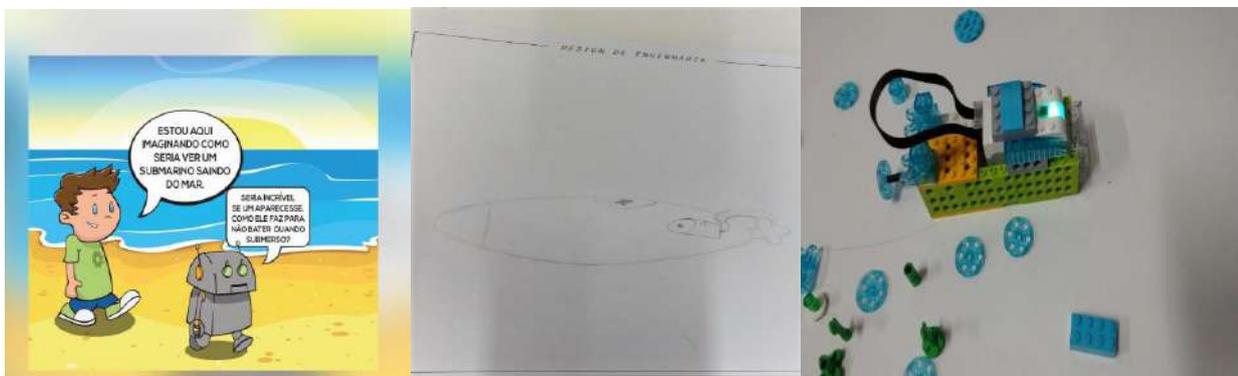


Figure 6. Submarine activity (novel, design’s prototype and the submarine model built – 5th grade)

Every project/activity has keywords related to four dimensions: technology; science; vocabulary and engineering to guide the teaching and learning process. For example, in the submarine activity these are the keywords:

Technology	Science	Vocabulary	Engineering
Relation between structures and motors	Experiment and relation to motors, design and engineering;	Motors, structures, design, machines;	Description and explanation of construction;
Assembling components	Simple machines;		Test and evaluation;
Evaluation;	Scientific investigation;		Engineering design;

Table 1. Keywords dimensions for educational robotics project/activity

Chapter 2

Design curriculum for educational robotics – A curriculum model approach

We consider the perspective of robotics in the curriculum in a broader way and its integration into the curriculum permeates both the curricular framework itself and afterschool projects. However, we prioritize the integration of robotics in the curriculum, as the afterschool projects have specific characteristics, allowing greater flexibility in the development of the projects.

Relevant element of this aspect lies in the fact that schools, in general, do not have the “clear” direction about what to teach related to robotics, for instance, they usually have difficult to choose related any robotics content and thus take different paths in relation to the choice of didactic materials and contents for this component.

Indeed, this is fundamental when we discuss about integrating robotics into the curriculum. Differently from subjects such as mathematics, science, geography, historically constituted as the school core curriculum with structured contents for each school year(educational policies), which contribute to a diversity of didactic-pedagogical contents.

For instance, we can think of schools that are planning curriculum of robotics based on the concepts of technology linked directly with the materials of this resource, such as learning programming and the use of sensors and motors. In this case, schools demonstrate difficulties in aggregating contents from different subjects that constitute the school core curriculum.

Another example are schools that privilege content linked to subjects such as science, mathematics, physics, which limits the concepts of technology learning.

Therefore, to think of robotics integration in the curriculum is indispensable to considerate the perspective of three axes: science, technology and subjects, according to the model we propose below:

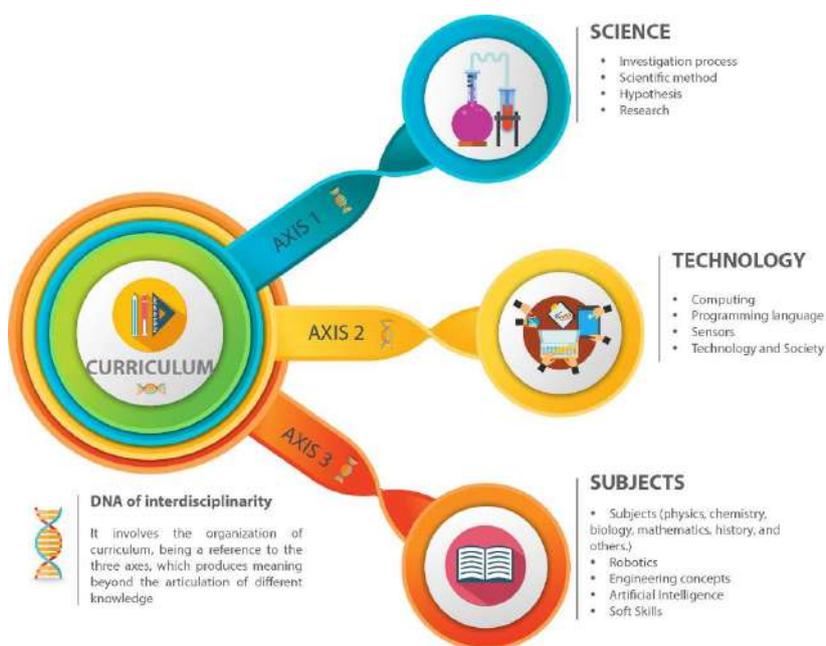


Figure 7. Educational Robotics curriculum model

These aspects provide guidance to the design of robotics curriculum in formal education environment and their integration in a meaningful way, having as reference the construction of knowledge and the autonomy of students in the teaching-learning process.

In **science**, elements such as investigation process, research, hypothesis, scientific method, among others are central. This axis contributes to a curriculum guided by the immersion of the student in a process of investigation of the studied phenomena, in research and tests of hypotheses.

About **technology**, we need to consider the knowledge of functioning parts such as sensors, motors, electronics, programming language, the field of computation itself, computational thinking, and advances in technology (as a technological artifact).

Finally, **subjects**, which contemplate the school core curriculum (Physics, Language, History, Sciences, and Mathematics), those referring to robotics, engineering, artificial intelligence, creativity, as well as soft skills such as teamwork, collaborative learning, among others.

The organization of these three axes is what we call the "DNA" of interdisciplinarity. In this sense, it refers to the whole process of knowledge production, constituting itself not as the simple inter-relation between knowledge, but the concrete constitution of all knowledge produced in the activity or project.

Interdisciplinarity is the production of meaning in the whole process, and not only the "mixing" of areas of knowledge around a theme/project, in other words, it enlarges the interconnections and produces

knowledge that did not exist before, involving aspects of philosophy, anthropology and sociology. (Fazenda, 2010)

Therefore, we propose this model for educational robotics curriculum design, in which for every project or activity the three axes must be present as a core curriculum, with interdisciplinarity as a DNA considering a link to the three axes and balancing the importance of content knowledge related to the teaching and learning process.

Considering this context, robotics curriculum can contribute to an emancipatory teaching-learning process for both student and teacher. These aspects usually encounter challenges in order to actually materialize during day-to-day of school education, given the relation time/space, available resources, teacher training, among others.

Integrating robotics is fundamental, because it goes beyond a technological resource that allows the active participation of students in constructing knowledge. It has the potential to contribute to the development of projects that aim the emancipation of students in learning complex concepts and skills development of the 21st century.

In addition, it contributes not only to the construction of a multi-referenced curriculum, which considers both the historically constituted core contents and the particular contexts of each school for the development of pedagogical projects, but also for the strengthening of a culture of technology use in education that has as fundamentals the autonomy and emancipation of students in the teaching-learning process.

It is not, therefore, simply to add robotics in the curriculum framework because it is interesting, to "conquer" new students, nor to use this technological resource at a few times during the school year.

Creativity in the context of integrating robotics in the curriculum stands out as another fundamental element. Integration projects of such technology that contemplate activities that do not allow students to create in all steps described are limited to only superficially incorporate robotics.

Thus, during the stages of a robotic activity, students need to exert creativity, for example, they cannot receive ready-made assembly model and instead they need to build the device based on the challenge proposed at the beginning of the activity. They need to create the device's programming and not receive it ready to just test it.

In this sense, creativity must permeate the learner's action during all stages of a robotic activity, in order to maximize the reach of this technological resource in the teaching-learning process and, therefore, ensure the integration of robotics in the curriculum significantly.

References

- Blikstein, P. (2013) Digital fabrication and "making" in education: The democratization of invention. In J. Walter- Herrmann & C. Bóching (eds.). *FabLabs: Of Machines, Makers and Inventors* (pp. 1-21). Bielefeld: Transcript Publishers, 2013.
- Fazenda, I. C. A. (2010) *Interdisciplinaridade*. São Paulo: Papirus.
- Papanikolaou, K, Frangou, S., Alimisis, D. (2008) Teachers as designers of robotics-enhanced projects: the TERECoP course in Greece. In *SIMPAR*. 1, 2008, Itália. Veneza, volume 6472, 556 p.
- Papert, S. (1980) *Mindstorms: Computers, Children and Powerful Ideas*. NY: Basic Books.
- Papert, S. (1987) Computer criticism vs. technocentric thinking. *Educational Researcher*, 16(1), 22-30.
- Piaget, J. (1974) *To understand is to invent*. N.Y.: Basic Books.
- Resnick, M., Berg, R.; Eisenberg, M. (2000) Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation. *Journal of the Learning Sciences*, 9(1), 7-30.
- Resnick, M. (2007) Sowing the seeds for a more creative society. *Learning & Leading with Technology*, 35(4), 18-22.
- Ortiz, J., Bustos, R., Rios, A. (2011) System of indicators and methodology of evaluation for the robotics in classroom. *Proceedings of the 2nd International Conference on Robotics in Education (RiE 2011)* (pp. 63-70). Vienna, Austria: Austrian Society for Innovative Computer Sciences. http://www.innoc.at/fileadmin/user_upload/_temp_/RiE/Proceedings/37.pdf, 2011.

Forming Concepts for Programming Conditional Statements in the Primary School

Miroslava Černochová, miroslava.cernochova@pedf.cuni.cz

Radek Čuma, radek.cuma@seznam.cz

Hasan Selcuk, hasan.selcuk@pedf.cuni.cz

Faculty of Education, Charles University, Prague, Czech Republic

Abstract

The Ministry of Education, Youth and Sport of the Czech Republic is currently preparing an update of curriculum documents for primary, lower and upper secondary school education in which two major changes to be made: (1) A concept of **digital literacy** will be incorporated into all school subject across the curriculum in accordance with DigComp 2.0 defined by JRC EC; (2) Instead of the existing compulsory subject ICT, **a new compulsory subject of Informatics** focused on computational thinking development, will be introduced in all levels of education. In the context of the forthcoming curricular changes, not only educational activities, but also research on the way in which pupils of different ages acquire basic information concepts has great importance.

The authors conducted a case study focussed on discovering how pupils of primary school (especially Year 3, 4 and 5) acquire, use and understand some programming conditional statements and loops (IF-THEN, IF-THEN-ELSE; REPEAT/ REPEAT-UNTIL). Programming conditional statements are undoubtedly one of the fundamental algorithmic concepts that pupils will need to understand and use in programming and algorithmic thinking development. How can pupils apply them in programming? Does it make sense to introduce these programming conditional statements into programming activities in primary education?

The research was carried out in 2017/18 at a small village school among 31 pupils (17 girls and 14 boys) of Year 3, 4 and Year 5 during 16 lessons of a compulsory subject "Work with a computer". Pupils usually worked in groups of three or four. The activities were designed in accordance with a proposal of requirements for algorithm skills and programming development in primary school education. The research was organised into four phases: (i) Preparatory phase (out of school), (ii) CSunplugged activities with a special set of paper cards and LEGO toys, (iii) Activities in a virtual environment Code.org, and (iv) Testing acquired skills and knowledge.

Findings showed that firstly, primary school pupils are able to use programming conditional statements and loops, nevertheless the Year 3 pupils can lose motivation and willingness to work if something is wrong or if they are not successful, and also they can have some linguistic barriers how to describe more details verbally their algorithmic schemes. Secondly, it has a sense to introduce these conditional statements and loops into primary education if we create for pupils conditions to link their concrete ideas based on manual operations (such as with a Lego toy) to their experiences gained in a virtual programming environment (such as Code.org).

Keywords

conditional statements; loops; algorithm; programming; computational thinking; unplugged activities; code.org

Introduction

The Ministry of Education, Youth and Sport of the Czech Republic is currently preparing an update of the curriculum document Framework Educational Programme (FEP), see (MoYES, 2013). In the new FEP for elementary/primary schools (for pupils aged 6-15), two major changes relating to the digital education strategy (MoEYS, 2014) will be made. (1) Pupil's **digital literacy**, which was formed through compulsory ICT, will be now developed in all subjects. The concept of digital literacy will be incorporated into the curriculum in accordance with its form defined in DigComp 2.0 described in R. Vuorikari et al.

(2016). (2) In the curriculum for primary schools, instead of ICT a **new compulsory subject of Informatics, which will focus on the development of information thinking**, will be introduced. Informatics will be included in both primary and lower secondary education.

The changes awaiting primary schools are relatively radical and should be put into practice by 2021.

Research aim

Since 2017, nine Czech faculties of education have been working closely together to develop and validate teaching materials, methodological guidelines for teaching a new subject of Informatics and for validating these at several selected schools (starting with pre-school centres and kindergartens and ending with secondary schools). At the same time, courses and subjects for teachers of kindergartens, primary and secondary schools are being prepared to be ready for the planned curricular changes. All nine faculties of education innovate study programmes for student teachers of all subjects including ICT and Computer Science.

In the context of the forthcoming curricular changes, not only educational activities, but also research on the way in which pupils of different ages acquire basic information concepts has a great importance.

The research, we have done, is focussed on discovering how pupils of primary school (especially Year 3, 4 and 5) acquire, use and understand when designing programs involving selected commands and functions associated with some programming conditional statements (IF-THEN, IF-THEN-ELSE; REPEAT/ REPEAT-UNTIL). Programming conditional statements are undoubtedly one of the fundamental algorithmic concepts that pupils will need to learn and use in programming and algorithmic thinking development. How do they understand these concepts? How can they apply them in programming? Does it make sense to introduce these programming conditional statements into programming activities in primary education?

In our case study research, we decided to design and test a pedagogical experiment with primary school pupils a methodical approach to teaching programming with programming conditional statement like IF-THEN, IF-THEN-ELSE, REPEAT, REPEAT-UNTIL which can lead to easier understanding and acquiring algorithms of this type of task and learning activity.

Theoretical framework

The activities for pupils were prepared in accordance with a draft document of requirements for skills and knowledge for a computational thinking (NÚV, 2017), especially for algorithm and programming development in primary school education.

The content here (Table 1) was drawn upon in preparation for the research.

Table 1. A proposal of standards and continuity in algorithm and programming development (NÚV, 2017)

Algorithm development and programming	A pupil is able to read and interpret text or symbolic scripts of the algorithm and to explain their individual steps.
	A pupil is able to describe a simple problem, to design and to explain individual steps of its solution.
	A pupil is able to adjust and modify a ready procedure for a similar problem. S/he is able to verify a correctness of a proposed procedure and to find and debug errors.
	A pupil is able to recognise if two different algorithms can solve the same problem.
	In a block-oriented programming language, a pupil is able to design a computer program. S/he is able to test it and debug errors in it.

	A pupil is able to recognize recurring patterns, to use cycles for repeating and to apply sub-programs (sub-routines). S/he is able to use events to run sub-routines and scripts.
--	--

Methodological design

The research was carried out in 2017/18 at a small village school among pupils of years 3, 4 and 5 during 16 lessons of a compulsory subject “Work with a computer”. In the research there were N1 = 15 pupils (8 girls and 7 boys) of Year 3 and Year 5 and N2 = 16 pupils (9 girls and 7 boys) from Year 4. Pupils usually worked in groups of four.

Tasks were designed to develop the skills to compile algorithm development with using hand-made and ready-made teaching tools for CS unplugged activities.

The case study was organised in these phases (Table 2):

Table 2. Phases of the case study

<i>Preparatory phase</i>	weekend playing and games with several pupils (outside school) managed by a teacher - researcher
<i>CS unplugged activities</i>	Activities without a computer in (8) groups with 3-4 pupils. The two groups consisted of Year 3 pupils, four groups were Year 4 pupils and two groups of pupils from Year 5. Pupils were divided into mixed groups so that in each group there were girls and boys and pupils with different study success (marks, grades). For Csunplugged activities there were developed a set of cards from which pupils could compile algorithms and learn to read procedures and interpret and perform them (using a vehicle constructed in Lego).
<i>Activities in a virtual environment</i>	Tasks in a virtual Code-Oriented Programming Environment Code.org were solved by pupils individually in a 1: 1 model, each pupil worked on one device. The pupils' results were recorded in the online environment and further analyzed.
<i>Testing acquired skills and knowledge</i>	Control tasks for pupils were assigned with the aim to find out to what extent pupils understood the meaning and principle of using commands with conditional statement (Repeat X-times, Repeat Until, IF - THEN, IF – THEN – ELSE)

Research data were collected using video and audio records by means of: (1) observation of pupils' behaviour and activities; (2) records of processes how pupils solved problems and tasks; (3) testing how pupils understand selected algorithmic concepts. Pupils were very often asked (i) to describe and interpret how they proceeded in solving a problem, (ii) to read the codes, and (iii) to interpret meaning of the assembled algorithms (using, for example, a vehicle made in Lego). Collected data were added to data gained using the environment Code.org with the aim to analyse a progress in pupils' thinking.

Expected conclusions/findings

The data are currently being analysed. The results will be available in June 2018. The analysis of the collected data should contribute to understanding how students comprehend their programming, how and if they are able to find errors in their work.

Experience shows that primary school pupils are quick to lose motivation and willingness to work if something is wrong or if they are not successful. The same goes for programming. If teaching programming is not responsive to pupils' capability, a lot of pupils can be discouraged. By creating a bridge between a real and virtual world of programming using unplugged activities, we can achieve work in a virtual environment which will not be difficult for pupils.

In our case study, the pupils demonstrated that they correctly understand the meaning and principle of using an algorithm when programming conditional statements. The pupils also showed that they can comprehend what events occur at each step of the program, to which the said unplugged activities with a set of cards greatly contributed, especially in activities where the students manually demonstrated an assembled algorithm using vehicles made from Lego blocks.

The pupils subsequently transferred these learned skills into tasks in Code.org which led to the desired link between unplugged activities and activities in a virtual programming environment.

Resources

Futcek, G., Moschitz, J. (2011) Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming. In: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. ISSEP 2011: Informatics in Schools. Contributing to 21st Century Education. Springer, 2011, pp 155-164. Available at https://publik.tuwien.ac.at/files/PubDat_199953.pdf

MoEYS (2013) Rámcový vzdělávací program pro základní vzdělávání. Praha: Výzkumný ústav pedagogický v Praze, 2013. Available at: <http://www.msmt.cz/file/43792/>

MoEYS (2014) Strategie digitálního vzdělávání do roku 2020. Available at: <http://www.msmt.cz/uploads/DigiStrategie.pdf>

NÚV (2017) Informatika - rámec očekávaných výstupů (prosinec 2017). Tabulka pro posouzení návaznosti. Available at: https://docs.google.com/spreadsheets/d/1op92O_ZFNcLRbKxm6FcanUmPjSdP_Dd2JAznlfG5YDU/edit#gid=1456952308

Vuorikari, R., Pune, Y., Crretero, S., Van Den Brade, L. (2016) DigComp 2.0: The Digital Competence Framework for Citizens. Update Phase 1: The Conceptual Reference Model. European Union, 2016. ISBN 978-92-79-58876-1.

Designing Constructionist Learning Environments with Computational Design and Digital Fabrication

Christos Chytas, *christos.chytas@uni-oldenburg.de*

Computing Education Research Group, University of Oldenburg, Germany

Ira Diethelm, *ira.diethelm@uni-oldenburg.de*

Computing Education Research Group, University of Oldenburg, Germany

Abstract

Makerspaces like fab labs (digital fabrication laboratories) are open workshops that promise to democratize the means of production and technical knowledge. Even though such laboratories are often seen as innovation spaces for small business they are receiving increasing attention as informal learning environments for STEAM (Science, Technology, Engineering, Art, Mathematics) subjects. "Making" as a set of learning activities roots in learning theories of educators like Seymour Papert. Inspired by Papert's constructionist learning theory, we designed, implemented and evaluated workshops on digital fabrication and computational design for children and youth. The workshops' goal was to introduce computational concepts and programming as means of personal expression through the creation of computational design models that could be fabricated in our labs.

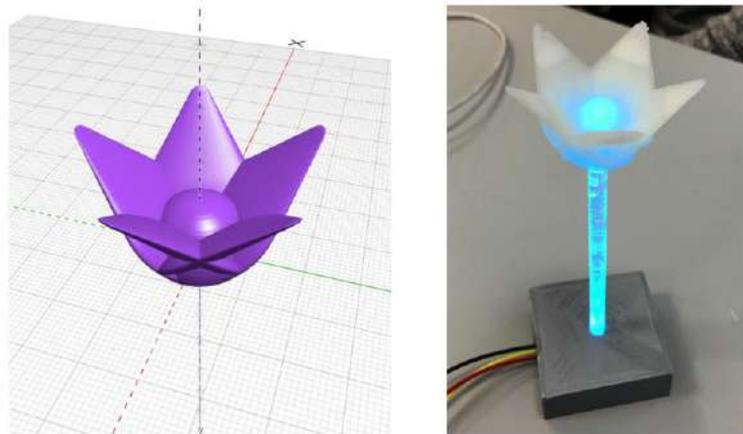


Figure 1. Generated 3D parametric model of a flower which was 3D printed to make an interactive artistic project

The workshops' concept provided opportunities for creativity, personal expression, collaboration and content rich learning activities to create artistic, practical or entertaining artifacts. We identify the elements of maker tools and culture that enhanced the learning experience in our workshops. We discuss implications and challenges of these elements for educators who wish to use digital fabrication for programming learning activities.

Keywords

constructionism; computational design; digital fabrication; the maker movement; computing education; Fab Lab

Introduction and Theoretical Background

The Maker Movement is on the rise and what started as a hobbyist community for tinkering and crafts has now expanded to reach millions of people in a growing number of physical and online spaces. Making activities often include crafts and the use of sophisticated technologies like digital fabrication machines (3D printers, laser and vinyl cutters, CNC routers etc.), CAD (Computer Aided Design)

software and microcontrollers. These tools are becoming increasingly accessible for everyone to use (Anderson, 2012). The expiration of patents in 3D printing and the decreasing price of other digital fabricators like laser cutters have triggered the rise of open-source software for 3D modelling (e.g. OpenSCAD), vector graphics (e.g. Inkscape) and image processing (e.g. GIMP) among others. Furthermore, online makerspaces dedicated to open-source hardware like Thingiverse.com provide digital models that can be shared under the creative commons licenses. Such models can also be created through computational/parametric design tools and be customized by users of diverse experience in design.

By focusing on design and construction, making provides exciting opportunities to explore STEAM subjects, including engineering design (Blikstein, 2013a), programming through physical computing with the use of microcontrollers and microcomputers to make interactive artifacts (Blikstein, 2013b) or computational design (Jacobs and Buechley, 2013; Dittert et al., 2014; Chytas et al., 2018). Such activities attract increasing attention from educators who strive to support the development of engineering and computing skills in a meaningful and interesting way to foster 21st century skills. According to Vossoughi and Bevan (2015), making settings *“are generally to inspire interest, foster engagement, develop understanding of the processes and concepts at the center of making activities, and support students’ identities as thinkers, creators and producers of knowledge”*.

Literature review on the maker movement in education (Blikstein, 2013a; Vossoughi and Bevan, 2015) shows that making inherits ideas of educators like Dewey, Froebel, Montessori, Vygotsky and especially those of Piaget and Papert. Seymour Papert inspired by Piaget’s Constructivism developed Constructionism, a learning theory that emphasizes on learning by constructing mental models and that the learning experience can be further enhanced by creating something tangible that can be shared with others. *“Constructionism--the N word as opposed to the V word—shares constructivism’s connotation of learning as “building knowledge structures “irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it’s a sandcastle on the beach or a theory of the universe”* (Papert and Harel, 1991).

Few decades ago, Papert and his colleagues developed Logo (Papert, 1980), a programming environment to introduce aspects of mathematics and computer science to children. In computing education programming is a major activity that is important for everyone to learn. Even more essential is the development of computational thinking skills and digital literacy which also require at least basic knowledge of algorithms and computational concepts. Logo was aligned with the theory of constructionism and provided exciting opportunities for children and youth to engage in programming (and math) activities.

Advancements in technology and the accessibility of digital fabrication tools allow us to create digital designs and turn them into physical artifacts faster and cheaper than ever. Eisenberg suggests that a maker-centred CSE would *“situate computers as elements in a creative technological landscape that includes 3D printers and scanners, a growing selection of sensors and actuators”*. According to him, combining elements of CSE with making culture would expand its focus, not only on software and digital realities but physical artifacts and hardware as well (Eisenberg, 2017).

Moreover, computation has expanded to reach industrial design and architecture, opening new possibilities for creative computing in the physical realm. Computational design has a long history in computing education since the popularization of Logo (Papert, 1980) and other educational tools like Turtle Art. Even though there is a plethora of promising tools that promise more creativity in computing education, we still believe that there is a gap in current practices with digital fabrication and computational design in constructionist learning environments.

Meanwhile, educators and academics have called for best practices to take advantage of the tools of making in the classroom. Even though digital fabrication in the realm of computing education is receiving increasing attention from the constructionist community, the findings on computational design have not progressed as much as in physical computing with the use of microcontrollers. Furthermore, there is an ongoing trend on the use of digital fabrication and especially 3D printing in education but the potential of such technologies for programming learning activities is still something new for most educators.

Chytas et al. (2018) focused on the use of computational design and digital fabrication for computationally rich learning activities and their potential to develop a programmer's mindset by creating personally meaningful artifacts. Dittert et al. (2014) and Jacobs and Buechley (2013) focus on the use of the *Processing* programming language to generate 2D shapes that can be manufactured by laser cutters and its impact on the empowerment of workshop participants to use programming for creative purposes. Kastl et al. (2017) focused on the use of 3D modelling through turtle graphics to create 3D models that could be fabricated by 3D printers, as motivation to support programming and mathematics actions in educational settings.

Methodology

To investigate best practices for coding learning activities with digital fabrication, we follow a mixed methods approach. Similar with the method of Katterfeld et al. (2014), our approach includes *iterative circles of design and research* on digital fabrication in educational context in order to improve our current workshops. The evaluation of our workshops is based on before and after surveys, observations, evaluations of artifacts and interviews with the participants. This evaluation helps us to better understand their expectations, profound experience, self-efficacy on the use of technology, their wishes as well as what they liked or did not like during the workshops.

From 2016 until now, we evaluated workshops with more than 50 participants aged from ten to 17 years old in formal and informal learning spaces. All these spaces were equipped with 3D printers and had close access to a laser cutter as well. The number of participants in our workshops varied from small groups of two to a class of 16 students. The workshops' duration is usually between three to five days so that the students have enough time to explore design and programming practices in depth and build physical artifacts of their designs through digital fabrication technologies. Our educational setting follows a constructionist approach, meaning that the participants are encouraged to familiarize with the technology by themselves and learn-by-doing. The workshops are usually supported by two to four tutors and researchers who act as facilitators. The implementation of the workshops included 1) demonstration of digital and physical artifacts, 2) description of the values and ideas behind the maker movement and making, 3) a short introduction of computational/parametric design (CAD and programming features), 4) the "making" phase, 5) the demonstration of the produced artifacts and 6) the reflection on the experiences from the workshop.

To introduce programming under the lens of computational design we use two parametric design tools: a) BlocksCAD (an online platform which exploits block-based parametric design tools) and b) OpenSCAD (an open-source parametric design software which includes a syntax similar with the C programming language). An example of code from both tools and the generated 3D models are illustrated in Figure 2 and 3 respectively. These tools share two important similarities which were the determining factor to use this combination in our workshops. Both tools are free to use and share the same essential commands for parametric design which fall under four main categories: 1) *Shapes*, 2) *Transformations* (e.g. commands to move or scale the generated models), 3) *Constructive Solid Geometry Operations* (e.g. commands to merge two models together or subtract one set of existing 3D solids from another) and 4) *Programming Features* (e.g. iterations, conditional statements, functions and the use of parameters to change the properties of a model). The first three categories of commands are broadly used in most hobbyist and professional CAD tools while the last one adds computational elements like iterations and customization features to the design.

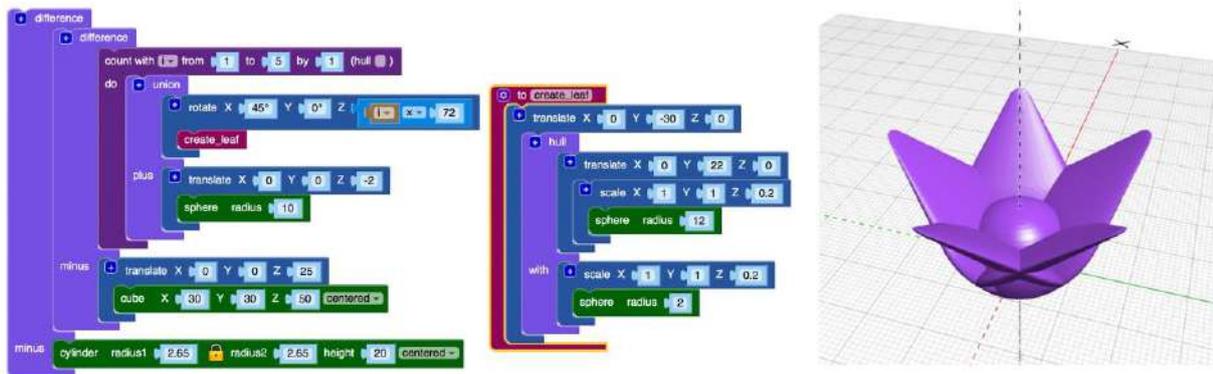


Figure 2. Generated 3D parametric model of a flower through blocks. The code of the workshop’s participant includes modules (which group parts of code for future reuse), iterations, conditionals and core CAD elements (shapes, transformations and constructive solid geometry operations).

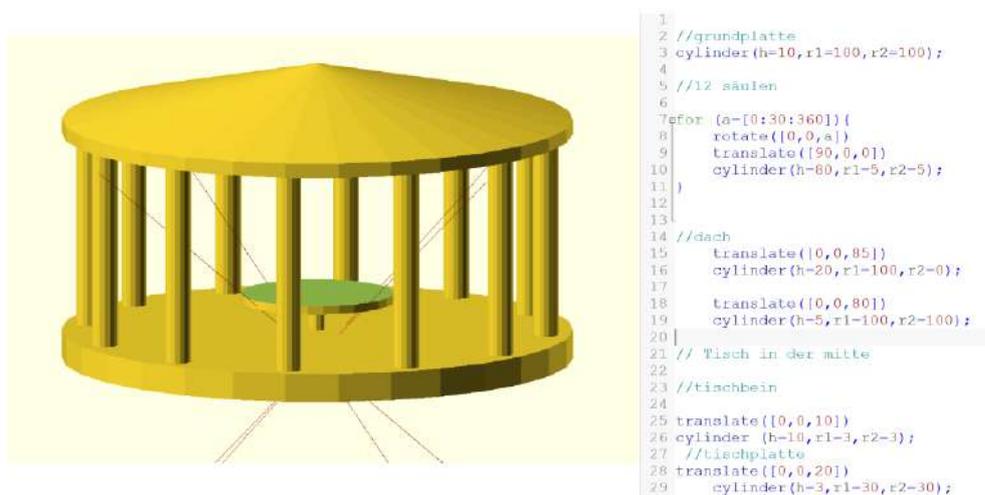


Figure 3. Generated 3D parametric model of a building through a text editor which includes programming features and CAD elements.

In the following sections we provide elements of computational design and maker culture that we found to be suitable to enhance personal expression in programming activities.

Computational Design and Constructionist Learning

In maker culture, technical skills like coding are often used for personal expression through the creation of physical projects that are linked with social, educational and entertaining activities. Within the constructionist community, there is a rich discussion on the benefits of making tangible media that can be enjoyed and shared with others (Blikstein, 2013). The tools of making like open-source 3D printers and programming CAD tools (e.g. parametric design software) promise to bring creative computational activities in the physical realm. However, according to Katterfeld et al. (2015), current research on digital fabrication in educational context should focus not only in developing computing skills and competences, but also nurturing deep and sustainable learning about the medium through personal development. After ten years of implementing digital fabrication workshops, the authors highlight three ideas as crucial in achieving deep sustainable learning with digital technologies. These are: *begreifbarkeit* (as the ability to deeply understand (grasp) but also grab something tangible), *Imagineering* (as “means to invent and create yet unknown products that relate to personal life worlds”) and *self-efficacy* (Katterfeld et al., 2015). Their work focuses on constructionist learning environments for digital fabrication with the use of physical computing construction kits and provides implications of their research to extend the educational concept to also include fabricators like 3D printers and laser cutters.

Inspired by their research, we aim to extend this concept with computational design and digital fabrication machines in the spotlight. We find computational design to be a very promising design asset that can greatly contribute to constructionist learning through the creation of tangible projects that can be personally meaningful and enhance the share-ability and customization of the designs (Chytas et al., 2017).

To this point, we evaluated eight workshops that took place in formal and informal learning settings. The first three workshops took place at a university fab lab and were evaluated in-depth using qualitative methods, either by taking part in individual or group semi-structured interviews (regarding their experience and impression of the workshops, the difficulties they met and the things that they would like to be different), observations and evaluation of artifacts. The low number of participants in all three workshops (two participants per workshop) gave us the opportunity to capture with higher precision the reactions and attitudes of participants towards computational design. With some participants we also had the opportunity to introduce non-programming 3D modelling tools to compare with computational design and investigate which features the participants preferred to use. Two workshops took place at a school makerspace and were evaluated with quantitative methods using questionnaires and statistical analysis of their code, while three workshops took place at our informatics learning lab and were analysed with triangulations of the previous methods.

Based on the evaluations of the workshops that were described above, we identify four more elements that showed the potential to enhance the learning activities and personal expression with digital fabrication and computational design. Summarizing our findings, we concluded that when designing a learning environment that exploits digital fabrication and computational design, the following should be kept in mind:

Connecting computational design with our lifeworld

Our workshops aimed at empowering youth and children to use design and programming as means for expression and artistic creation. The benefits of computation could be reflected through designing geometries that included recursive elements and complex design patterns. Designing such patterns without computation requires longer times that tend to be frustrating. Before engaging children and youth in computational practices with digital fabrication they need to understand the role of programming and design in their daily life. Computational design enables the creation of designs so complex that could not be created handmade or with non-programming CAD tools. The participants from our workshops created artistic, practical or entertaining artifacts that were intended for diverse uses ranging from household objects to figures of popular animations and items that were intended for social good. The evaluation from the interviews showed that the participants were more motivated to engage in computational design activities and put more effort into creating something tangible that can be used in their daily life by them or others. Participants that were engaged in personally meaningful projects were intrinsically motivated to ask themselves about the benefits of computing science in their lifeworld and how it can connect with entertainment, personal expression or real-world problem solving in both the digital and physical realm.

Encouraging exploration

We found the demonstration of digitally fabricated artifacts essential to trigger imagination for computational design and digital fabrication. Demonstrating digital and physical artifacts in fab labs, maker faire, makerspaces and websites is a fundamental feature of Maker culture (Posch et al., 2010; Eisenberg, 2017). Novices cannot fully explore the possibilities of digital fabrication technology in the limited time of the workshops. Even people with significant experience in programming and design, can learn about new properties of innovative materials and production or assembly techniques. For example, cutting wooden surfaces with laser cutters in specific patterns can give them the ability to bend wooden surfaces and produce curved objects as illustrated in Figure 4a. When it comes to production of big objects a puzzle-like (snap-fit) modelling like the one illustrated in Figure 4b., can result in an easy and cheap assembly, reducing considerably the production time in comparison with alternative manufacturing methods like 3D printing. Learning about these possibilities of digital fabrication technology requires a lot of time to explore autonomously but the demonstration of physical and digital artifacts provides the basics to trigger the design process.



Figure 4. Bending wood and snap-fit patterns.

Encouraging purposeful design

Computational design tools for hobbyists and professionals provide opportunities to explore interesting patterns, concepts and geometries in design through computational practices. However, there were times when participants generated models that were aesthetically appealing but not intended (Chytas et al., 2018). Creating artistic shapes and geometries that impress by combining programming elements with CAD involves complex thinking processes that could lead to the exploration of new concepts and schemata which cannot be easily calculated and visualized in our minds. Even though such situations could also be fruitful opportunities to explore computation under the lens of design, we need to ensure that the activities are aligned with deep sustainable learning. We found that the creation of objects that are intended to be combined with other technologies (e.g. the creation of a model that is meant to include a microcontroller with LED lights) or serve a specific purpose like replacing a broken part or fit perfect on a surface can overcome this situation in artistic projects as well.

Computational design as a social experience

Let the learners engage in collaborative activities and create a culture of sharing to explore the social dimension of computation and design. Maker culture is aligned with collaboration, sharing and helping others in physical and online spaces and communities (Martin, 2015; Kostakis et al., 2015). The workshops' concept provided opportunities for brainstorming about different ideas, solutions for designs, reflection on the artifacts that were created, working collaboratively on projects, helping or getting feedback from others and getting involved in discussions about personally meaningful projects. Furthermore, the use of computational/parametric design and the rise of the open-source hardware models on the internet enhance the share-ability of the designs. Designers can reach everyone to use their design, modify it by changing the parameters and code or further improve it. In his book *Makers*, Anderson (2012) states that *"The ability to easily "remix" digital files is the engine that drives community. What it offers is an invitation to participate. You don't need to invent something from scratch or have an original idea. Instead, you can participate in a collaborative improvement of existing ideas or designs"* (p. 74). This ability enhances collaboration and sharing by providing customization possibilities that are often missing from traditional non-programming CAD tools.

Conclusion

Computational design and digital fabrication have not been widely used for programming learning activities by youth and children. In this work, we emphasize on computational design as a powerful tool to support design and coding learning activities in constructionist learning settings. We combine computational design with digital fabrication technologies to highlight programming as means for creation in the digital and physical realm. Inspired by the work of Katterfeld et al. (2015) on designing learning environments with digital fabrication, we further focus on elements from computational design that are often missing from the digital-physical swift. These elements can greatly contribute to constructionist learning through the creation of tangible projects that can be personally meaningful and enhance the share-ability and customization of the designs. After evaluating our workshops on computational design, we report best practices on learning with digital fabrication by exploiting the possibilities of computation in modern design tools. We provide examples of embracing opportunities

and ideas behind the maker movement to take advantage of the increasing accessibility of the new tools and foster 21st century skills that are not intended only for industry but personal development and expression as well.

References

- Blikstein, P. (2013). Digital fabrication and 'making' in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, 4, 1-21.
- Blikstein, P. (2013, June). Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th international conference on interaction design and children* (pp. 173-182). ACM.
- BlocksCAD. <https://www.blockscad3d.com/>
- Chris, A. (2012). *Makers: The new industrial revolution*. New York: Crown Business.
- Chytas, C., Tsilingiris, A., & Diethelm, I. (2018, April). Learning programming through design: An analysis of parametric design projects in digital fabrication labs and an online makerspace. In *Global Engineering Education Conference (EDUCON), 2018 IEEE* (pp. 1978-1987). IEEE.
- Chytas, C., Diethelm, I., & Lund, M. (2017) Parametric Design and Digital Fabrication in Computer Science Education.
- Dittert, N., Katterfeldt, E.-S. & Wilske, S., (2014). Programming Jewelry: Revealing Models behind Digital Fabrication. Short paper at FabLearn Europe: Digital Fabrication in Education Conference. June 2014. Aarhus, Denmark.
- Eisenberg, M. (2017). Approaching Computer Science Education Through Making. In *New Directions for Computing Education* (pp. 35-44). Springer, Cham.
- GIMP. <https://www.gimp.org/>
- Inkscape. <https://inkscape.org/>
- Jacobs, J., & Buechley, L. (2013, April). Codeable objects: computational design and digital fabrication for novice programmers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1589-1598). ACM.
- Kastl, P., Krisch, O., & Romeike, R. (2017, November). 3D Printing as Medium for Motivation and Creativity in Computer Science Lessons. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 27-36). Springer, Cham.
- Katterfeldt, E. S., Dittert, N., & Schelhowe, H. (2015). Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction*, 5, 3-10.
- Kostakis, V., Niaros, V., & Giotitsas, C. (2015). Open source 3D printing as a means of learning: An educational experiment in two high schools in Greece. *Telematics and informatics*, 32 (1), 118-128.
- Martin, L. (2015). The promise of the maker movement for education. *Journal of Pre-College Engineering Education Research (J-PEER)*, 5 (1), 4.
- OpenSCAD. www.openscad.org/
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36 (2), 1-11.
- Posch, I., Ogawa, H., Lindinger, C., Haring, R., & Hörtner, H. (2010, June). Introducing the FabLab as interactive exhibition space. In *Proceedings of the 9th International Conference on Interaction Design and Children* (pp. 254-257). ACM.
- TurtleArt. <https://turtleart.org/>
- Vossoughi, S., & Bevan, B. (2014). Making and tinkering: A review of the literature. *National Research Council Committee on Out of School Time STEM*, 1-55.

Developing Mathetic Content Knowledge using an Emergent Systems Microworld

Sugat Dabholkar, sugat@u.northwestern.edu

Learning Sciences and Center for Connected Learning, Northwestern University, USA

Gabriella Anton, gabby.anton@gmail.com

Learning Sciences and Center for Connected Learning, Northwestern University, USA

Uri Wilensky, uri@northwestern.edu

Learning Sciences, Computer Science and Center for Connected Learning, Northwestern University, USA

Abstract

In this paper, we define and develop a theoretical construct Mathetic Content Knowledge (MCK) in the context of children's learning. We discuss acquiring MCK using a special type of constructionist learning environment, which we call Emergent Systems Microworlds (ESMs). ESMs allow students to engage with emergent phenomena in an exploratory way. We argue that the students who participated in our ESM-based curricular unit, GenEvo about Genetics and Evolution not only learned disciplinary core ideas, but also possibly developed mathetic insights into how to learn by engaging in the scientific inquiry process. The GenEvo curriculum incorporates a series of computational models designed using NetLogo that follow the agent-based modeling approach to emergent systems. In this curriculum, students design and conduct computational experiments in the ESM learning environment to figure out the answers to the guiding questions collectively build towards the ideas about emergent properties in the ESM.

We argue for the importance of fostering Mathetic Content Knowledge, knowledge of how to learn by engaging in discipline specific inquiry practices, in general, and specifically for science education. We also discuss how systematic exploration of computational models using ESM learning environments could be an effective way to develop science MCK.

Keywords

Mathetic content knowledge; emergent systems microworlds; science inquiry practices; constructionism; design

Introduction

Papert, in his book *Mindstorms*, introduced the term 'mathetics' to talk about the guiding principles that govern learning (Papert, 1980). Papert made an argument that just like pedagogy is a word for art of teaching, there should be a word for the art of learning. His proposed candidate for that word is 'mathetics'. This word has the same Greek root as mathematics; *mathamein* is a verb which means to learn. Mathetics has much broader connotation as the same root, *math* has in case of the word 'polymath', which means a person of many learnings. Papert later elaborated on this idea in his book *The Children's Machine*, where he talked about mathetics as the art of learning (Papert, 1993). We argue that as learning scientists, when we design learning environments to support different kinds of learning, it is also critical to evaluate whether and how the design supports this 'art of learning'.

In a somewhat related context Shulman introduced the concept of pedagogical content knowledge (PCK), when he argued for the importance of research questions dealing with the content of the lessons taught in the field of teaching and teacher education (Shulman, 1986). Pedagogical content knowledge is about teachers' interpretations and transformations of subject-matter knowledge in the context of facilitating student learning which also incorporates understanding of common learning difficulties and preconceptions of students (Magnusson, Krajcik, & Borko 1999). This has argued to be central part of teacher training programs and argued to be a central area for research in the field of education (Van

Driel, Verloop, & de Vos, 1998). With more incorporation of technology in education, the idea of PCK has been extended to Technology-enhanced PCK (TPCK) (Niess, 2005). Niess (2005) mentions that with technology becoming an integral component or tool for learning, science and mathematics teachers must develop an overarching conception of their subject matter and what it means to teach with technology. All these constructs focus on role of teachers and on the art of teaching. In this paper, we combine these two strands the art of teaching in the context of subject-specific knowledge and technology, and the art of learning, which we call Mathetic Content Knowledge.

Each discipline has a discipline specific practices of engaging in inquiry and construction of knowledge. MCK is strongly connected to these discipline specific practices. The art of learning and being good at this art is different for mathematics, languages, history or sciences. We argue that Emergent Systems Sandbox (ESM), a special kind of constructionist learning environment, allows students to develop MCK in the context of the inquiry practices they engage in. ESMs are a specifically designed to support students in creating, exploring, and sharing virtual models and model-based artifacts of dynamic systems that exhibit emergent phenomena. In this paper, we discuss how students develop MCK when learn genetics and evolution using an ESM-based curriculum called GenEvo.

Research in mathematics and science education over the past few decades has investigated and shown the effectiveness of model-based inquiry in classrooms, both in fostering students' thinking skills and in learning of mathematical and scientific concepts. Specifically, researchers have demonstrated that learning based on investigations of models leads to development of competence in disciplinary inquiry practices such as constructing argumentation based on evidence and communicating it effectively to others (Passmore & Svoboda, 2012; Windschitl et al., 2008; Schwarz et al. 2009) as well as can support content mastery (Stewart et al. 2005). The Next Generation Science Standards (NGSS) stipulate developing and using models as one of the eight core scientific practices (NGSS Lead States, 2013). In particular, these standards suggest that models should be developed "to predict and show relationships among variables between systems and their components in the natural and designed worlds". We demonstrate that model-based inquiry using with an ESM goes beyond merely engaging students this specific scientific inquiry practices; in another paper, we have argued elsewhere that the students that participated in our ESM-based curriculum meaningfully engaged students in several other inquiry science practices recommended by the NGSS (Dabholkar et al., 2018). Engaging meaningfully in scientific inquiry practices is critical to develop Mathetic Content Knowledge for science. As students construct their own knowledge in a microworld by engaging in scientific inquiry practices, such as constructing explanations and engaging in arguments using evidence, they also understand the scientific process of knowledge construction. This understanding is critical for developing MCK for science.

GenEvo: An ESM-based curriculum about genetics and evolution

The GenEvo curriculum incorporates a series of computational models designed using NetLogo (Dabholkar et al., 2016). NetLogo is an agent-based modeling software that has been used for research work regarding emergent systems as well as to design educational curricular units (Wilensky, 1999). The design of computational models in the GenEvo follows the agent-based modeling approach to emergent systems that has been demonstrated to be effective for fostering deep understanding of disciplinary core ideas (e.g., electricity, the particulate nature of matter) as well as crosscutting ideas such as complex systems thinking and computational thinking (Blikstein & Wilensky, 2004; Levy & Wilensky, 2006; Wilkerson-Jerde & Wilensky, 2010).

In this curriculum, students are first presented with a computational model of a bacterial cell with a genetic circuit in which certain components interact in specific manner (See Figure 1) (Dabholkar et al., 2016).

The students explore and play with the model to figure out these interactions and engineer the genetic circuit to make their cells 'fitter' to reproduce. In the next two subunits, students explore and tinker with the models of genetic drift and natural selection. Finally, the cells where genetic circuits are designed by the students will 'compete for survival' in a limited resource environment. These computational models are intentionally designed specifically from the agent-based perspective of modeling emergent systems. In each model, the agents and their behaviors at the micro-level are computationally coded.

The interactions between the agents and their interactions with the environment result in emergence of patterns at macro-level (Wilensky & Resnick, 1999; Wilensky, 1999b). In this curricular unit, the emergent properties of biological systems that students investigate include, genetic regulation, carrying capacity, genetic drift and natural selection. Students design and conduct computational experiments in the ESM learning environment to figure out the answers to the guiding questions in the curriculum. These answers collectively build towards the ideas about emergent properties in the ESM.

Learning using ESM-based curricula

There are two ideas that are central to the learning using ESMs and ESM-based curricula, which we call, 'big-M' Models and 'little-m' models. This theoretical framework has been developed and discussed in detail in the context of Emergent Systems Sandboxes, which is a specific kind of ESM (Brady et. al, 2015). Big-M models are fundamental scientific paradigms (Kuhn, 2012) that form the fundamental basis for design of ESMs. Every entity in the microworld follows the rules that are specified by the Big-M model. Incorporation of these rules involves heavy simplifications of the existing scientific paradigm. However, an ESM is designed in a way that it captures Big-M principles in sufficient details for students to engage meaningfully with those. In contrast, when students participate in an ESM-based curriculum they construct little-m models. Little-m models can be thought of as personal hypotheses or theories about how a system functions. As students construct their little-m models, the consequence of these rules become salient to them. Since, in the most cases, these rules are not available for editing, a construction in an ESM will always be faithful to the Big-M model. It may not produce the aggregate-level behaviors that a student intends, but the outcomes will always be logically determined by the rules of the Big-M Model. Exploring such ESMs and learning to construct little-m models in them would gradually nudge the learner's intuitions into alignment with the Big-M model. Several studies that have used ESMs indicate that it's not a smooth trajectory for students to move from little-m to Big-M in terms of their conceptual understanding of the system. The actual trajectories of these transitions are different for each student. Such open-ended scaffolded explorations along different trajectories have been demonstrated to be effective for students to develop deep understanding of ideas central to a Big-M model.

Data collection and analysis

The data used in this paper is from a Computational Modeling in Biology course based on the GenEvo curriculum. The first author of this paper was the lead-designer of the ESM and the curricular unit and the lead-teacher of these implementations. We conducted this course twice during a weekend extra-school program for middle school students conducted by a talent-development center in a mid-western university in the United States; and in a residential summer camp in a western city in India where students from all over the India participated. The students participated in both these programs were of age 11 to 14 and intellectually advanced based on their academic performance. There were 6 female and 8 male students of mixed racial and ethnic backgrounds; the break-up of self-reported racial and ethnic backgrounds was, 6 White non- Hispanics, 4 Asians, 1 White Hispanic, 1 American Indian or Alaskan Native, 2 Others. In the summer residential program in India, 15 students participated of which 8 were females and 7 were males. All the students were of Asian Indian origin. We collected data in various forms, namely videos of student discussions, screen-capture videos to capture students' investigations of computer models, workbooks in which students wrote their observations and explanations, and the computational artifacts (models and screenshots) that students created.

We use mixed-methods analysis to investigate whether students learned disciplinary core ideas through their participation in ESM-based curricula and how they engaged with science inquiry practices. Using quantitative approach, we have demonstrated elsewhere that the students learned disciplinary core ideas about genetics and evolution using pre- and post- tests (Dabholkar et al., 2018). In order to characterize students' engagement in inquiry science practices used both bottom-up and top-down process coding approach. The bottom-up approach involved process coding to describe student engagement and teacher strategies, whereas the top-down codes are from NGSS recommended science and engineering practices (Miles, M. B., Huberman, A. M., & Saldaña, J., 2014; NGSS Lead

States, 2013). The analysis that we present in this paper is case-based analysis of student learning Mathetic Content Knowledge in this ESM-based computational learning environment.

Developing MCK through scaffolded exploration of GenEvo curriculum

In the course based on the GenEvo curriculum, students work in groups, conduct model investigations individually within a group, and then present, discuss and debate their observations, claims and theories. In this part, we present data of students-teacher discourse, where the students discuss their own theories about an emergent concept, a big-M idea, 'carrying capacity' and how it affected the growth of a population in a computational model. All the students had performed their experiments using the computational model in the GenEvo curriculum (based on the screen recording data) before this discussion started.

Students used their prior knowledge as well as the knowledge they constructed through their explorations of computational models. Owen and Randi⁵⁸ presented their arguments based on their prior knowledge. Owen had mentioned that 'carrying capacity' is the amount of food an organism carries and that led him to an incorrect inference of his observation.

Owen: "....., if you have more **carrying capacity**, that means the **cells can carry a lot more food** which means they can split faster"

In the above sentence, Owen attributes carrying capacity as a property of a cell, whereas in fact it is property of the environment. This is an example of level-slippage where a learner attributes patterns or properties of macro-level to micro-level or vice-versa (Wilensky & Resnick, 1999; Levy & Wilensky, 2008). In the computational model, students can change carrying capacity settings to conduct different computational experiments in order to figure out what it means and how it influences the population growth. Such playful explorations resulted in Hasan constructing the knowledge about carrying capacity through his own explorations. When asked about who he knew that answer, Hasan mentioned, '*I was just playing with it and I noticed.... the amount of cells....*'

The following is the complete except as an example of students arguing about what 'carrying capacity' means based on experimental evidences or their prior knowledge.

Teacher – "Oh! Do you? I don't. Can you explain it (what carrying capacity means) again to me?"

Owen – "So pretty much, **if you have more carrying capacity, that means the cells can carry a lot more food** which means they can split faster like in the last thing, which means they will be bigger and eat a lot more and need a lot more food."

Teacher – "Do you all agree to that?"

Hasan – "It seems like he is saying that carrying capacity is how much food they sustain but **carrying capacity is how much the map...the square can hold.**"

Randi – "Ya, because carrying capacity is how many ..."

Hasan – "**cells**"

Randi – "Ya, and he is talking about food."

(Transcript from video data, March-19, 2017)

In this conversation, Owen attributed 'carrying capacity' as a property of a cell, which is based on his prior incorrect conception which he tries to connect with his experimental observation of *cells splitting faster*. The teacher then directed the question to the rest of the class. This is an important teacher strategy of not correcting Owen's answer, rather than trying to correct Owen's conception and reasoning. As a response to the teacher's question, Hasan and Randi argued differently. Hasan referred to the *map or the square* which represents the environment in the computational model to demonstrate his little-m idea, which is his contextual understanding of the big-M concept, carrying capacity. Randi supported him in the argument. This analysis demonstrates that these students used different reasoning

⁵⁸ All the names used in this paper at pseudonyms.

approaches to explain their answer, some of which were based on the prior knowledge whereas some were based on the experimental investigations. It also shows that the student, Hasan, who did not have any prior knowledge, mentioned that he was just *playing around* with it and noticed the *amount of cells*. It is also important to notice that Hasan is referring to his explorations of computational model as playing around. In this class, as the students learn the concept of carrying capacity, they also investigate how to figure out what carrying capacity means and how it affects the system they are studying. This understanding that knowledge can be constructed and verified through systematic investigations, and that's how the scientific knowledge is generated is critical for developing science MCK.

This kind of development of MCK is also evident in another part of our analysis of students' reflections of what they learned when they participated in systematic exploration of the computational models in the curriculum. In the weekend course at a Midwestern suburb in the United States, by the end of the second day the teacher asked what they learned in the two days and how. After Alex mentioned all the disciplinary ideas they learned like genetic regulation, cell producing energy and that effecting their growth rates, genetic drift and natural selection, the teacher asked about how they learned these ideas, if they worked like scientists, if they figured out stuff like scientists. Alex continued, "*Yeah, because we came up with some kind of theory and we built off of it.*" Tanya, another student in the class added, "*Like Alex said, we came up with theories and we worked to either prove them right or prove them wrong.*" (*Italicized part is from the transcript from video data, April-01, 2017*). In the further discussion, the students talked about which theory they proved to be right and which theory they proved to be wrong. Tanya, who had strongly supported a theory (her little-m idea) that lactose is bad for cells and blue color for death, was quick to point out that it was the theory that was proved to be wrong. Throughout the discussions over the two days the teacher never mentioned that a theory is right or wrong, until the students presented evidence in support of it and against it, and argued about it. When the teacher asked if this experience was different from the other learning experiences, Alex said, "Yeah, it was a lot different. Because we have done this kind of stuff before with the models that are interactive. But, normally it's just like follow these steps and combine those to find answer." He was possibly referring to the open-ended nature of the scaffolded explorations where there was no 'standard procedure' to arrive at a 'correct answer', rather the investigative approach for gathering evidence in support of arguments was emphasized in such kind of ESM learning environments.

In the previous part, we have discussed how students engage in learning inquiry science practices with an ESM-based curriculum and what they think about that engagement in scientific practices. In this part, we discuss how students expressed what they learned the course and how they learned it. These are the quotes from the students in who participated in the Computational Modeling of Genetics and Evolution course in India. The first quote is from Amita, who did not participate much in the discussions and when she did, she made some very sharp points. Amita wrote that "*In the past few days, I lived my life like a scientist. I made observations and presented those in front of others. I learnt a lot of stuff which I believed till now was impossible.*" Amita referred to something new that she learned which was beyond her imagination. Dinesh wrote that "*The most important thing I learnt is to observe and learn from others. I also learned that it is important to give credit to other people for their contributions....*" Two important things reflected in Dinesh's writing are science being a collective enterprise of constructing knowledge and citing others to give credit for their work. Dipti mentioned that "*... I got to know a new way of learning.*" Akshay mentioned that "*For me, the most important thing to learn was how to learn.*" Dipti and Akshay's quotes are indicative of the fact that the students thought about the process of learning. Vasumitra's quote captures how he thought he learned; he wrote "*The thing I learnt was designing my own experiments, collect the proof and make observations. I also learnt that it is fine to be wrong and it ultimately leads to our betterment.*"

Reflections of these students on the process of learning in this learning environment indicate that how they understood the process of constructing knowledge by the scientific community as well as how they perceived their role in this science classroom. Vasumitra's reflection is an indication of how he thought that science MCK, which is about constructing knowledge by conducting experiment, collecting proofs and making observations, and more importantly making mistakes is critical for one's learning in a science class.

Conclusions and Implications

In this paper, we argued for the importance of fostering Mathetic Content Knowledge, knowledge of how to learn by engaging in discipline specific inquiry practices, in general, and specifically for science education. We also discussed how an ESM-based curriculum can be effective in fostering learning of MCK. With increased emphasis on the scientific inquiry practices in science curricula and use of model-based inquiry learning for the same, characterization of inquiry learning with environments like ESM and understanding design principles for such environments is critical. We have presented two kinds of evidences using case-based analysis in this paper. First, we analyzed student participation in an ESM-based curriculum and demonstrated how students use various reasoning strategies in engagement in an argument, how they can use evidence from ESM to support their arguments, and to construct knowledge. We also presented data about how students view their own engagement in inquiry science practices and how they view their learning with an ESM-based curriculum. We argue that systematic exploration of computational models using ESM learning environments are effective in developing science MCK.

Acknowledgements

We thank Connor Bain and Ümit Aslan who helped us in designing the models and curricular units. We also thank Aniruddh Sastry who has helped in implementation of the curricular unit in India. We are grateful to Educational Initiative's ASP for helping us conduct the research in India. Authors 1 and 2 gratefully acknowledge Learning Sciences program at Northwestern University for the funding support.

References

- Barnett, J. (2003). Examining pedagogical content knowledge: The construct and its implications for science education, *87*(4), 615-618.
- Blikstein, P., & Wilensky, U. (2010). MaterialSim: A constructionist agent-based modeling approach to engineering education. In *Designs for learning environments of the future* (pp. 17-60). Springer US.
- Brady, C., Holbert, N., Soylu, F., Novak, M., & Wilensky, U. (2015). Sandboxes for model-based inquiry. *Journal of Science Education and Technology*, *24*(2-3), 265-286.
- Dabholkar, S., Bain, C. and Wilensky, U. (2016). NetLogo GenEvo 1 Genetic Switch model. <http://ccl.northwestern.edu/netlogo/models/GenEvo1GeneticSwitch>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Dabholkar, S. & Wilensky, U. (2016). GenEvo Systems Biology curriculum. <http://ccl.northwestern.edu/curriculum/genevo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Dabholkar, S., Anton, G., & Wilensky, U. (2018) (accepted) GenEvo - An emergent systems microworld for model-based scientific inquiry in the context of genetics and evolution. *Proceedings of the International Conference for the Learning Sciences*.
- Goldstone, R. L., & Son, J. Y. (2005). The transfer of scientific principles using concrete and idealized simulations. *The Journal of the Learning Sciences*, *14*(1), 69-110.
- Kuhn, T. S. (2012). *The structure of scientific revolutions*. University of Chicago press.
- Levy, S. T., & Wilensky, U. (2008). Inventing a "mid level" to make ends meet: Reasoning between the levels of complexity. *Cognition and Instruction*, *26*(1), 1-47.
- Levy, S. T., & Wilensky, U. (2011). Mining students' inquiry actions for understanding of complex systems. *Computers & Education*, *56*(3), 556-573.
- Magnusson, S., Krajcik, J., & Borko, H. (1999). Nature, sources, and development of pedagogical content knowledge for science teaching. In *Examining pedagogical content knowledge* (pp. 95-132). Springer, Dordrecht.

- Miles, M. B., & Huberman, A. M. (1984). *Qualitative data analysis: A sourcebook of new methods*. In *Qualitative data analysis: a sourcebook of new methods*. Sage publications.
- National Research Council. (2013). *Next generation science standards: For states, by states*.
- Niess, M. L. (2005). Preparing teachers to teach science and mathematics with technology: Developing a technology pedagogical content knowledge. *Teaching and teacher education*, 21(5), 509-523.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. BasicBooks, 10 East 53rd St., New York, NY.
- Passmore, C. M., & Svoboda, J. (2012). Exploring opportunities for argumentation in modelling classrooms. *International Journal of Science Education*, 34(10), 1535-1554.
- Schwarz, C. V., Reiser, B. J., Davis, E. A., Kenyon, L., Achér, A., Fortus, D., ... & Krajcik, J. (2009). Developing a learning progression for scientific modeling: Making scientific modeling accessible and meaningful for learners. *Journal of research in science teaching*, 46(6), 632-654.
- Sengupta, P., & Wilensky, U. (2009). Learning electricity with NIELS: Thinking with electrons and thinking in levels. *International Journal of Computers for Mathematical Learning*, 14(1), 21-50.
- Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational researcher*, 15(2), 4-14.
- Stewart, J., Cartier, J. L., & Passmore, C. M. (2005). Developing understanding through model-based inquiry. *How students learn*, 515-565.
- Wilensky, U. (1999). *GasLab NetLogo* [computer software]. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. <http://ccl.northwestern.edu/netlogo>.
- Wilensky, U. (1999). GasLab—An extensible modeling toolkit for connecting micro-and macro-properties of gases. In *Modeling and simulation in science and mathematics education* (pp. 151-178). Springer, New York, NY.
- Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach. *Cognition and instruction*, 24(2), 171-209.
- Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and technology*, 8(1), 3-19.
- Windschitl, M., Thompson, J., & Braaten, M. (2008). Beyond the scientific method: Model-based inquiry as a new paradigm of preference for school science investigations. *Science education*, 92(5), 941-967.
- Van Driel, J. H., Verloop, N., & de Vos, W. (1998). Developing science teachers' pedagogical content knowledge. *Journal of research in Science Teaching*, 35(6), 673-695.

Making Together: Cultivating Community of Practice in an All-Girl Constructionist Learning Environment

Caitlin Davey, *csd2126@tc.columbia.edu*
Teachers College, Columbia University USA

Sawaros Thanapornsangsoth, *st2839@tc.columbia.edu*
Teachers College, Columbia University USA

Nathan Holbert, *holbert@tc.columbia.edu*
Teachers College, Columbia University USA

Abstract

This practice paper investigates instructional practices that support relationships among young female makers. It summarizes a design-based research study where a group of all-female makers constructed toys for younger students in their immediate school community. Makers supported one another through contributing ideas, building together, sharing expertise, and providing helpful encouragement. The finding shows that the classroom practices extended beyond physical construction to include playful interactions amongst makers such as: switching projects, working together outside class time, and a myriad of other activities both related and unrelated to making. Through the process of making, they developed individual expertise that contributed to the sharing of knowledge within their classroom community. Drawing on literature from constructionist design paradigms and community of practices, data from this Making and Engineering class describes an emergent community of makers. Additionally, this paper highlights the value of makers creating personally and socially meaningful projects in collaboration with others. Finally, we describe the flexible and playful environment of the Making and Engineering classroom that contributed to how makers shape their shared practices.

Keywords

constructionism; community of practice; maker education; girls in making

Introduction

Makerspaces are venues where individuals gather around a shared interest in making to learn, create, and share expertise. Sharing of expertise has been identified as a key feature of these spaces that helps to induct new members and expand the making community (Halverson & Sheridan, 2014). This practice paper aims to illustrate the importance of near-peer communities of practice (Lave, 1991) within an all-female makerspace as research on collaboration in makerspaces has thus far focused on inter-generational or adult learners who develop their expertise to teach or mentor younger makers (Blikstein, 2013; Halverson & Sheridan, 2014; Resnick & Rusk, 1999; Holbert, 2016). Our study investigates the collaboration between elementary-aged makers in a Making and Engineering class. This paper examines how young makers who are newly introduced to Making and Engineering, support one another during the making process and gain individual expertise through their collaborations. In examining this community of young makers our paper aims to answer the following questions:

- How does working alongside peers influence the process of making?
- How do young female makers adopt and develop specialties when assisting one another during making activities?
- How did the Making and Engineering classroom structure influence the practices developed by a community of young female makers?

In making artifacts alongside their peers, an emergent set of practices developed among the makers. Several cases will be analyzed to examine evidence of peer-support indicative of a community of practice within the constructionist classroom project. The values of this community of practice uniquely

supported the peer-to-peer and peers-to-client relationships of this instructional design. Sharing newly formed expertise, collaborating, and providing supportive feedback were facets of their making practice. Through assisting others, makers reinforced their learning of essential engineering and design skills.

This paper presents cases of students that exemplify the community of practice formed within the all-girl maker classroom. These cases emerged from field note data analysis and one-on-one interviews.

Literature Review

Constructionism: “In the world” tangible and sharable knowledge

The activities employed in the Making and Engineering class draw on the constructionist design paradigm. Papert’s constructionism extends Piaget’s constructivism by proposing that the construction of knowledge “in your head” happens best when constructing tangible and shareable objects “in the world” (Papert & Harel, 1991; Papert, 1993). Constructionism is a “framework for action” (DiSessa & Cobb, 2004,) situated and pragmatic (Ackermann, 2001; Noss, 2010). In constructionist design learners build public artifacts that are personally and socially meaningful. This construction process leverages diverse ways of thinking, knowing, and practicing (Turkle & Papert, 1990) and situated learning so it is about developing concrete relationships with objects and ideas (Ackermann, 2001; Wilensky, 1991).

The creation of an artifact, which could be either physical or virtual, allows learners to externalize their mental models and iterate on their thinking throughout the making process. Additionally, it enables learners to see and critique one another’s work (Papert & Harel, 1991). Resnick (2004) highlights the value of learners playfully creating personally-meaningful projects in collaboration with their peers. He believes that learning should be a social activity where learners share ideas, collaborate on projects, and build on others’ work. In their “Instructional Software Design Project” research, Kafai and Harel (1991) found that by ensuring all students in a class worked towards the same goal, in this case creating math games to teach younger students about fractions, the students were able to support each other as they faced similar problems, shared ideas, helped others and discussed technical problems. Some students in this study choose to work together by discussing problems with a partner or working together as a team. Even those who chose to work alone, sought help or inspiration from others’ for their work. Working on the same project goal is an engine to drive constructionism and communities of practice. However, the environment needs to be designed to encourage and support learners’ collaboration and sharing. Learners learn better when they are immersed in an environment and community that supports their interests (Papert, 1980). Access to other creators can be especially important for deepening learner’s expertise through receiving feedback, brainstorming ideas, working on projects together, and finding encouragement (Ito et al., 2009).

Community of practice

Learning in a community of practice is inherently social (Lave & Wenger, 1991). “Communities of practice sprout everywhere, in the classroom as well as on the playground, officially or in the cracks” (Wenger, 1998, p.6). Instead of didactic instruction from a teacher to a learner, this viewpoint describes a diverse set of essential actors and forms of participation. Lave (1991) explains that structure and experience reinforce each other such that they shape the relations among persons acting, settings, situations, and systems of activity. Near-peers, members of a community matched in ability, are identified by Lave and Wenger (1991) as important in the circulation of knowledgeable skill.

Just as there is a rich field of actors, there may be many different ways of participating in a community of practice. As members develop mastery through peripheral participation, they move into more central modes of participation within the community. However, there may be no such thing as a central role. The notion of legitimate peripheral participation outlined by Lave and Wenger (1991) is used to suggest that there are “multiple, varied, more- or less-engaged and inclusive ways of being located in the fields of participation” defined by a community (p. 36). Developing an identity gives meaning to shared skills which, are then incorporated into identities. Forming an identity as a member of a community and developing skills are mutually reinforcing.

Participation is central to learning in a community of practice. Learners absorb and become absorbed in the culture of practice, which increasingly provides them with occasions to make the culture of practice

their own (Lave & Wenger, 1991). As described by Halverson and Sheridan, “communities of practice emerge around makerspaces as members co-participate in a range of activities” that go beyond making to unrelated socializing (Halverson & Sheridan, 2014, p. 502). Makerspaces can be understood as communities of practice where making activities are a part of a larger in-person, or online, community.

Methodology

Population and Site

This practice paper is part of a larger design-based research project called “Bots for Tots” which aims to increase diversity in maker and engineering design activities (Holbert, 2016). In investigating girls’ participation in maker activities, we draw on data from the second iteration of the project, where the population is solely female participants. This iteration of Bots for Tots took place at an all-girls private school in a suburban area in the North-Eastern United States. 41 fourth-grade makers (aged 9-11) from two classes (20-21 students per class) participated in the study as a part of their Making and Engineering class, a bi-weekly class which ran 45-minutes per session. Throughout the academic year, students participated in 17 sessions in total as described in Table 1.

The school has a long tradition of pairing fourth grade “Big Sisters” with first grade “Little Sisters” under its Big Sister/Little Sister mentorship program. For the first time, as part of the Bots for Tots project, the fourth grade Making and Engineering class had the explicit goal of designing and building “dream toys” for their first grade Little Sisters. The class began with two sessions of making with a 2D- to 3D-objects with cardboard in order to familiarize them with tools and materials in the lab as well as preparing them up for creative activities ahead. After the two cardboard sessions, the fourth-grade makers interviewed their Little Sisters using the “My Client Profile” worksheet. During this interview makers questioned their Little Sisters about toys by asking: What kind of toys do you like? If you could imagine any toy, what would it look like and how would you play with it? Makers then used the completed My Client Profile to discuss and brainstorm toy ideas with their classmates. After three sessions of prototyping, makers met their Little Sisters again to receive initial feedback on their prototypes before they began working on their final toy construction. The makers then spent seven sessions building and completing their final designs. Finally, the makers delivered their newly constructed toys to their Little Sisters and had a playdate where the girls discussed the toy’s design and construction while playing together.

Table 1. Structure of the Making and Engineering class.

Name of activity	# of sessions the activity occurs	Major activities
Make 2D to 3D cardboard animals	3	Makers drew animals they selected on a piece of paper and used cardboard to make it 3-D.
Interview with Little Sister	1	Interviewed first-graders about their dream toys
Brainstorm with small group	2	Shared information. Each maker gained from their Little Sister and asked for classmates’ input about their design ideas.
Prototype	2	Made prototypes
Meet Little Sister for prototype feedback	1	Showed Little Sister their prototype and asked for feedback
Complete final toy design	7	Revisited the feedback from Little Sister and planned for improvements. Finalized toy construction
Toy delivery and play date	1	Met Little Sister for toy delivery, explained the design of the toy, and played together.

Data Collection

All names used in this paper are pseudonyms chosen by the makers.

Interview: While all 41 makers participated in the study, 12 were randomly selected for one-on-one interviews. We interviewed the makers at the beginning of the year, before the Making and Engineering class had begun to determine makers' experience with technology, construction, and crafts as well as knowledge of relevant Making and Engineering concepts or skills. Interviews lasted approximately 40 minutes per participant. We interviewed the same makers again at the end of the year after the class had concluded. In the post interview, makers were asked about their experience working with their friends and making toys for their Little Sister. Our goal was to understand how making with others influenced the overall making process and how makers may have developed and adopted specialties when collaborating. All interviews were initially video-recorded then, transcribed first by both the authors and then by an independent service.

Field notes: Detailed field notes were taken during observations of the Making and Engineering class. These observations focused on how the fourth-grade girls interacted with their classmates in the Making and Engineering learning environment (offering help, sharing materials, developing the sense of expertise, providing each other feedback and support).

Artifacts: A variety of maker artifacts were produced throughout this project. These include worksheets (My Client Profile worksheet, a worksheet to record feedback from their Little Sisters, and others), photographs of participants working on their projects, as well as photographs of their toy designs throughout the construction process. These artifacts provided a broad picture of each participant's work, such as whether they worked independently or alone, as well as their level of expertise both in technique and constructed toy. All artifacts were de-identified.

Results

Idea Generation

Idea Sharing

After conducting the initial client interviews with their Little Sisters, the teacher had makers gather in small groups to brainstorm ideas. Linda noted under "Concerns" on her "My Client Profile" worksheet that her Little Sister was, "hesitant about answering making me think that toys are not her favorite and she said that she likes toys that draw." Three of her peers provided suggestions about how to design a toy that would address these concerns. One girl tried to convince Linda to make a huge stuffed crayon so it would be, "soft like a fuzz ball." Another, added it could be a double-sided marker. Building on this idea, another maker told Linda the marker should have four colors. While Linda wasn't initially interested in a multi-sided marker, at the end of the class session on her post-it note summary she wrote, "Double sided marker. Put a squishy thing on top. Four sided marker/crayon and put squishy things on the handle."

Co-constructing

As there were more fourth-grade Big Sisters than first-grade Little Sisters, four makers had to share a Little Sister with another maker. Fourth-grade makers Susie and Paula decided to make a toy together for their Little Sister, while Betty and Chloe made two separate toys.

Susie and Paula interviewed their Little Sister together and wrote similar findings on both of their "My Client Profile" worksheets. For example, under the heading "Likes" they wrote, "yellow, blue, medium, real animals, panda, cheetah, hard" and under "Concerns," they wrote, "don't break, no spider, and colorful". She also made the observation that their Little Sister "likes dolls with clothes." In order to assist the makers with planning their projects, the instructor made a "Brainstorming" worksheet as a helpful guide. The makers were asked to answer the following questions: "I want to explore:" and "My sketch for (Little Sister's name)" Susie and Paula both crossed out the word "my" and "I" on their worksheets and changed those words to read, "we" and, "our" as shown in Figure 1. They summarized their shared design plan as making a "wooden or plastic doll with movable parts and brown hair." They worked together for the duration of the school year and consistently divided their making efforts.

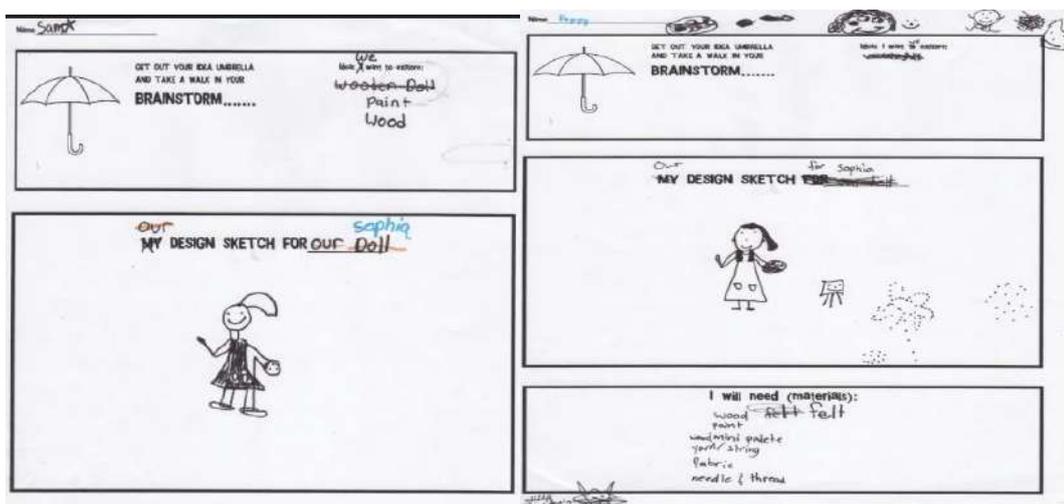


Figure 1: Susie and Paula's Brainstorming worksheet.

In contrast, Betty and Chloe did not work together. In the post-interview Betty reflected back on her making process. She described how she didn't make the toy her Little Sister asked for. Betty said, "I shared my Little Sister with my friend Chloe. I was trying to make her a stuffed bunny...I feel like we should have worked together. It would have turned out a lot better...yeah." Betty described that as the making process progressed she realized, "that it was kinda dumb," not working together.

Making Process

Overcoming challenges

Betty intended her dollhouse to have two levels. But, she cut one piece of wood slightly smaller than the others so when she glued her house together it was slanted. She looked unhappy with her design so her classmate provided encouragement by saying that it looked "like a cool modern loft house!" Another agreed and said, Betty could decorate her dollhouse as a fun, whimsical witch house. However, Betty still appeared unsatisfied. She decided to take the second floor out and redesign her roof. Though this process delayed her work, Betty offered continued help to her friend Maya.

Maya's project was a stuffed animal and Betty said she "helped Maya a couple of times because [Maya] just didn't know how to sew." Betty taught Maya to hand sew because there was a queue to use the sewing machine. Betty humorously talked about her experience supporting Maya, "I just kind of showed her how to like sew, and she's like, 'Oh.' And then she tried to use the sewing machine and it just got all bunched up. I'm like, 'Okay Maya, you go hot glue this onto my house and I will cut all this off.'" Maya and Betty worked on each other's projects to finish their toys in time.

Sharing expertise

In the post-interview Hailey said, "I asked my mom to teach me the sewing machine at my house so that helped me a bit when I sat down at the sewing machine there [referring to the one in the classroom], I certainly know how to use it, to thread it and stuff so that's a big help." Hailey described the best part of the class as helping others saying, "you got to help other people...like...oh can you help me thread the needle?" referring back to the sewing machine needle. She added that "[helping others] felt really good." Hailey helped others sew by providing procedural cues such as, "you gotta pin it. You're going to start here...leave a place open to stuff," and to another maker, "pick the guide up. Pick the needle up. Rotate." In one instance, Madison wanted to begin sewing her project. She asked Hailey, "can I sew it now?" Hailey responded, "can I just get you started?" Madison replied, "can I do it? I'm kind of a control freak." Hailey then said, "trust me, I do it too." After more prompting, Hailey stepped back from the machine. For Hailey's own project she sewed a pillow for her Little Sister. To test it, Hailey placed the pillow on the ground and rested her head on it. She then solicited feedback on her pillow design by inviting a friend over to also rest their head.

Wrapping up

Facing time constraints

Towards the end of the school year, Margie had finished her toy early, a purple stuffed tennis ball. In the post-interview Margie said that her friend, Cathy suggested that she make a tennis court to go with the tennis ball. While Margie used her extra time to supplement her toy, Ellie walked around the class and shouted, “who needs help?”. LillyJane mentioned Ellie in her interview. She said that she received help from Ellie and that if she had extra time, she would have offered similar assistance to her peers, “If I was finished with mine, people were coming around and helping people like Ellie. She helped me. But like I didn’t get, because I was doing my stuff, and I didn’t have enough time to help other people.” Hailey also mentioned Ellie as a person she went to get confirmation from on her ideas.

Erica had her friends help her by finding materials and painting her doll house because it “was pretty big”. Similarly, Betty was making a dollhouse and needed help making items to furnish it. She told us during the interview that she saw Chloe “just wandering around doing nothing, so it was like, ‘Can you help me make the chairs?’” Holly was also behind in finishing her toy doll during the final class period and enlisted her classmate Anna to help her make a skirt and hair for the doll. When the instructor told the class to clean up, Holly asked Anna if she could visit her house over the weekend to help her with the project.

Discussion

Making with others

Working with others influenced the makers process of constructing toys for their Little Sisters. The Making and Engineering class created a community that involved making, brainstorming, designing, helping, valuing others’ ideas, and sharing their projects. When Linda was struggling to generate an idea that responded to her Little Sister’s toy preferences, she asked for input from her peers. Linda incorporated all of their ideas to create her soft three-sided marker project. Valuing the ideas of others emerged as a characteristic of this community. Makers reified their practices through the toys they created for their Little Sisters (Wenger, 1998). Though Linda was responsible for her own project, her work reflects the contributions of her four fellow makers.

Making tangible objects acted as a medium for the makers to communicate their abstract ideas to each other. It became an external representation of the makers’ ideas that enabled them to play and to gain a better understanding of possibilities and limitations (Resnick, 2013). Sharing the same Little Sister, Susie and Paula needed a common vision to construct together effectively. From the beginning of the process, they compared their “My Client Profile” worksheets to generate toy ideas. On their “Brainstorming” worksheet, they both sketched out a picture of a girl wearing an apron with a paint palette as shown in Figure 2. This sketch can be understood as a mutual concrete representation of their project. While they individually sawed each wooden piece, they would frequently piece them together to align their designs. The artifact they shared allowed Susie and Paula to communicate more clearly about their progress towards their shared vision.



Figure 2: Susie and Paula were sawing wood together to make different parts of their doll.

Creating an object allowed learners to explore and expand their ideas but also to see their thinking and provide feedback on a visible artifact (Pepler & Hall, 2016). When Betty was disappointed by her slanted two-level dollhouse and wanted to quickly make a pillow instead. She received feedback and encouragement from her peers, that the house looked “cool” and “whimsical,” after this reassurance she decided not to start a new project, but figured out a way to improve her existing one. Just as “having a tool to perform an activity changes the nature of the activity,” so did having objects to share their ideas around change the makers’ process of making and helping one another (Wenger, 1998, p. 59). Betty was able to refer to her project and how it differed from her intended design. But also, her classmates supported her by reframing her project, providing encouragement, and easing her disappointment so that she was able to progress past this mistake her project.

Developing expertise

As the makers progressed in their projects they began to develop specialized areas of expertise. In doing so, some members such as Hailey can be understood as moving into more central roles within their community of practice (Lave & Wenger, 1991). Though makers were not assigned stations to oversee, Hailey took it upon herself to improve her sewing skills by asking her mother to teach her at home. She used her expertise to provide support to her peers and assist them in developing their own abilities. Hailey is shown assisting her peers in Figure 3. The makers can be understood as displaying different learning trajectories or modes of participation, as not all makers developed expertise in one domain. Hailey’s proficiency was developed to support a core practice that emerged within this community, supporting others. As expressed by Hailey, “[helping others] felt really good.” Her knowledge of sewing allowed her to support others and made her feel proud.



Figure 3: Hailey (first from the left) helping her friends at the sewing machine station.

Similarly, Ellie positioned herself as a creative contributor to the classroom community by walking around the room and providing suggestions to her classmates. Ellie’s project, a giant stuffed purple snake, was highly praised by her friends and teachers. Ellie’s skills at making mutually reinforced her identity as a proficient maker that she adopted within the Making and Engineering classroom (Lave & Wenger, 1991). She expressed her expertise, by going around the classroom and providing help to her friends who were struggling with their own projects. For Ellie, being an expert wasn’t simply about having specific skill, soft toys, but being able to be a generalist who could assist anyone in the classroom.

Making and Engineering classroom

Though the makers were designing for others, by working alongside each other their construction became more personally meaningful. An important instructional decision within the Making and Engineering class was to allow makers the freedom to move around and express themselves. The program also ran throughout the academic year. This allowed makers to have enough time to experiment and tinker with multiple new ideas and iterate on their designs (Resnick, 2013). The makers practice within this space involved activities that extended beyond making. Halverson and Sheridan (2014) described communities of practice in makerspaces as involving activities that moved beyond making to include playing board games, caring for resident pets, and taking walks together. Similarly,

the much younger all-female makers in this Making and Engineering school-based class were able (and allowed) to talk, sing, and move around the classroom as they worked. The makers expressed their enjoyment of making together. When Hailey wanted to test the pillow she had created she lied down on the floor and invited a friend to rest their head next to her to try out the pillow as well. The classroom environment allowed makers to more freely interact with their peers, whether it was inviting a friend to rest their head on your pillow or by singing a song together.

Despite the open-ended instructional approach, many students experienced time constraints that impacted their making. As Lave (1991) explains structures and practices reinforce each other to shape the relations among actors, settings, and systems of activity. Structures contributing to the formation of this community can be understood as the single gender school, the “Little Sister” program, and the tight time restrictions of each classroom period. When Maya couldn’t complete her stuffed animal because she didn’t know how to use the sewing machine she sought help from a friend. Her friend, Betty tried to teach her but she still had difficulties operating the machine in Figure 4. Betty hadn’t finished her own project yet despite wanting to assist her friend. They switched their projects so their abilities could match the remaining tasks and they both could work more efficiently. Another set of makers, Anna and Holly, indicated that they would continue working together outside the classroom. Holly invited Anna to her house on the weekend to help her finish her toy. Despite time-constraints, the flexible working environment allowed these makers to creatively address their problems whether by continuing to work together outside of class or by dividing up tasks across projects.



Figure 4: Betty teaching Maya how to use the sewing machine.

Conclusion

Through our analysis of young female makers who formed an informal community of practice, we provide insights into what learning looks like in a school-based constructionist learning environment. Working with others influenced the makers processes of constructing toys for their Little Sisters. As clearly demonstrated in the collaborative efforts of Susie and Paula, making changed from being an “I” and “my” process to a “we” and an “our” one. As the makers created their projects, they developed different areas of expertise. Whether specific knowledge or general making skills, makers used their abilities to complete their own projects as well as assist others. The classroom practices extended beyond physical construction to include playful interactions amongst makers such as: singing, resting on the floor, switching projects, working together outside class time, and a myriad of other activities both related and unrelated to making. By looking into how young girls make and interact in a constructionist learning environment, we can better facilitate and cultivate the learning community where learners help one another, share their knowledge, and create meaningful projects together.

References

- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication*, 5(3), 438.
- Blikstein, P. (2013). Digital fabrication and 'making' in education: The democratization of invention. *FabLabs: Of machines, makers and inventors*, 4.
- DiSessa, A. A., & Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *The journal of the learning sciences*, 13(1), 77-103.
- Halverson, E. R., & Sheridan, K. (2014). The maker movement in education. *Harvard Educational Review*, 84(4), 495-504.
- Holbert, N. (2016). Leveraging cultural values and "ways of knowing" to increase diversity in maker activities. *International journal of child-computer interaction*, 9, 33-39.
- Ito, M., Baumer, S., Bittanti, M., Cody, R., Stephenson, B. H., Horst, H. A., ... & Perkel, D. (2009). *Hanging out, messing around, and geeking out: Kids living and learning with new media*. MIT press.
- Kafai, Y., & Harel, I. (1991). Learning through design and teaching: Exploring social and collaborative aspects of constructionism.
- Lave, J. (1991). Situating learning in communities of practice.(pp. 63-82).
- Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge university press.
- Noss, R. (2010). Reconstructing Constructionism. In J. E. Clayson & I. Kalas (Eds.), *Constructionist approaches to creative learning, thinking and education: Lessons for the 21st century*. Paris, France.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1—11.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Basic books.
- Peppler, K., & Hall, T. (2016). The make-to-learn youth contest: Gaining youth perspectives on learning through making. *Makeology: Makerspaces as learning environments*, 1, 141-157.
- Resnick, M., & Rusk, N. (1999). 11. The Computer Clubhouse: Technological Fluency in the Inner City. *High technology and low-income communities: prospects for the positive use of advanced information technology*.
- Resnick, M. (2013). Lifelong Kindergarten. *Cultures of Creativity*. LEGO Foundation.
- Turkle, S., & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of women in culture and society*, 16(1), 128-157.
- Wenger, E. (1998). *Communities of practice: Learning, meaning, and identity*. Cambridge university press.
- Wilensky, U. (1991). *Abstract meditations on the concrete and concrete implications for mathematics education*. Epistemology and Learning Group, MIT Media Laboratory.

Assessment of Modeling Projects in Informatics Class

Natasa Grgurina, *n.grgurina@rug.nl*
University of Groningen, Netherlands

Erik Barendsen, *e.barendsen@cs.ru.nl*
Radboud and Open University, Netherlands

Cor Suhre, *c.j.m.suhre@rug.nl*
University of Groningen, Netherlands

Klaas van Veen, *klaas.van.veen@rug.nl*
University of Groningen, Netherlands

Bert Zwaneveld, *zwane013@planet.nl*
Open University, Netherlands

Abstract

The introduction of the new Informatics curriculum in the Netherlands in 2019 raises the need for new teaching material that includes practical assignments and guidelines for their assessment. As a part of our research project on teaching Computational Science (modeling and simulation), we participate in these efforts and developed a curriculum intervention and an assessment instrument consisting of a practical assignment and grading rubrics to assess student's level of understanding. The rubrics we developed can be used both for formative and summative assessment. In this paper we describe the design of this assessment instrument and indicate further research directions focusing on validation of this instrument.

Keywords

modeling and simulation; NetLogo; assessment; SOLO-taxonomy

Introduction

In the Netherlands, where informatics is an elective subject in grades 10 and 11 of the senior general secondary education spanning grades 7 through 11 (in Dutch: HAVO) and in grades 10 through 12 of the pre-university education spanning grades 7 through 12 (in Dutch: VWO), the new 2019 informatics curriculum recognizes the importance of modeling and includes an elective theme comprised of modeling and simulation, together called *Computational Science*. It is described by the high-level learning objectives: “*Modeling: The candidate is able to model aspects of a different scientific discipline in computational terms*” and “*Simulation: The candidate is able to construct models and simulations, and use these for the research of phenomena in that other science field.*” (Barendsen & Tolboom, 2016). The curriculum does not provide further details about these objectives, instruction or assessment. In line with the Dutch tradition, this is left to educators and authors of teaching materials. The elaboration of these learning objectives, the development of teaching materials, assessment tools and teacher training courses are already taking place and we both participate in these endeavors and monitor the developments.

This study is a part of a larger research project on teaching Computational Science in the context of informatics in Dutch secondary education, investigating pedagogical aspects and teachers' pedagogical content knowledge (PCK) about modeling. (For clarity, in this paper the terms modeling, simulation modeling and computational science all refer to the learning objective computational science.) Following Magnusson et al. (Magnusson, Krajcik, & Borko, 1999), we distinguish four elements of content-specific pedagogy: (M1) goals and objectives, (M2) students' understanding and difficulties, (M3) instructional strategies, and (M4) assessment. Previously, we refined the CSTA definition of computational thinking

(CT) (Grgurina, Barendsen, Zwaneveld, van de Griff, & Stoker, 2013), made initial explorations of teachers' PCK (Grgurina, Barendsen, Zwaneveld, van Veen, & Stoker, 2014a; Grgurina, Barendsen, Zwaneveld, van Veen, & Stoker, 2014b) and of the computational modeling process (Grgurina, Barendsen, van Veen, Suhre, & Zwaneveld, 2015), obtained an operational description of the intended learning outcomes (ILO) of the learning objective Computational science — thus focusing on Magnusson's element M1, observed students working on modeling tasks — focusing on Magnusson's element M2, and established what data sources were suitable for assessment — Magnusson's element M4 (Grgurina, Barendsen, Zwaneveld, van Veen, & Suhre, 2016), and finally, investigated teachers' initial pedagogical content knowledge on modeling and simulation (Grgurina, Barendsen, Suhre, van Veen, & Zwaneveld, 2017). In our subsequent study, we focus on monitoring the levels of understanding in the learning outcomes of students engaging in modeling projects - Magnusson's element M4 - and address the following research question: *What are the characteristics of the assessment instrument for assessment of the intended learning outcome for computational science?* In this paper, we describe the design of this assessment instrument. The results of the entire study will be reported elsewhere.

Background and Related Work

Computational Thinking: Modeling

Formulating problems in a way that enables us to use a computer to solve them and representing data through abstractions such as models and simulations are integral parts of computational thinking (CT) (CSTA Computational Thinking Task Force, 2011). With the arrival of computers into schools, new venues are created to aid students' learning in various disciplines through the use of computer models (Blikstein & Wilensky, 2009; Van Overveld, Borghuis, & van Berkum, 2015). Wilensky argues, "Computational modeling has the potential to give students means of expressing and testing explanations of phenomena both in the natural and social worlds" (2014), as do Caspersen and Nowack (2013). Indeed, modeling plays a significant role in the development and learning of science (Justi & Gilbert, 2002) and informatics equips the students to actively engage in learning science by providing tools and techniques to engage in modeling, thus enabling them to provide meaning to the learning both of the discipline at hand (Gilbert, 2006) and informatics. In the Informatics curriculum, for the intended learning outcomes of the learning objective Computational science, in one of our previous studies we developed an operational description that describes the modeling cycle for simulation modeling through its elements purpose, research, abstraction, formulation, requirements/specification, implementation, verification/validation, experiment, analysis, and reflection (Grgurina et al., 2016).

Assessment

Brennan and Resnick focused on assessment of the development of CT during learning in informal settings and developed a CT framework distinguishing three dimensions: computational concepts describing the concepts designers employ as they program, namely "*sequences, loops, parallelism, events, conditionals, operators, and data*"; computational practices describing the practices designers develop as they program, namely "*being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing*", and computational perspectives describing the perspectives designers form about the world around them and about themselves, namely "*expressing, connecting and questioning*". (Brennan & Resnick, 2012). Zhong et al. brought these three dimensions of CT into the classroom when designing an assessment framework for elementary school students and they redefined them as follows: computational concepts as "*objects, instructions, sequences, loops, parallelism, events, conditionals, operators, and data*"; computational practices as "*planning and designing, abstracting and modeling, modularizing and reusing, iterative and optimizing, and testing and debugging*", and computational perspectives as "*creative and expressing, communicating and collaborating, and understanding and questioning*" (Zhong, Wang, Chen, & Li, 2016). Using this framework, Lye and Koh analyzed 27 intervention studies in K-12 aimed at the development of computational thinking and found that the majority focuses on computational concepts and only six on computational practices. In order to promote focus on computational practices and computational perspectives in a K-12 classroom, they suggest an instructional approach providing "*a constructionism-based problem-solving learning environment, with information processing, scaffolding and reflection*

activities.” (Lye & Koh, 2014) Since assessments can provide learning opportunities, Brennan and Resnick offer six suggestions for assessing computational thinking via programming, among others to make assessment useful to learners, to incorporate creating and examining artifacts, and to have the designer illuminate the whole process. (Brennan & Resnick, 2012).

These views are corroborated by the findings in our prior study on informatics teachers’ pedagogical content knowledge (PCK) of modeling and simulation, where we learned that the interviewed teachers mostly suggest hands-on approach to learning and that the preferred assessment form for most of them would be a practical assignment lasting several weeks, where student groups would construct models and use them to run simulations and conduct research while extensively documenting the whole process. At the same time, we observed a great diversity in the assessment criteria teachers mentioned, yet very few corresponding quality indicators used to judge to what extent these criteria are met (Grgurina et al., 2017).

In the eyes of the students, the assessment defines the actual curriculum, according to Biggs and Tang, who advocate a criterion-referenced system where the objectives are imbedded in the assessment tasks. In their *constructive alignment network*, the curriculum is stated in the form of clear intended learning objectives (ILO) specifying the required level of understanding, the teaching methods engage students in doing things nominated by the ILO’s and the assessment tasks address these ILO’s. The learning outcomes can be classified using the Structure of the Observed Learning Outcome (SOLO) which describe the learning progress through five levels of understanding. The first three levels — prestructural, unistructural and multistructural — are considered to be quantitative in the sense that prestructural indicates missing the point, unistructural means meeting only a part of the task and multistructural shows a further quantitative increase in what is grasped: “*knowing more*”. Relational, on the other hand, indicates a qualitative change indicating conceptual restructuring of the components — “*deepening understanding*”, and extended abstract takes the argument into a new dimension: (Biggs & Tang, 2011). Meerbaum-Salant et al. interpreted SOLO as five ordered categories:

- **Prestructural:** Mentioning or using unconnected and unorganized bits of information which make no sense.
- **Unistructural:** A local perspective – mainly one item or aspect is used or emphasized. Others are missed, and no significant connections are made.
- **Multistructural:** A multi-point perspective – several relevant items or aspects are used or acknowledged, but significant connections are missed and a whole picture is not yet formed.
- **Relational:** A holistic perspective – meta-connections are grasped. The significance of parts with respect to the whole is demonstrated and appreciated.
- **Extended abstract:** Generalization and transfer – the context is seen as one instance of a general case.

According to them, while the strength of the SOLO taxonomy lies in the fact that it offers a holistic, rather than a local perspective, “*using [it] for various types of activities, simultaneously, is not straightforward*”, so they combined the Bloom’s taxonomy and the three intermediate categories of the SOLO taxonomy in order to assess how novice programmers learned programming with Scratch (Meerbaum-Salant, Armoni, & Ben-Ari, 2013). Whalley et al. noted that previous research had indicated difficulties in mapping from student code to the SOLO taxonomy “*since the mapping process seems very context bound and question specific*”. To alleviate this problem, they developed a mapping framework where first, the salient elements are identified at syntactic level of the code; subsequently, basic replicable and discernible features such as redundancy, efficiency, generalizability and integration are abstracted from the code itself, and finally, SOLO mapping takes place to the five SOLO categories they suggest for code writing solutions (Whalley, Clear, Robbins, & Thompson, 2011).

The issue of assessing the learning of the students engaged in larger programming projects attracts attention as well. Casto and Fisler explored how to track program design skills through the entire CS1 course and suggest a multi-strand SOLO taxonomy, thus corroborating the idea that using SOLO taxonomy simultaneously for various types of activities is not straightforward. They suggest a multi-strand SOLO-taxonomy without the extended abstract level, since none of the students in their study reached that level (Castro & Fisler, 2017). A multi-strand SOLO taxonomy is in line with the idea that one assessment task might address several ILOs and vice versa, one ILO might be addressed by several assessment tasks (Biggs & Tang, 2011). Assignments for complex tasks encompassing diverse ILOs — such as going through a modeling cycle by formulating a problem, pinpointing the

research question, designing a model and using it to answer the research question — warrant the elaboration of criteria defining performance for each of the ILOs involved.

Assessment instrument

Based on these findings, we developed constructionist teaching material about agent-based modeling with NetLogo, meant for the informatics students in the 11th and 12th grades who are preferably no novice programmers but rather somewhat experienced, probably in other programming languages. The teaching material covers all the aspects of the ILO's of Computational science we identified earlier (Grgurina et al., 2016), and focuses not only on computational concepts such as programming to implement the model, but also on computational practices such as the validation of the model and computational perspectives such as formulating the research question to be answered through the use of the model. Together with this teaching material, we also developed an assessment instrument on which we focus here.

Following suggestions for the rubrics construction by Wolf and Stevens (2007), from the modeling cycle we first identified the criteria that defined performance as: stating the case and the research question, designing the model, implementation, validation, experiment, analysis, answering the research question, reflection, and additionally, logbooks. Subsequently, we designed an assessment instrument consisting of a practical assignment that provides several cases and research questions for students to choose from, a detailed description of the modeling process they need to engage in, and a corresponding rubric based on SOLO taxonomy with unequally weighted criteria defining performance. The description of SOLO categories was based on the interpretation by Meerbaum-Salant et al., stressing the progression from the local to the global perspective.

An example of the cases provided is the question whether sustainable human life is possible on Mars. The students are pointed to the websites of NASA and SpaceX to learn about the current state of affairs and subsequently have to explore whether, after the initial supplies and shelter were delivered, it would be possible to produce sufficient water, air and food to survive and thus whether it would be possible to found a sustainable human colony on Mars. Among other cases are the questions, what is better for traffic flow on a junction: a roundabout or traffic lights, and to investigate the optimal number and task division of bank counters as to minimize the waiting time of the customers with various needs. In line with our dedication to stimulate student engagement, the students are allowed to come up with their own research questions instead.

Assignment

The assignment consists of a number of questions the students need to answer in writing while designing their model and using it to answer their research question. After forming groups and choosing a case to model, the students answer the following questions:

Case and research question. Describe what you are going to model and with what purpose: (1) What do you know about this phenomenon? If need be, carry out the necessary research. (2) What part of your phenomenon would you like to build a model of? (3) What do you hope to observe from this model? (Questions 2 and 3 suggested by Wilensky & Rand (2015).)

Design the model. Design a model following the questions listed here. Describe the considerations and choices you make. (E.g., *“The sheep can reproduce. If two sheep meet, there is a chance of 20% that a new sheep will be breed. We decided not to take into account the gender of the sheep because that is not relevant in this case.”*) (1) What are the principal types of agents involved in this phenomenon? (2) In what kind of environment do these agents operate? Are there environmental agents? (3) What properties do these agents have (describe by agent type)? (4) What actions (or behaviors) can these agents take (describe by agent type)? (5) How do these agents interact with this environment or each other? (6) If you had to define the phenomenon as discrete time steps, what events would occur in each time step, and in what order? (All questions suggested by Wilensky & Rand (2015).)

Implement the model. Implement the model in NetLogo. Write your code in small chunks and keep testing!

Validate the model. (1) Microvalidation: to what extent does the agents' behavior resemble the observations of the phenomenon in reality? If the behaviors are (somewhat) dissimilar, is this variation relevant to your research question? (2) Macrovalidation: to what extent does the behavior of the system as a whole resemble the observations of the phenomenon in reality? If the behavior is (somewhat) dissimilar, is this variation relevant to your research question?

Experiment, analysis and conclusion. Use the model to answer your research question: (1) Describe the experiment in detail. If you use Behavior Space, report the number of experiments conducted and the parameters used. (2) Report the findings in an appropriate manner (e.g., a narrative, a table, a graph, etc.) (3) Analyze the results. (4) Answer the research question.

Reflection. Reflect on your modeling process: (1) What went well and what could be better? (2) Did you make any assumptions which, in retrospect, you would like to reconsider? (3) Are there any aspects of your model which you would like to change? Are there any aspects of your model (agents or behavior) you decided not to include in your model while now you believe they do need to be included? Make a wish list of aspect of your model that need to be added, removed or changed in the next version of the model.

In addition, the students were asked to keep a logbook recording all their activities, problems, successes and dead ends they encountered; possible explanations for problems and successes, and finally, lessons learned.

Grading Rubrics

After we identified the criteria that defined performance, we created performance descriptions (Wolf & Stevens, 2007) to describe the appropriate level of understanding for intended learning outcomes (Biggs & Tang, 2011). Here we quote some of these descriptions:

Case and research question

- **Prestructural:** (1) Nothing or simplistic idea of the phenomenon. Performed no research. (2) Nothing, or a few non-specific remarks but missing the point (3) Research question not clear
- **Unistructural:** (1) Some general description. Performed no research or only limited to isolated aspects of the phenomenon (2) Few isolated aspects of the phenomenon identified. (3) Identified the question from a local perspective.
- **Multistructural:** (1) Performed some research. Able to name more relevant aspects of the phenomenon, but mentions no relations among these aspects (2) Described what (part of the) phenomenon is being modeled. (3) Described the question from a multi-point perspective.
- **Relational:** (1) Performed research. Complete idea of the phenomenon. Able to name relevant aspects of the phenomenon, have insight into relations among these aspects (2) Described what (part of the) phenomenon is being modeled. (3) The research question clear and predicts possible outcomes.
- **Extended abstract:** (1) Additionally, described the relation of this phenomenon to other phenomena in the world and/or conceptualized this phenomenon so as to be able to use it other contexts restricted and its relevance explained. Stated its relevance for other phenomena. (2) Additionally, theorize about possible generalization of the model or transfer into a different context. (3) Additionally, theorize about possible generalization or transfer into a different context.

Design the model and implement it

- **Prestructural:** No agents mentioned.
- **Unistructural:** A few agents and actions identified.
- **Multistructural:** Several agents and actions described.
- **Relational:** Agents, actions and interactions correct and substantiated. Their contribution to the whole acknowledged.
- **Extended abstract:** Additionally, generalize or hypothesize about similar models in different contexts or extend the model beyond the minimal requirements.

Validate the model

- **Prestructural:** Nothing. No working program.
- **Unistructural:** Identified some resemblances and differences between the model and reality. Relevance for the research question not clear.
- **Multistructural:** Described resemblances and differences between the model and reality. Relevance of the differences for the research question not clear.
- **Relational:** Resemblances and differences between the model and reality described. Analyzed and explained their relevance for the research question.
- **Extended abstract:** Additionally, hypothesized over model adjustments to improve its validity for a more general purpose.

Results and Further Research Direction

In this paper, we described the design of our assessment instrument for the assessment of the intended learning outcomes for Computational Science consisting of a practical assignment covering the ILO's defining Computational Science and an accompanying rubric based on SOLO taxonomy that describes the levels of understanding. During the design process, we faced many challenges due to the fact that some ILO's of modeling are at the core of informatics (e.g. implementation of the model), while others are not often seen in an informatics classroom (e.g. experiment). Even for implementation, which comes down to programming, it was not easy to find related work addressing assessment of programming at just the right level of granularity. The same holds true for validation: while there is plentiful literature on validation of computational models, to our best knowledge there is none focusing on the assessment of validation in a formal learning setting.

So far, several teachers used our teaching material and assessment instrument to teach Computational Science in the Informatics class of the 11th and 12th grade of pre-university education (VWO) in the Netherlands. We are collecting and analyzing feedback from them and their students in order to aid the on-going project of development of teaching materials and assessment instruments. Specifically, the current version of the assessment instrument will be analyzed to establish its reliability, validity and objectivity, and in particular, it will be scrutinized in relation to the descriptions and attainability of currently proposed levels of understanding as specified in the rubrics.

Acknowledgments

This work is supported by the The Netherlands Organisation for Scientific Research grant nr. 023.002.138.

References

- Barendsen, E., & Tolboom, J. (2016). Advisory report (intended) curriculum for informatics for upper secondary education. Enschede: SLO.
- Biggs, J., & Tang, C. (2011). Teaching for quality learning at university. McGraw-Hill International.
- Blikstein, P., & Wilensky, U. (2009). An atom is known by the company it keeps: Content, representation and pedagogy within the epistemic revolution of the complexity sciences.
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada,
- Caspersen, M. E., & Nowack, P. (2013). Model-Based thinking & practice.
- Castro, F. E. V., & Fisler, K. (2017). Designing a multi-faceted SOLO taxonomy to track program design skills through an entire course. Proceedings of the 17th Koli Calling Conference on Computing Education Research, pp. 10-19.
- CSTA Computational Thinking Task Force. (2011). Operational definition of computational Thinking for K-12 education. Retrieved 10/16, 2013, from <http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf>

- Gilbert, J. K. (2006). On the nature of "context" in chemical education. *International Journal of Science Education*, 28(9), 957-976.
- Grgurina, N., Barendsen, E., Suhre, C., van Veen, K., & Zwaneveld, B. (2017). Investigating informatics teachers' initial pedagogical content knowledge on modeling and simulation. *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pp. 65-76.
- Grgurina, N., Barendsen, E., van Veen, K., Suhre, C., & Zwaneveld, B. (2015). Exploring students' computational thinking skills in modeling and simulation projects: A pilot study. *Proceedings of the Workshop in Primary and Secondary Computing Education*, pp. 65-68.
- Grgurina, N., Barendsen, E., Zwaneveld, B., van de Grift, W., & Stoker, I. (2013). Computational thinking skills in Dutch secondary education. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, pp. 31-32.
- Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Stoker, I. (2014a). Computational thinking skills in Dutch secondary education: Exploring pedagogical content knowledge. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, pp. 173-174.
- Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Stoker, I. (2014b). Computational thinking skills in Dutch secondary education: Exploring teacher's perspective. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, pp. 124-125.
- Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Suhre, C. (2016). Defining and observing modeling and simulation in informatics. *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pp. 130-141.
- Justi, R. S., & Gilbert, J. K. (2002). Science teachers' knowledge about and attitudes towards the use of models and modelling in learning science. *International Journal of Science Education*, 24(12), 1273-1292.
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61.
- Magnusson, S., Krajcik, J., & Borko, H. (1999). Nature, sources, and development of pedagogical content knowledge for science teaching. In J. Gess-Newsome, & N. G. Lederman (Eds.), *Examining pedagogical content knowledge* (pp. 95-132) Kluwer.
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239-264.
- Van Overveld, K., Borghuis, T., & van Berkum, E. (2015). From problems to numbers and back. Lecture notes to 'A discipline-neutral introduction to mathematical modelling'. Eindhoven: Eindhoven University of Technology.
- Whalley, J., Clear, T., Robbins, P., & Thompson, E. (2011). Salient elements in novice solutions to code writing problems. *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114*, pp. 37-46.
- Wilensky, U. (2014). Computational thinking through modeling and simulation. Whitepaper Presented at the Summit on Future Directions in Computer Education. Orlando, FL. [Http://www.Stanford.Edu/~coopers/2013Summit/WilenskyUriNorthwesternREV.Pdf](http://www.Stanford.Edu/~coopers/2013Summit/WilenskyUriNorthwesternREV.Pdf).
- Wilensky, U., & Rand, W. (2015). *An introduction to agent-based modeling: Modeling natural, social, and engineered complex systems with NetLogo* MIT Press.
- Wolf, K., & Stevens, E. (2007). The role of rubrics in advancing and assessing student learning. *The Journal of Effective Teaching*, 7(1), 3-14.
- Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, 53(4), 562-590.

Constructionist Experiences in Teacher Professional Development: A Tale of Five Years

Daniel Hickmott, daniel.hickmott@uon.edu.au

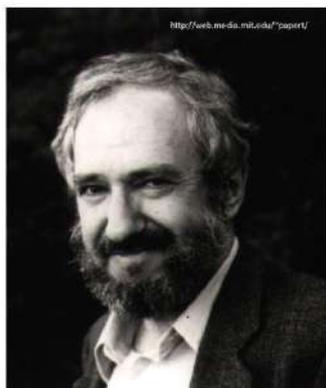
School of Education, The University of Newcastle, Australia

Elena Prieto-Rodriguez, elena.prieto@newcastle.edu.au

School of Education, The University of Newcastle, Australia

Abstract

Computational thinking and coding have recently become compulsory elements in the Australian K-8 curriculum that should be taught using 'authentic learning challenges' (ACARA, 2018a). However, very few teachers, particularly in the primary school setting, have been schooled on computational thinking or coding and rarely possess pedagogies to teach them authentically. A range of professional development opportunities are currently being offered to impart this knowledge, both for content and pedagogy. In this paper, we provide an account of the evolution we have experienced when designing and improving professional development workshops for teachers in coding and computational thinking. We reflect on our challenges and successes, and attest that it was only after 'discovering' Constructionism in late 2015 that we have been able to prepare activities that truly emulate the authentic learning experiences that teachers are required to use in their classrooms.



"I am convinced that the best learning takes place when the learner takes charge."

— Seymour Papert

© Steve Wheeler, University of Plymouth, 2015



Learning from pioneers: the influence of Seymour Papert on our professional development

Keywords

teacher professional development; constructionism; computing; programming

Introduction

Teacher preparation for new digital technologies curricula across the world has become a topic of interest in recent years. Globally, coding and computational thinking appear almost daily in the news, and there seems to be a strong sense within industry that they are skills essential for the workforce of the future (Chalmers & Quigley, 2017; Pappano, 2017).

In September 2015, the national Australian Curriculum was officially endorsed. This curriculum includes the new Digital Technologies (DT) subject, within the Technologies learning area, which is focussed on the teaching of "computational thinking and information systems to define, design and implement digital solutions." (ACARA, 2018b). This subject will be mandatory for all Australian students from Kindergarten to Year 8 and available as an elective for Year 9 and 10 students.

Introducing a subject that has a focus on computing reflects a current global trend in K-12 Information and Communications Technology (ICT) education. Several countries have begun to change ICT curricula to focus less on the learning of particular software packages and to focus more on the teaching of skills core to computing, such as computational thinking and programming (Webb et al., 2016). It is argued that teaching students these skills will allow them to become creators, rather than just consumers of, digital technologies (Bower & Falkner, 2015).

Australian educational stakeholders have concerns about the feasibility of implementing the new DT subject. Falkner, Vivian, and Falkner (2014) revealed that *“a consultation with Industry, Community and Education stakeholders in Australia showed that 55% of respondents had concerns about manageability of the implementation of the curriculum, while 45% of respondents did not think that its learning objectives were realistic”* (p. 6). One of the stakeholders’ main concerns is whether teachers are adequately prepared to teach computational thinking and programming because, unless they have qualifications in computing, they are unlikely to have had formal experience learning these skills. This is particularly true for primary school teachers, who don’t usually have the option to complete a technology major and who are often generalist teachers (Vivian, Falkner, & Falkner, 2014).

Several teacher Professional Development (PD) initiatives have been developed by universities and private organisations to address concerns related to teacher preparation for the Australian DT subject (Commonwealth of Australia, 2016). These initiatives include Massive Online Open Courses (MOOCs), such as those run by the University of Adelaide’s Computer Science Education Research (CSER) group (Vivian et al., 2014), and face-to-face workshops (Prieto-Rodriguez & Berretta, 2014).

Similar PD initiatives have also been developed in other parts of the world. In Europe, for example, a transnational initiative, TACCLE, combines learning through a website, news related to coding in schools, and the sharing of resource reviews by teachers (García-Peñalvo, 2016). In England, researchers, who were working with the Computing At School (CAS) organisation, developed a model of PD that is holistic and sustained for teachers implementing the Computing curriculum (Sentance, Humphreys, & Dorling, 2014). In the United States (US), computing PD research has mainly been conducted by computer science academics (Menekse, 2015). Menekse (2015) systematically reviewed US articles on computing PD published between 2004 and 2014. One of the aims of this review was to evaluate the effectiveness of these programs, according to five factors that the author determined to be indicators of effective PD from a literature review. The results of the review indicated that many of the face-to-face PD opportunities did not have these factors of effective PD present in them. Menekse suggested that one of the reasons why these factors were not present could be that these PD opportunities are often conducted by computer science faculty academics, who may have limited collaboration with other education researchers and practitioners.

In this paper, we reflect on our experience designing and improving PD for teachers in computing. We describe the learning process that we followed, as computer science academics new to computing education, trying to implement relevant approaches to PD that are valued by Australian teachers and useful in preparing them for the DT curriculum. Looking back, it is clear to us that it was after we started looking at our work through the lens of Constructionism, that we were able to make meaningful changes to our PD. The learning process we underwent to identify the changes to be made, highlighted the need to explore certain forms of PD that are not prevalent in recent computing education literature. We believe these forms of PD are not only useful for preparing existing teachers, but also for assisting in the development of pre-service teaching courses in computing. These courses should be designed so that they are relevant to current practice, particularly for primary school teachers.

Our Journey

Our team has been conducting computing PD workshops over the last five years. These workshops have been held at our university, conducted face-to-face, and have ran over two or three days. In 2013 and 2014, these were only available for high school teachers but in 2015 we began to include primary school teachers as well. Since their inception, the three overarching aims of the workshops have been to: 1) communicate the applicability and importance of computing to a wide range of research areas and careers, 2) provide examples of activities that address DT concepts and 3) provide resources for

teachers to use in their classrooms. These workshops have involved computational thinking and programming exercises with step-by-step instructions, collaborative problem-solving exercises, and presentations by academics and industry representatives.

The design and implementation of the workshops has evolved each year as a result of participants' feedback, which has been collected through validated surveys (Prieto-Rodriguez & Berretta, 2014), review of the literature, and reflection. Initially, the changes were a direct consequence of the feedback received through the surveys, particularly within the open-ended questions. Later, since the 2016 workshops, changes were made to incorporate more constructionist activities.

When comparing the responses to the items common to all our surveys, it became clear that, as we included Constructionist approaches to the design of the workshops, the satisfaction with the content presented and its applicability to classroom practices increased. In particular the major changes introduced in 2016, as seen in Figure 1, seemed to increase the 'sense of community', 'inspiration to improve teaching' and 'use of applications of computer science', as we included more collaborative and hands-on activities (note: the asterisk next to the year indicates a primary school focus, all others are secondary school).



Figure 1. Mean responses to selected survey questions in the workshops delivered from 2013 to 2016. Items in a 4-point Likert scale with Strongly Disagree coded as 1 and Strongly Agree coded as 4.

Before Constructionism

The first PD workshop took place in 2013 and ran over three days. This workshop included a day of lectures presented by researchers from a variety of disciplines who incorporate computing in their work. The other two days involved a combination of presentations and some hands-on activities. Analysis of survey data of participants in 2013 showed that there were certain misconceptions with regards to the nature of computer science that we were able to unmask (XXX, 2014). In terms of the pedagogical approach utilised, feedback from the participants suggested that to improve future workshops we should include "...more hands on with Robotics..." and "more hands on activities that we can take back and use in school". It was clear that many of the teachers preferred to build their own knowledge rather than have it delivered to them in an instructionist manner.

In 2014, we increased the number of hands-on activities as a result of this feedback and reduced the workshop length to two days. Feedback from 2014 indicated that teachers wanted more ideas for projects that they could use with their students. One teacher suggested that we include "Realistic projects for school students e.g. project ideas when using C++" in future workshops. Reflecting back on these comments, we see now that teachers were looking for us to help them provide context and purpose to their students.

In 2015, we took a more systematic approach to the design of the PD and explored the use of student-centred learning approaches in the two two-day workshops that we ran that year. The hands-on activities included in the two previous years were written as step-by-step guides with explicit instruction on how to complete the activity. We adopted a framework for instruction based on constructivist principles to help us design and run collaborative problem-solving exercises. The Five Cs (5Cs)

framework (Tom, 2015) is built on 5 constructs identified in the literature: *Consistency, Collaboration, Cognition, Conception, and Creativity*. The sessions in the two workshops were thus purposely designed to investigate participant responses to different pedagogies.

We used self-reported measures to compare perceptions about the activities designed with the 5Cs Framework against those that were step-by-step activities. Preliminary results were presented at the Constructionism 2016 conference (Prieto-Rodriguez & Hickmott, 2016). Our analysis showed that while concepts were understood consistently for all constructs, the Scratch session was the one where the teaching methods were preferred by participants and higher order thinking was more present. This session was the only one not involving collaboration and it was not designed using the 5Cs Framework.

The open-ended responses to the survey prompted us to reconsider what was valued by teachers and made us come to the realisation that higher order thinking and time to 'tinker' should be a focus of our PD. This can be illustrated with one of the participant's survey responses stating that time to 'tinker' would have been desirable: "It would have been fantastic to have time to do a visual graphing experiment similar to the social media graphing presented on Tuesday morning. Use responses from the group to carry out an activity that could be done in the classroom".

After Constructionism

In 2016, inspired by our attendance to the *Constructionism in Action* conference in Bangkok, we conducted two two-day workshops that incorporated Constructionist principles. The results from the 2016 surveys seemed to indicate that the inclusion of Constructionist approaches enhanced the satisfaction with all aspects of the workshop (see Figure). The workshops from that point thereafter, not only aimed to be more student-centred and include more hands-on experiences, but were also inspired by Constructionism.

As suggested by participants in the surveys, we had begun including an increasing number of hands-on activities in the workshops. However, it became apparent to us that it was important to incorporate hands-on activities that involved open-ended problem solving, in which teachers construct their knowledge in context, rather than just having hands-on activities that involved explicit step-by-step instruction.

To incorporate these Constructionist activities, we considered suggestions outlined by several Constructionist PD researchers and practitioners. For example, Martinez and Stager (2013) argue that PD is often "too meta" (p. 200), and suggest that teacher educators should provide PD where teachers experience learning from a student's perspective. Martinez and Stager designed and implemented *Constructing Modern Knowledge*, enabling teachers to spend four days of uninterrupted time working collaboratively on a project of their own creation. Brennan provides a similar argument concerning PD learning environments, stating, "teachers should have learning experiences that are comparable to their students' learning experiences, situated within a supportive community of fellow teachers" (p.293).

Brennan (2015) developed and implemented *ScratchEd*, a form of PD for computing that incorporated Constructionist principles. The main intent of *ScratchEd* is to support the use of technology for creating meaningful projects rather than focussing on the use of specific technologies. Like Martinez and Stager's *Constructing Modern Knowledge* workshops, *ScratchEd* is designed to involve experiences for teachers that comparable to their students' classroom experiences. Brennan, however, recognises that providing fully Constructionist learning environments can be at odds with K-12 mainstream education. She calls the difficulty of providing PD aligned with "the lived reality of K-12 education" (p.295) while preserving the ideal of a Constructionist environment, the "tension between the actual and the aspirational" (p.295). She also identifies four other tensions, which have guided and given focus to our reflections about the PD we provide.

We agreed with the sentiment that PD is often "too meta" (Martinez & Stager, 2013), and upon reflection, recognised that our previous PD opportunities had suffered from being too instructionist at times. However, like Brennan, when we introduced Constructionist learning into our PD, we had to negotiate some tensions.

In 2016, we ran two two-day workshops: a workshop for primary school (K-6) teachers and a workshop for high school (Years 7-12) teachers. These workshops involved activities and presentations that were aligned with the curriculum outcomes for the relevant school level. For example, in the primary school workshop, teachers took part in an activity where they used Scratch to learn about *visual programming*, which is a concept in the K-6 Australian Digital Technologies curriculum. In the 2016 workshops, in an attempt to move towards more Constructionist learning, we began to include some activities where teachers could choose their own direction for learning. In the high school workshop, for example, teachers could choose between learning about teaching Data Science with R or being introduced to general-purpose programming with Sonic Pi. Also, in the primary school workshop, we included extra activities for teachers who were already proficient at *visual programming*. Additionally, in the primary school workshop, we included a collaborative lesson planning activity, in which teachers worked together to construct a plan for introducing computing into their classes. However, we could not include longer self-directed learning, and had to negotiate with pressures such as the limited time available to teachers to attend the workshops, or the number of PD staff offering support to teachers working on individual projects.

In 2017, we revisited Brennan's tensions (2015) as a lens to improve the PD. The analysis of these tensions and our own reflections on how we negotiated them, were instrumental to envisioning the changes we made to the workshops in 2017. One of the main changes introduced that year, which responded to our reflection on the *tension between the actual and the aspirational* identified by Brennan (2015), was the introduction of workshops addressing specific curriculum outcomes.

In light of this, we ran two new specific workshops, one for primary teachers and one for high school mathematics teachers. These two workshops aligned entirely with areas of the curriculum and were designed to use computing as an exploratory tool for teachers (and subsequently their students) with mathematics. The first of these workshops utilised one of the modules produced for the *ScratchMaths* project (Benton, Hoyles, Kalas, & Noss, 2017). We mapped the content of the module to the Australian curriculum. The second workshop targeted a new area of the Year 12 curriculum, Minimum Spanning Tree algorithms. In this workshop, teachers learned and used *Edgy*, an adaptation of *Snap!* for graph theoretical explorations (Cox, Bird, & Meyer, 2017). Teachers coded Kruskal and Prim's algorithms with varying degrees of guidance after reviewing other content, such as graph theory and data structures, necessary for the completion of the task.

We also ran two more two-day workshops, similar to the ones in 2016 but more general in scope, and deliberately targeting outcomes, such as *critical and creative thinking*, from the General Capabilities of the Australian National Curriculum (ACARA, 2018b). These two workshops specifically addressed issues of differentiation of learning and learners - *the tension between novice and expert* identified by Brennan (2015).

In workshops prior to 2017, we had assumed that the teachers attending were novice to computing, and we felt that most of the PD content had to be heavily scaffolded. However, teachers with expertise in computing also attended the workshops. To better support these expert teachers, thus helping us negotiate this last tension, we interviewed a local teacher who had participated in our workshops in 2013, and had been invited to present in subsequent years' workshops. This teacher is a high school DT teacher with an academic and professional background in computing. She is experienced in teaching computing, regularly presents at conferences and assists in training her colleagues and thus fits in the category of *Master Teacher* in the classification devised by Sentance et al. (2014). Master Teachers are experienced educators chosen by CAS in England to organise and run face-to-face workshops with teachers in their local community. We asked questions to ascertain the extent to which her approach to teaching and learning was a constructionist one:

Question: "Do you learn using YouTube, read instructions, or tinker?"

Answer: "Yes! First place I go is YouTube and tinkering is the best part of learning and teaching."

Question: "Do you use learning resources created specifically for teachers? If so, what were the most helpful aspects of these sort of experiences?"

Answer: “Sometimes but these resources often lack room for creativity or make teachers into dependant zombies instead of content creators. However these can be great for networking and idea generation.”

We also asked the teacher to provide insights into their learning for the classroom. We found that tinkering is also a part of the process to develop their pedagogy.

Question: “Do you find that the activities you find need to be modified to suit your needs? If so, which type of activities and how do you modify them?”

Answer: “Most often, yes. Usually tech PD or resources are aimed at beginners so I often extend or find additional bits to meet my needs or particular learning requirements for my students.”

Her responses to these questions quite clearly indicate that her approach to both learning and teaching is a Constructionist one, and that she is at a level where step by step instruction is no longer necessary. One of the foci of our future PD endeavours will be to incorporate more differentiation, which would allow expert teachers to learn during our PD while respecting that other teachers need more guidance.

Discussion and Future Work

The recently endorsed Australian National Curriculum includes General Capabilities (such as *critical and creative thinking*), which teachers are expected to assess and report across all subject areas. In addition, this curriculum includes computational thinking and programming as part of the new DT subject that should be taught using ‘authentic learning challenges’ (ACARA, 2018a). These changes in the Australian curriculum provide an ideal opportunity for teachers to include Constructionist activities in K-12, which involve computing and open-ended problem solving. As argued by Angeli et al. (2016), there are many opportunities for K-12 educators to include authentic learning challenges that incorporate computing and also address outcomes from diverse content areas. In Australia, these authentic learning challenges could be particularly beneficial to K-6 teachers, who often teach multiple content areas, as they could target outcomes from a variety of subjects and also address the General Capabilities.

Due to the large number of teachers that need to be upskilled in computing, there has been a focus on creating computing PD that is scalable. An example of PD designed to be scalable that has been very successful, are the MOOCs designed by CSER. These MOOCs have reached many teachers, both nationally and internationally (Falkner, Vivian, Falkner, & Williams, 2017; Vivian et al., 2014), and CSER received government funding for a national rollout of their initiative, focussing on teachers in remote and disadvantaged areas (Birmingham, 2016).

There are many advantages of providing PD through online courses. Online courses such as CSER’s MOOCs, are useful scalable solutions, as teachers can access them on their own time and at their own pace, and they are generally free of cost. However, as Brennan (2015) argues, it can be difficult to provide Constructionist learning experiences solely through an online course. As PD providers who have been influenced by Constructionist ideas, we believe that the inclusion of authentic learning challenges in K-12 should be encouraged. Thus, we reflect that our PD should include face-to-face components and authentic learning challenges to prepare teachers effectively. This reflection has led us to make further changes to our PD which we plan to implement in 2018.

In 2018 we will run three PD programs. The first program will be a two-day workshop for ‘novices’, with a strong emphasis on the Australian General Capabilities and will incorporate authentic tasks that we will develop with the teachers. Our second PD program will focus on expert teachers who are interested in running professional learning events for other teachers at their school or in their professional learning communities. Like the first workshop, it will involve hands-on activities where teachers will familiarise themselves with the content to be delivered, but they will also learn to run workshops for colleagues. We will facilitate a session with the aim of teaching about the logistics of running this type of event. We believe that this scalable model can complement online learning platforms whilst providing opportunities for Constructionist learning.

Our third PD program is focussed on sustained engagement and will run over six months with primary school teachers. This program will incorporate a greater depth of Constructionist experiences for

teachers. Finding ways to provide such PD is the main aim of our future research, and constitutes the core research of Author 1's PhD studies. As well as improving PD, these studies could inform the design of learning experiences for pre-service teachers in computing, which could help address pipeline issues (Yadav, Sands, Good, & Lishinki, 2018).

We believe that the insight gained from designing, reflecting on, and researching PD has not only improved our PD programs, but also has the potential to create Constructionist learning opportunities for pre-service teachers that will ensure that they are able to include authentic learning challenges in their future teaching.

Acknowledgements

The authors would like to thank Google Inc. for funding the CS4HS workshops held at our institution. We would also like to thank the university academics who presented talks and gave laboratory tours during the workshops, as well as the administrative staff of our institution.

References

- ACARA. (2018a). Digital Technologies curriculum rationale. Retrieved from <https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/rationale/>
- ACARA. (2018b). F-10 Curriculum - General Capabilities. Retrieved from <https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/>
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: implications for teacher knowledge. *Journal of Educational Technology & Society*, 19(3), 47.
- Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging Primary Programming and Mathematics: some findings of design research in England. *Digital Experiences in Mathematics Education*.
- Birmingham, S. S. (2016). Closing the 'digital divide' for disadvantaged students [Press release]. Retrieved from <http://www.senatorbirmingham.com.au/Latest-News/ID/2928/Closing-the-digital-divide-for-disadvantaged-students>
- Bower, M., & Falkner, K. (2015). *Computational Thinking , the Notional Machine , Pre-service Teachers , and Research Opportunities*. Paper presented at the Australasian Computing Education Conference, Sydney, Australia.
- Brennan, K. (2015). Beyond Technocentrism: Supporting Constructionism in the Classroom. *Constructivist Foundations*, 10(3), 289-296.
- Chalmers, J., & Quigley, M. (2017, 13 October). Twenty ideas for our schools and politicians in the new machine age. *The Sydney Morning Herald*. Retrieved from <https://www.smh.com.au/technology/twenty-ideas-for-our-schools-and-politicians-in-the-new-machine-age-20171012-gyzfm7.html>
- Commonwealth of Australia. (2016). *STEM Programme Index*. Retrieved from <http://www.chiefscientist.gov.au/2016/01/spi-2016-stem-programme-index-2016-2/>
- Cox, R., Bird, S., & Meyer, B. (2017). *Teaching Computer Science in the Victorian Certificate of Education: A Pilot Study*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Falkner, K., Vivian, R., & Falkner, N. (2014). *The Australian Digital Technologies Curriculum: Challenge and Opportunity*. Paper presented at the Australasian Computing Education, Auckland, New Zealand.
- Falkner, K., Vivian, R., Falkner, N., & Williams, S.-A. (2017). *Reflecting on Three Offerings of a Community-Centric MOOC for K-6 Computer Science Teachers*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, Washington, USA.

García-Peñalvo, F. J. (2016). *A brief introduction to TACCLE 3—coding European project*. Paper presented at the International Symposium on Computers in Education (SIIE), Salamanca, Spain.

Martinez, S. L., & Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom: Constructing modern knowledge* press Torrance, CA.

Menekse, M. (2015). Computer science teacher professional development in the United States: a review of studies published between 2004 and 2014. *Computer Science Education*, 25(4), 325-350.

Pappano, L. (2017, 4 April). Learning to Think Like a Computer. *The New York Times*. Retrieved from <https://www.nytimes.com/2017/04/04/education/edlife/teaching-students-computer-code.html>

Prieto-Rodriguez, E., & Berretta, R. (2014). *Digital technology teachers' perceptions of computer science: It is not all about programming*. Paper presented at the Frontiers in Education Conference (FIE2014), Madrid, Spain.

Prieto-Rodriguez, E., & Hickmott, D. (2016). *Preparing teachers for the Digital Technologies curriculum: preliminary results of a pilot study*. Paper presented at the Constructionism 2016 Conference, Bangkok, Thailand.

Sentance, S., Humphreys, S., & Dorling, M. (2014). *The network of teaching excellence in computer science and master teachers*. Paper presented at the Proceedings of the 9th Workshop in Primary and Secondary Computing Education, Berlin, Germany.

Tom, M. (2015). Five C Framework : A student-centered approach for teaching programming courses to students with diverse disciplinary background. *Journal of Learning Design*, 8, 21-37.

Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology*, 22.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2016). Computer science in K-12 school curricula of the 21st century: Why, what and when? *Education and Information Technologies*, 1-24.

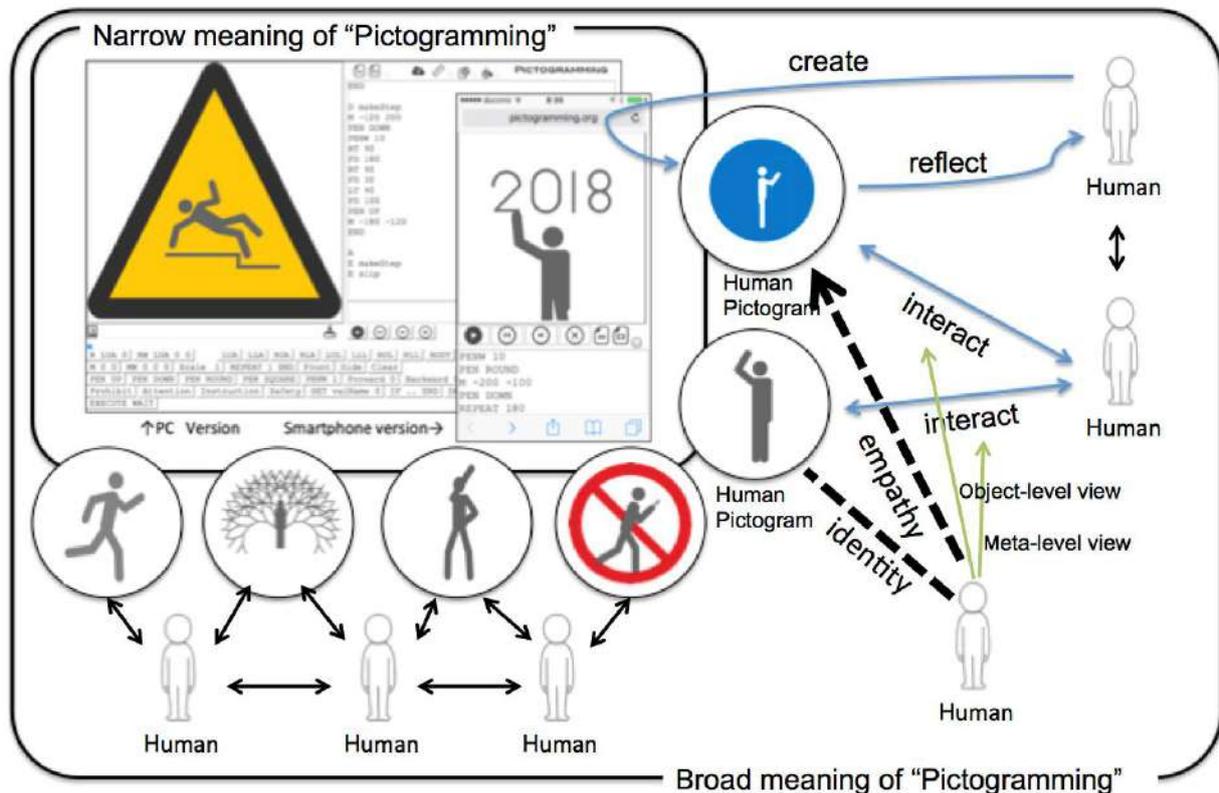
Yadav, A., Sands, P., Good, J., & Lishinki, A. (2018). Computer Science and Computational Thinking in the Curriculum: Research and Practice. In J. Voogt, G. Knezek, R. Christensen, & K.-W. Lai (Eds.), *Handbook of Information Technology in Primary and Secondary Education* (pp. 1-18). Cham: Springer International Publishing.

Pictogramming: Learning Environment Using Human Pictograms Based on Constructionism

Kazunari Ito, kaz@si.aoyama.ac.jp
Aoyama Gakuin University, Japan

Abstract

We have been developing a pictogram authoring tool “Pictogramming.” Pictogramming is based on two words: “pictogram” and “programming.” The basis of this application is the use of a human-shaped pictogram (i.e., a “human pictogram”). Pictograms are an abstract representation of a person or an object. Creating a posture of a human pictogram is strongly associated with the embodied knowledge. And the works with free themes are created with one's ego syntonic and are firmly and positively based on one's culture. A useful pictogram is made by strongly connecting with one's culture. That is, constructing human pictograms are very familiar with syntonic learning insisted on by Papert. We can expand this concept beyond the limits of the proposed application (we call this the “narrow meaning of Pictogramming”) to any activities that correspond to human pictograms, which is termed the “broad meaning of Pictogramming.”



Narrow and broad meaning of Pictogramming

In this paper, we first explain the function of the application and discuss the broad meaning of Pictogramming from the viewpoint of constructionism.

Keywords

Logo; human pictogram; syntonic learning; duplication of viewpoint, educational environment

Introduction

A pictogram is a graphical symbol that is used to understand a semantic concept based on the meaning of its shape [Ota 1987]. Pictograms have been studied and used in various fields such as counseling, safety, and facilities management. Especially there is large amount of research related to the use pictograms as alternative and augmentative communication (AAC) [Martínez-Santiago 2016].

Pictograms are designed to provide information regarding a human's action or status. And especially human shaped pictograms are widely shown in the various international standard pictograms set. So, the appendix of ISO 3864 provides guidelines for the depiction of a human-shaped pictogram itself (i.e., a "human pictogram").

Various programming languages have been developed for learning purposes. Logo is multi purpose programming language, for example, for AI and logical programming. And Logo is widely used for children to help them learn various mathematical concepts by operating a turtle robot/character on the screen [Seymour 1977]. Papert developed Logo and called this method syntonic learning; he argued in favor of its importance as a learning method [Seymour 1980]. Numerous extensions based on Logo have been developed. Laubomir developed EasyLogo [Salanci 2010], which adopts a grid for simplification and has an easy-to-use environment. StarLogo [Resnick 1996] can handle more than one turtle and can simulate a multi-agent system.

Scratch is a visual programming environment that was developed by the MIT Media Laboratory [Resnick 2009]. Users program in Scratch by dropping blocks of code. Scratch is suitable for use as an introductory programming course because no specific knowledge of syntax is needed and syntax errors cannot occur. In addition, Alice allows novice programmers to generate virtual worlds by controlling three-dimensional (3D) objects [Cooper 2003]. The users operate a cat in Scratch, a turtle in Logo (sometimes represented as a simple triangle), or some 3D objects in Alice; thus, the working style of novice students may influence their learning techniques. The main character shown in the initial stage of the programming language is a highly important parameter with which one can understand the concept of the programming.

So, unprecedented effectiveness may be achieved if a human pictogram is used as the main character in the programming learning environment. Indeed, virtual humans are used to enhance the learning environment [Halan 2012].

In this study, we developed a prototype programming learning environment called "Pictogramming" [Ito 2018]. Pictogramming is based on two words: "pictogram" and "programming." We evaluated and analyzed the effects and characteristics of Pictogramming from the viewpoint of constructionism.

Pictogramming—Learning Programming Application using a Human Pictogram (Narrow meaning of Pictogramming)

Overview

The application was implemented using HTML5, CSS, and JavaScript. The JavaScript library "processing.js" was used to execute the Processing format program, which is to display the human pictogram. The application is compatible with typical browsers and other applications do not need to be installed. Figure 1 shows a screenshot of when the application is accessed using a PC or smartphone browser. This application can be accessed at <http://pictogramming.org/>.

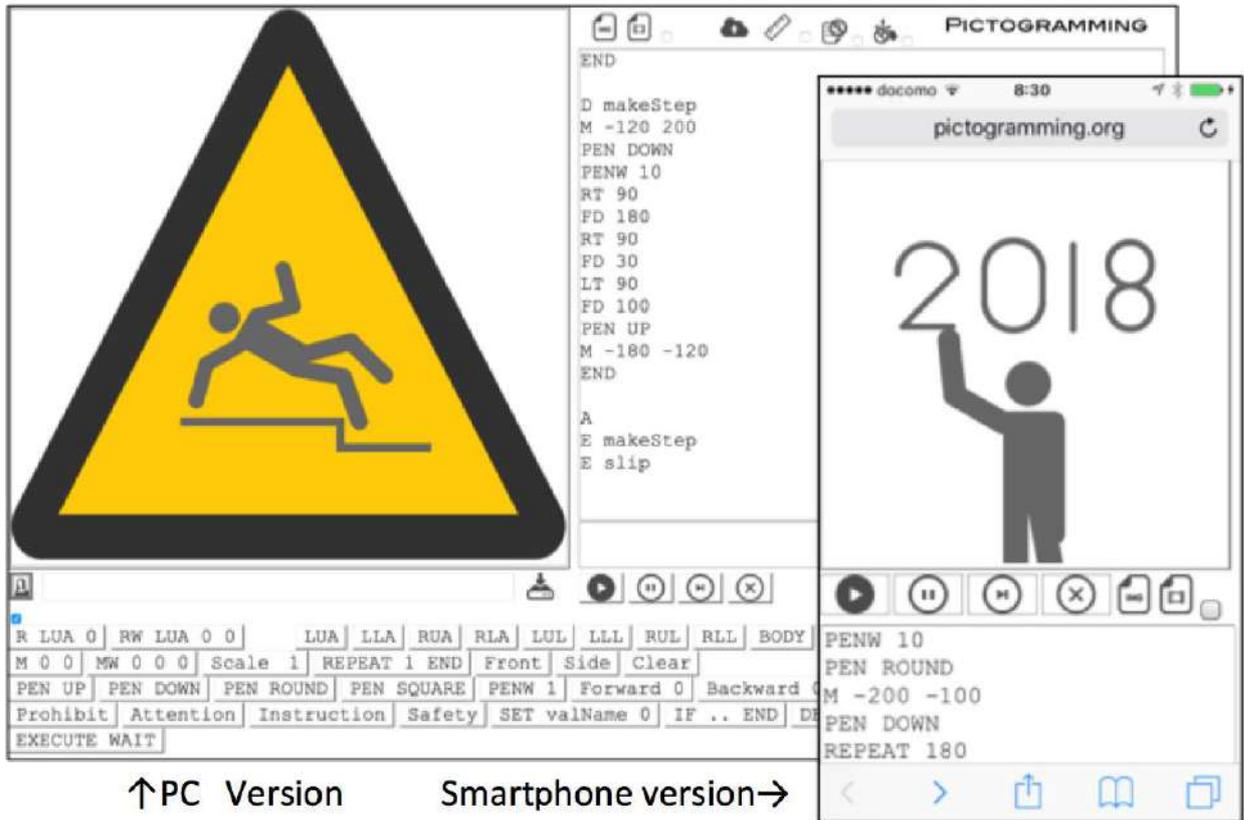


Figure 1. Screenshot of Pictogramming

A large human-shaped pictogram is displayed in the human pictogram display panel. The panel can display either the front or side views of the human pictogram, as defined by ISO 3864 Appendix, where both comprise nine parts: body and head (considered as a single part), two upper arms, two lower arms, two upper legs, and two lower legs. The size of each part conforms to ISO 3864 (see Figure 2). A programming panel indicating parts of the body in the human pictogram faces front, but it is assumed that the human pictogram is user yourself of reflected in a mirror. So, “left upper arm” and “left lower arm” are displayed on the left-hand side and users associate it with their left arm.

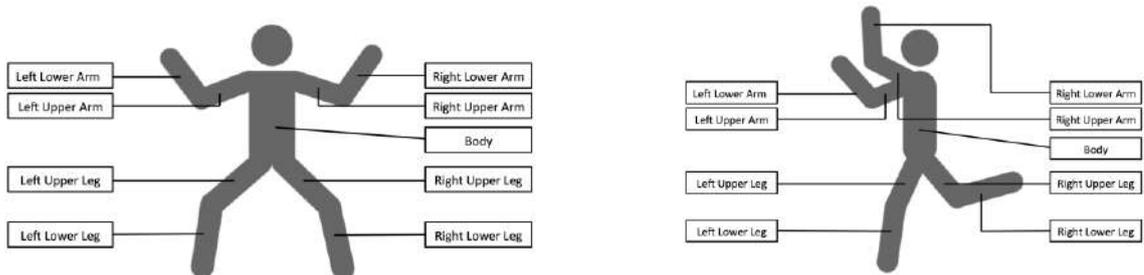


Figure 2. Front and side views of a human pictogram

Sample code

Operations on the human pictogram are input and defined in the program code description area. The input string to change states (positions) follows a format⁵⁹ that separates opcode and arguments with blanks, as follows.

opcode arg1 arg2 ...

Figure 3 shows a sample code. Line 1 changes the scale of the human pictogram. Lines 2 to 8 are in accordance with the LOGO programmer; this draws a rectangle represents the victor's platform. The "M(ove)" command shown in line 10 means move 50 px in the positive x-axis direction 200 px in the negative y-axis to get up on box. The "R(otate)W(ait)" command means rotate a part of the body over some seconds and the next command is not executed until the movement is complete. example, "RW LUA -120 1" shown in line means "rotate the Left Upper Arm (LUA) 120° clockwise for 1 s."

Lines 13 to 19 represent waving the left hand three times at a probability of 50%.

The commands are classified into two types. First, the command that changes the shape of the human pictogram is the "Pictogram Animation Command." The second type of command is almost equivalent to turtle graphics, i.e., the "Pictogram Graphics Command." Command names, arguments, and each process can be easily predicted from personal experiences or existing knowledge. Moreover, the commands and movements of the human pictogram are associated with fine granularity.

Thus, pictogram graphics focus on the movement of the human body, and pictogram animation focuses on the rotation of human body parts. The combination of these two types is a unique feature of Pictogramming.

Figure 4 shows an example of pictogram animation, pictogram graphics, and a combination of both types.

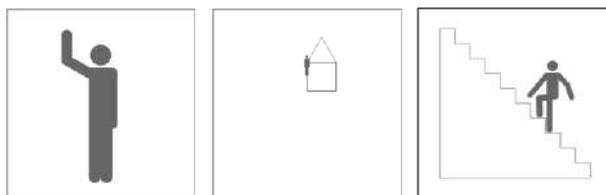


Figure 4. Examples of outputs from pictogram animation (left), pictogram graphics (center), and a combination of both types (right)

```

01: SC 0.3
02: PEN DOWN
03: // Draw victor's platform
04: REPEAT 4
05: FD 100
06: RT 90
07: END
08: PEN UP
09: // Get up on the victor's platform
10: M 50 -200
11: // Raise the left arm diagonally
12: RW LUA -120 1
13: // Wave hands to left and right 3 times
14: IF [rand(1,6) >= 4]
15: REPEAT 3
16: RW LLA -60 0.3
17: RW LLA 60 0.3
18: END
19: END
    
```

that
and
the
For
12

Figure 3. Sample source and output

⁵⁹ The command list can be seen at http://pictogramming.org/?page_id=475

Syntononic Learning

Papert developed Logo and noted that it is very important that children can execute Logo commands by pretending to be a turtle using their own bodies; this is called “syntononic learning” [Papert 1977][Papert 1980]. Papert also noted the following:

1. Body syntononic learning: Strongly associated with the senses of children and knowledge of their bodies.
2. Ego syntononic learning: Consistent with the self-consciousness of children as humans with intention, purpose, desires, likes, and dislikes.
3. Cultural syntononic learning: Linked to personal activities that are firmly and positively rooted in one’s culture.

The human pictogram resembles a body that represents the ego and the illustrated pictograms represent one’s culture. Thus, the human pictogram conforms to these three types of syntononic learning. Figure 5 shows works that were all created by the application. Figure 5(A)–(B) shows typical human movement, created mainly based on body syntononic learning. Figure 5(C)–(D) represents artistic, humor, some major person's famous action, and so on. These pictograms are difficult to categorize, but all are based on ego syntononic. Figure 5(E)–(F) shows the pictograms linked to personal activities based on rules, morals, and culture, and some of these are widely used in one's culture. Of course, these types of syntononic learning cannot be separated clearly, but these works are based on a combination of various kinds of syntononic learning.

					
(A) Running	(B) Waving hand	(C) Cupid	(D) Famous Japanese TV character's pose	(E) Do not use smartphone while walking	(F) Please knock when you enter a room

Figure 5. Example of works influenced by various type of syntononic learning

Pictogramming —Enhance constructionism using Human Pictogram (Broad meaning of Pictogramming)

Shift from instructionism to constructionism by reconstructing a human relationship

Now, we consider a mass classroom. Figure 6(A) represents a typical relationship between members of the classroom. The teacher instructs the students unilaterally. The human pictogram is introduced and the student starts to command or operate it (Figure 6(B)), and then begins to communicate with one's human pictogram (Figure 6(C)). Figure 6(D) is a scene in a classroom using the Pictogramming application. One student talks to another student about his/her pictograms and mimics the movement. As the pictogram is highly visible, the student happens to see the pictograms in the monitors naturally and acts with them. This means that one student communicates with other students’ human pictograms (Figure 6(E)), and communicates with other students on human pictograms (Figure 6(F)). The students program and create numerous works via this process using the philosophy of constructionism.

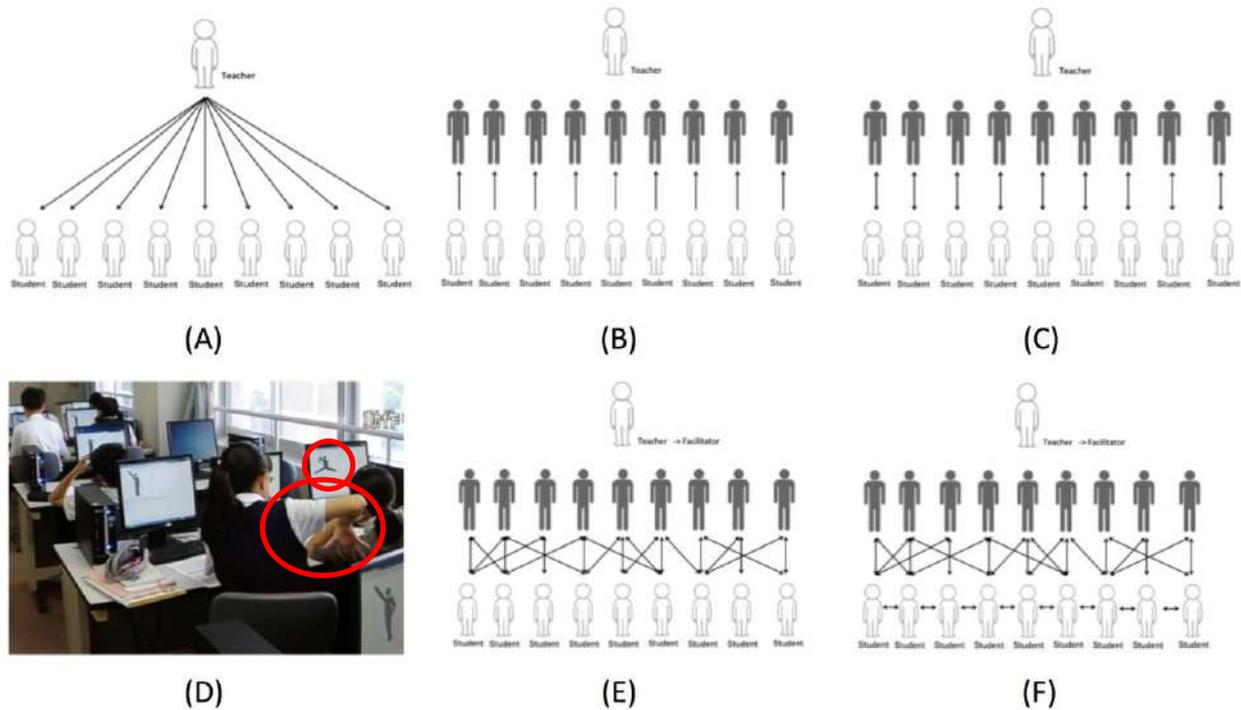


Figure 6. Shifting relationship between the student and the teacher by interference of human pictograms

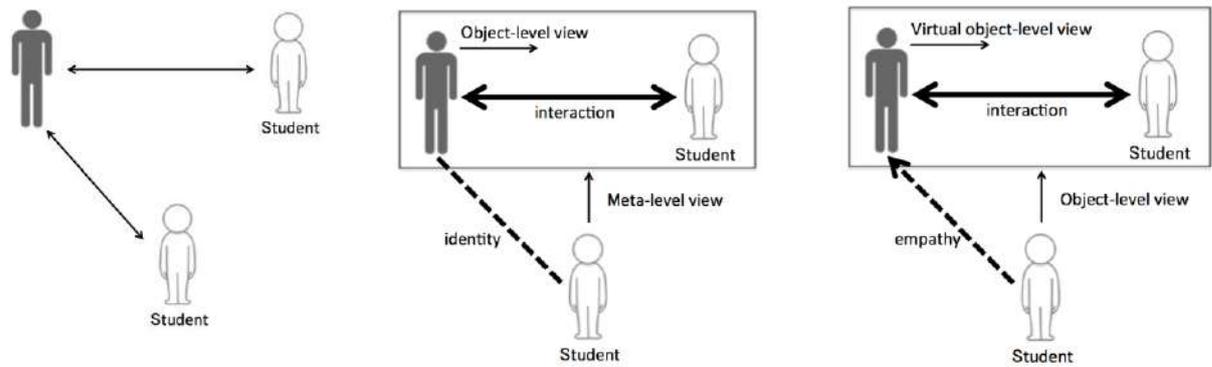
Experimental cognition and reflective cognition

A cognitive science researcher mentioned that the cognition could be classified into two types [Norman 1993]. One is experiential cognition, which is data-driven information processing with reactivation of information patterns in human memory. The other one is reflective cognition, which is concept-driven processing in deep reasoning such as decision-making and planning. Table 1 shows an example of activities of experimental cognition and reflective cognition.

Table 1. Example of activities of experimental cognition and reflective cognition

Experimental cognition	Reflective cognition
<ul style="list-style-type: none"> ● Participating in participatory activities. 	<ul style="list-style-type: none"> ● Learning how to play. ● Watching the state of activities and thinking instead of participating.

Okamoto associated a duplex viewpoint with these two types of cognitive processes [Okamoto 2005]. Figure 7 shows an image of a duplex viewpoint realized by a human pictogram. One is an “object-level view,” and the other is a “meta-level view.” These two viewpoints are tightly linked through the self-identification of “ego” (reflective subject) and “self” (experiential subject). In observing the interaction of others from a distance, such duality is generally not established. However, if the observer can emphasize his viewpoint with one participant in the interaction, he can also acquire the virtual object-level view so that one can experience the interaction as if it were one's own. This duplex viewpoint via empathy shows how first-person engagement can be achieved.



(A) Part of Figure 6(E)–(F) (B) Involvement as a participant (C) Involvement as an observer

Figure 7. Empathy channel: (B) involvement as a participant in an interaction using a human pictogram, (C) involvement as an observer of the interaction (panels (B) and (C) are based on [Okamoto 2005])

Conclusion

This paper presents an overview of a new programming learning application called Pictogramming and indicates the effectiveness of this application. Moreover, we proposed the broad meaning of Pictogramming, which is a learning environment based on the adoption of a human pictogram. We illustrated that this environment enhances the learning process with the philosophy of constructionism; then, we discussed the duplex viewpoint. In the future, we aim to demonstrate the effectiveness of using human pictograms by performing experiments and in practice.

References

- Cooper, S., Dann, W. and Pausch, R. (2003) Teaching objects-first in introductory computer science. In Proceedings: *34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. ACM, Reno, NV, USA. p. 191-195.
- Martínez-Santiago, F., García-Cumbreras, Miguel., Montejo-Ráez, A., Díaz-Galiano, Manuel. (2016) Pictogrammar: an AAC device based on a semantic grammar. Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications , p. 142-150.
- Halan, S., Rossen, B., Crary, M. and Lok, B. (2012) Constructionism of virtual humans to improve perceptions of conversational partners. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA. p. 2387-2392.
- Ito, K. (2018) Pictogramming - Programming Learning Environment using Human Pictogram. IEEE EDUCON 2018.
- Norman, D. A. (1993) Things That Make Us Smart: Defending Human Attributes in the Age of the Machine. Addison.
- Okamoto, M., Nakano, Y. I. and Nishida, T. (2005) Toward enhancing user involvement via empathy channel in human-computer interface design. Bold. L., et al. (Edo.), Lecture Notes in Computer Science Vol. 3490, Intelligent Media Technology for Communicative Intelligence, p. 111-121, Springer.
- Ota, Y. (1987) Pictogram Design. Kashiwashobo.
- Salanci, L. (2010) EasyLogo – discovering basic programming concepts in a constructive manner. In Proceedings: *Constructionism 2010*.
- Resnick, M. (1996) StarLogo: an environment for decentralized modeling and decentralized thinking. In *Conference Companion on Human Factors in Computing Systems (CHI '96)*, Michael J. Tauber (Ed.). ACM, New York, NY, USA, p. 11-12.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009) Scratch: programming for all. *Commun. ACM*, 52, 11, p. 60–67.

Seymour, S. (1977) A learning environment for children. *Computers and Communication: Implications for Education*, New York: Academic Press, p. 271-278.

Seymour, S. (1980) *Mindstorms, Children, computers, and powerful ideas*. Basic Books, Inc.

Human Pictogram Unplugged: Unified Learning Environment of Computer Science Unplugged Using Human Pictograms

Kazunari Ito, *kaz@si.aoyama.ac.jp*
Aoyama Gakuin University, Japan

Aoi Yoshida, *aoi@si.aoyama.ac.jp*
Aoyama Gakuin University, Japan

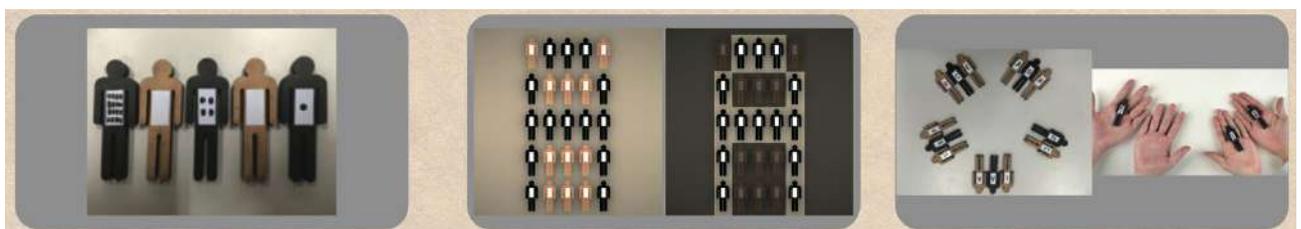
Takashi Yoneda, *yoneda@port.kobe-u.ac.jp*
Kobe University Secondary School, Japan

Yuichi Oie, *c8116002@aoyama.jp*
Aoyama Gakuin University, Japan

Abstract

We have been developing a unified learning environment for computer science unplugged (CSU) using human pictograms called “Human Pictogram Unplugged” (HPU). CSU is a methodology of teaching computer science without the use of personal computers. One of HPU’s characteristics allows students to conduct activities with a set of unified human-shaped pictograms (i.e., a “human pictogram”). Educators usually must prepare and maintain teaching materials for each CSU activity: a balance for “sorting” activity, worksheets for “image representation,” and special cards for “binary numbers,” for example. To decrease this economic and temporal burden is an important and notable issue in education. Human pictograms are abstract representations of a people. Activity themes using human pictograms would at once be more strongly connected to one’s experience and knowledge and also firmly and positively based on one’s culture. Therefore, HPU will promote syntonetic learning.

Activity examples of Human Pictogram Unplugged



Binary Number

Image Representation

Routing and Deadlock

In this

paper, we explain the concept of HPU, show example activities of HPU, and discuss HPU’s effects and characteristics from a constructionist viewpoint.

Keywords

human pictogram; computer science unplugged; syntonetic learning; duplication of viewpoint; educational environment

Introduction

Computer Science Unplugged (CSU) is a methodology for teaching the principles of computer science without the use of personal computers. CSU was developed by Dr. Tim Bell and Mike Fellows[2009][Bell 2012] and has since been adopted by many countries as a method of teaching computer science. The main approach of CSU is learning through indoor or outdoor physical activities. This is to say, the subject

is person. So, we implemented an idea that engages physical activities to unify all teaching materials using human pictograms.

A pictogram is a graphical symbol used to understand semantic concepts based on the meaning of its shape [Ota 1987]. Pictograms have been studied and used in fields such as counseling, safety, facility management, cross-cultural communication, and semiotics [Mori 2009, Hassan 2015]. Pictograms are also designed to provide information regarding a human's action or status. So, if human pictograms are used as main characters in computer science education, we may expect very high levels of effectiveness since virtual humans are often used to enhance learning environments [Halan 2012].

The next Japanese national curriculum will begin in 2020, and it aims to expand informatics in K-12. In such situations, Japan increasingly emphasizes computational thinking [Wing 2008].

Further, unplugged is the main activity used by teachers for ordinary school subjects, so teachers may come to think of unplugged as a more attractive method to teach computers. However, most elementary and secondary school teachers are not familiar with computer science. Indeed, the phrase "computer science" is often met with fear and disgust in the minds of teachers. Thus, using term "human pictogram" is one strategy to ease teacher anxiety about computer science activities.

Educators usually must prepare and maintain teaching materials for each CSU activity. For example, balances are used for "sorting" activities, worksheets are used for "image representation," and cards are used for "binary numbers." To lessen these economic and temporal burdens, we can adopt unified learning materials that can be used for multiple activities. This is a very important, notable issue for education.

For this study we developed "Human Pictogram Unplugged" (HPU), a unified computer science unplugged learning environment that uses human pictograms. We then discuss the effects and characteristics of HPU from a constructionist perspective.

Learning Material

We designed two types of pictograms: one made from wood, and one made from paper. Of course, human pictogram construction is not limited to these materials.

Wooden Based Human Pictogram

One type of pictogram is wooden and upright (Figure 1). The material is a 5.5mm-thick MDF (Medium Density Fiberboard) cut by a laser. One side is painted black to distinguish it from the opposite side. A seal-type, 1cm width by 2cm height white sheet is affixed to the chest areas of both sides so that one can write or erase symbols such as numbers and characters with a marker or an eraser.

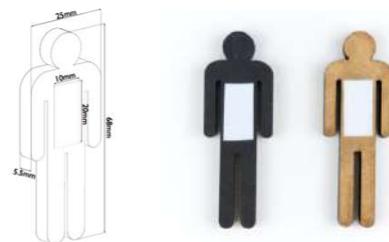


Figure 1. Wooden based pictogram

Paper Based Human Pictogram

The other pictogram is made from 55mm x 91mm name card paper (Figure 2). This paper is made of special material, so one can write and erase symbols such as numbers and characters using a marker or an eraser.

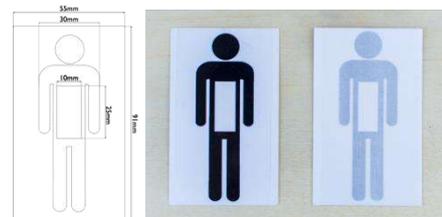


Figure 2. Paper based pictogram

A Set of Learning Material

The learner then uses the human pictograms, a white board marker, and an eraser (Figure 3).

We introduce color information (black and white) to human pictogram, and permit to write some characters on one's chest. The pictogram has been used colors to inform specific information, and normally used for visually or cognitive impaired person. And the representation style that some characters are attached to pictogram is also widely adopted in the public used pictograms. That's key

point that these human pictogram materials can be adopted to lots of unplugged activities. We also aim to help users imagine real humans who illustrate something to audience by paper or who play in some events with attaching bibs by drawing a rectangle shape on the chest area for writing some characters. He/she also uses restriction sheets to limit the human pictogram's movement for some activities.



Figure 3. A set of learning material

Activity List of Human Pictogram Unplugged

Overview

This section shows how we have taken Bell's activities [Bell 2015] and redesigned them to incorporate human pictograms. These activities are listed in Table 1.

Table 1. Activity list of HPU

Binary numbers	Data representation	Image representation	Information theory	Error detection and error correction
Search algorithms	Sorting algorithms	Sorting networks (Parallel processing)	Finite state automata	Routing and deadlock

Binary numbers

The original activity is called "Count the Dots" and uses binary or other notations to represent numbers. Five kinds of cards (labelled 16, 8, 4, 2, and 1 dots respectively on one side) are displayed by five children. Here, Figure 4 on the left is real activity, and Figure 4 on the right is simulated version using human pictograms.



Figure 4. Real and simulation

Data representation

This is one of additional activities for binary numbers. The original activity's name is "Sending Secret Messages." Data Representation here uses binary number representations of characters. For example, assign 0-25 to alphabet A-Z, and represent one character with five human pictograms. Figure 5 shows, as an example: "L (11: first row) O (14: second row) V (21: third row) E (4: fourth row)."



Figure 5.

Image representation

The original activity's name is "Colour by Numbers." Students learn image representation of pixel images and data compaction by expressing an array of numbers. The activity remixes "Forming human letter" by corresponding one pictogram to each pixel. Figure 6 shows arrangements, which can represent continuous number of same color's human pictograms "0131, 131, 5, 131, and 131."



Figure 6. Example of character A

Information theory

The original activity's name is "Twenty Guesses." The "answerer" guesses correct answers by asking a maximum of 20 questions, to which the "question master" can reply "Yes" or "No" clearly. The question master sets a correct answer and writes it on the back side in advance. Figure 7 shows that question master sets a number "42."



Figure 7.

Error detection and error correction

The original activity's name is "Card Flip Magic." Square magnet sheets are used, with different colors on each side arranged in a grid structure. First, a magician arranges an even number of same colored sheets in rows and columns. An audience member turns over one of sheets while the magician looks away. The magician later guesses the sheet. The HPU version of this activity uses human pictograms instead of magnet sheets. Figure 8 is an example.

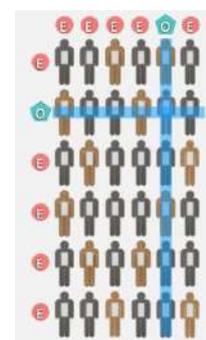


Figure 8.

Search algorithms

The original activity's name is "Battleships." An illustrated sheet with numbered warships is provided, and two players each guess the location of the other's target warship. In the HPU version, one human pictogram is the equivalent of one warship (Figure 9). Players write a comparative sign (like a number or a friend's family name) on a human pictogram's chest area.



Figure 10. Activity example of sorting algorithm



Figure 9. Activity example

Sorting algorithms

The original activity's name is "Lightest and Heaviest." A balance and some weights are provided, and player arranges the weights in order of weight. In the HPU version, each human pictogram's weight is same. Player writes a comparative sign (like a number or a friend's family name) on the chest area of a human pictogram, and sets each human pictogram's value. Figure 10 on the left is the initial state (Japanese family name is written on one's chest). Figure 10 on the right is the final state, where human pictograms are arranged in alphabetical order.

Sorting networks (Parallel processing)

The original activity's name is "Beat the Clock" and the goal of this activity is to teach parallel processing. Each pair of human pictograms is stored in one of the decks of first row (Figure 11). Each human pictogram is then compared to another pictogram in the same deck, and it moves to the next row deck as a result of comparison. The comparison and moving to next row level are conducted simultaneously. All human pictograms are sorted upon arriving at the final row level.

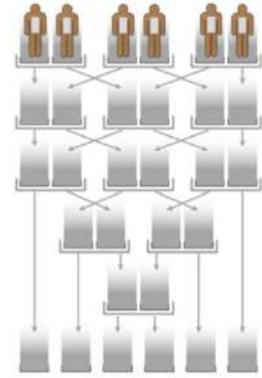


Figure 11. Example

Finite state automata

The original activity's name is "Treasure Hunt." The transition source, input, and indications of whether it is an initial status are described on the front of a pictogram, and the transition target is described on the back. Figure 12 on the left shows an example of finite state automaton, and Figure 12 on the right shows the same structure as the state is constructed by several human pictograms. The player turns over a pictogram with a status number written on the chest. The next status is then illustrated on the back of the selected pictogram. The player attempts to catch the target pictogram (Number 5 circled in the Figure 12(right)).

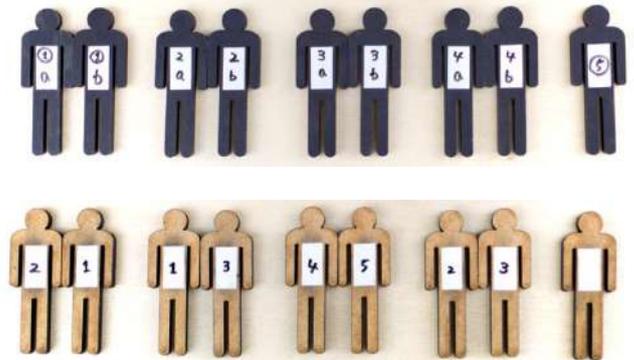
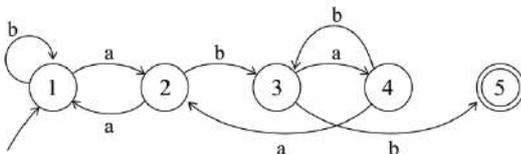


Figure 12. Example of finite state automaton (left) and representation by human pictograms (right)

Routing and deadlock

The original activity's name is "The Orange Game." In this activity, the learner can easily experience first and third person viewpoints. Figure 13 on the left is an activity with first-person point of you. Each player holds a pictogram in their hand, and passes pictograms from hand to hand to satisfy the arrangement conditions. Figure 13 on the right demonstrates a third-person viewpoint. Flesh-colored human pictograms, equivalent to real people, are arranged in a circle. Black pictograms located on the both sides of the flesh-colored pictograms are equal to pictograms in a player's hand in first-person point of view.

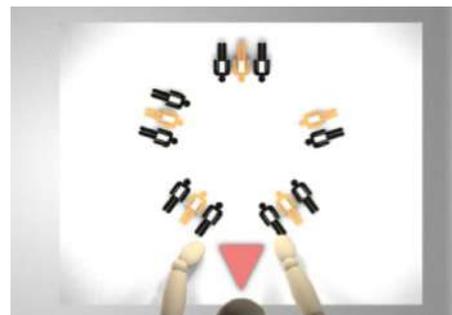


Figure 13. Activity with first-person point of you(left) and third-person viewpoint (right)

Enhance constructionism using Human Pictogram Unplugged

Shift from instructionism to constructionism by reconstructing human relationships

We now consider a large classroom. Figure 14(A) represents typical relationships between class members. The teacher instructs students unilaterally. Human pictogram unplugged is introduced, and the students handle (Figure 14(B)) and begin to communicate and reflect using their human pictograms (Figure 14(C)). Figure 14(D) portrays a multiplayer activity classroom scene. One student talks to the others about the human pictograms on the table. This means that one student communicates with other students using human pictograms (Figure 14(E)), and communicates with other students about human pictogram unplugged activities (Figure 14(F)). The students use the philosophy of constructionism in this process.

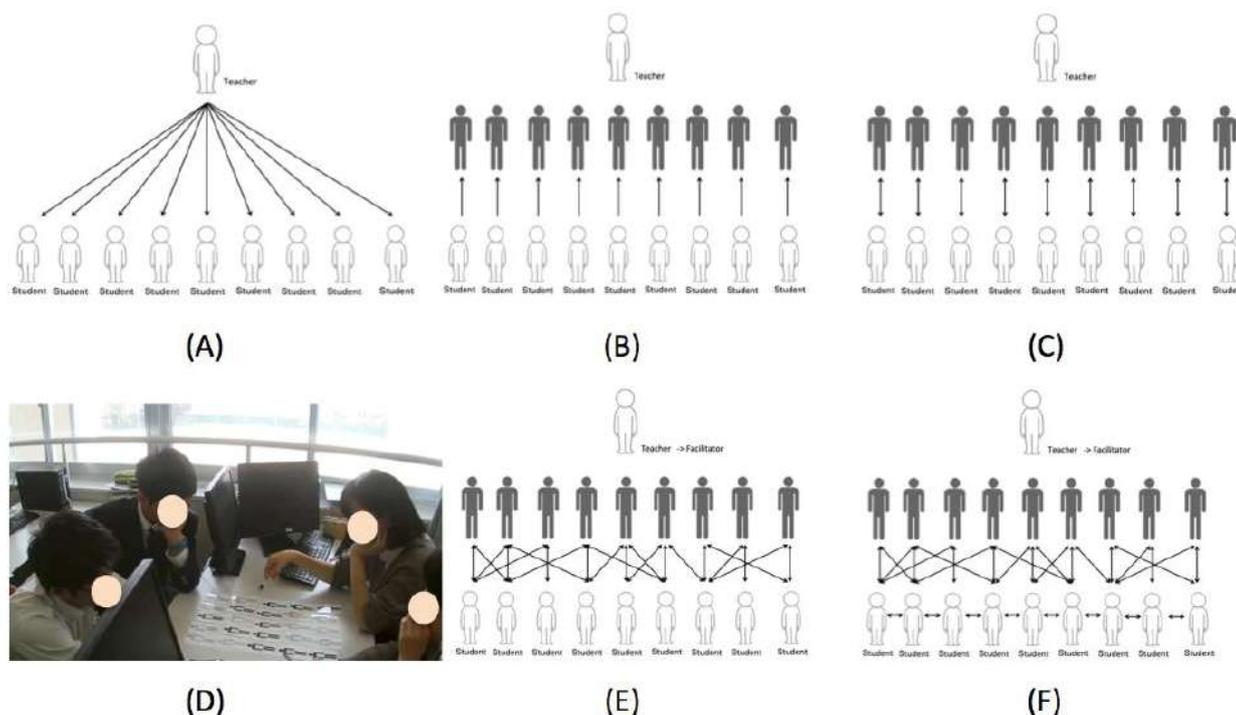


Figure 14. Shifting relationships between students and teachers by human pictogram interference

Experimental cognition and reflective cognition

A cognitive science researcher mentioned that cognition can be classified into two types [Norman 1993]. One is experiential cognition, which is data-driven information processing with reactivation of information patterns in human memory. The other is reflective cognition, which is concept-driven processing in the context of deep reasoning such as decision-making and planning. Table 2 shows examples of experimental cognition and reflective cognition activities.

Table 2. Example of activities of experimental cognition and reflective cognition

Experimental cognition	Reflective cognition
Participating in activities.	<ul style="list-style-type: none"> • Learning how to perform the activities. • Watching the activities and thinking of participating.

Okamoto associates a duplex viewpoint with these two types of cognitive processes [Okamoto 2005]. Figure 15 shows an image of a duplex viewpoint as realized by a human pictogram.

One is an “object-level view,” and the other is a “meta-level view.” These two viewpoints are tightly linked through the self-identification of “ego” (reflective subject) and “self” (experiential subject). In observing the interaction of others from a distance, such duality is not generally established. However, if the observer can emphasize his viewpoint with another participant in the interaction, he can also acquire the virtual object-level view of another participant so that one can experience as his own perspective. This duplex viewpoint via empathy shows how first-person engagement can be achieved.

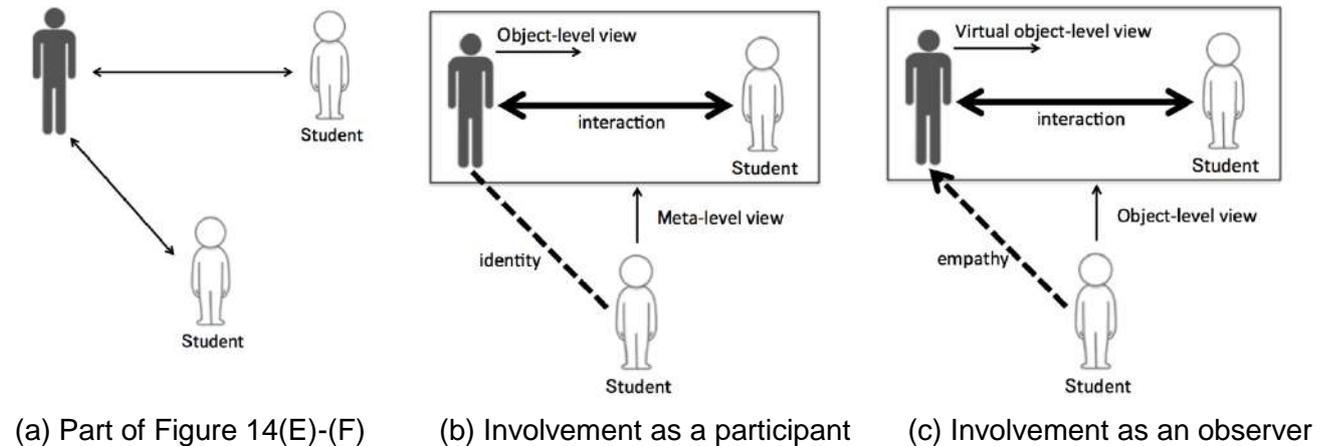


Figure 15. Empathy channel: (b) involvement as a participant in an interaction using a human pictogram, (c) involvement as an observer of the interaction (panels (b) and (c) are based on [Okamoto 2005])

Activities design conscious of syntonic learning, and scaffold for or simulation of real daily life

Papert, who developed Logo, insists that children can execute Logo commands by using their bodies to pretend to be turtles; this is called “syntonic learning” [Papert 1977, 1980]. Papert also notes the following:

1. Body syntonic learning: Strongly associated with children’s senses and knowledge of their bodies.
2. Ego syntonic learning: Consistent with children’s self-consciousness as humans with intention, purpose, desires, likes, and dislikes.
3. Cultural syntonic learning: Linked to personal activities that are firmly and positively rooted in one’s culture.

Unplugged encourages learners to design and create new activities. When designing and performing unplugged activities, it is very important to be aware of syntonic learning. The human pictogram is a kind of human body projection that represents ego, and it becomes increasingly easy to create new, culturally friendly activities since the human pictogram conforms to these three types of syntonic learning. The final goal of unplugged activities is to identify with real daily life. Human pictogram unplugged activities are human-centric, so they scaffold real daily life or a simulation of it.

Conclusion

This paper proposes and presents an outline of a computer science unplugged unified learning environment using a human pictogram called "Human Pictogram Unplugged." We illustrated that this environment enhances learning processes using constructionist philosophy. We have already used this method in several schools, and we shall soon analyze the effectiveness of using human pictograms by experiments.

References

- Bell, T. et al. (2009) Computer Science Unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, Vol. 13, No 1, pp. 20-29.
- Bell, T., Rosamond, F., & Casey, N. (2012). Computer Science Unplugged and Related Projects in Math and Computer Science Popularization. In H. L. Bodlaender, R. Downey, F. V. Fomin, & D. Marx (Eds.), *The Multivariate Algorithmic Revolution and Beyond* (pp. 398-456). (Lecture Notes in Computer Science; Vol. 7370). Online: Springer. DOI: 10.1007/978-3-642-30891-8_18
- Bell, T. et al. (2015) The English-language CS Unplugged activities in book form as a free download. https://classic.csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf (Last visited: March, 30th 2018)
- Hassan, M. M. E. (2015) The semiotics of pictogram in the Signage Systems. *International Design Journal*, Vol. 5, No 2, p. 301-315.
- Halan, S., Rossen, B., Crary, M. and Lok, B. (2012) Constructionism of virtual humans to improve perceptions of conversational partners. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*, ACM, New York, NY. p. 2387-2392.
- Mori, Y. Takasaki, T. and Ishida, T. (2009) Patterns in pictogram communication. In *Proceedings of the 2009 international workshop on Intercultural collaboration (IWIC '09)*, ACM, p. 277-280.
- Norman, D. A. (1993). *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Addison, Inc.
- Okamoto, M., Nakano, Y. I. and Nishida, T. (2005) Toward enhancing user involvement via empathy channel in human-computer interface design. Bold. L., et al. (Ed.), *Lecture Notes in Computer Science* Vol. 3490, *Intelligent Media Technology for Communicative Intelligence*, p. 111-121, Springer.
- Ota, Y. (1987) *Pictogram Design*. Kashiwashobo, Inc.
- Papert, S. (1977) *A Learning Environment for Children*. Computers and Communication: Implications for Education. New York, Academic Press, p. 271-278.
- Papert, S. (1980) *Mindstorms, Children, Computers, and Powerful Ideas*. Basic Books, Inc.
- Wing, J. M. (2008) Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 366(1881), pp. 3717–3725.

Some Reflections on Designing Constructionist Activities for Classrooms

Yasmin B. Kafai, *kafai@upenn.edu*
University of Pennsylvania, Philadelphia, PA, USA

Deborah A. Fields, *deborah.fields@usu.edu*
Utah State University, Logan, UT, USA

Abstract

In this paper, we present our guiding principles for designing a constructionist curricular unit with electronic textiles called “Stitching the Loop,” which introduces high school students to key concepts in in crafting, circuit design and computing. Our principles were to design for (1) engagement, (2) expression, (3) depth, (4) experiences, (5) audience, (6) collaboration, (7) reflection, (8) failure, (9) practicalities, and (10) iteration. Over three years, we worked together with dozens of high school teachers and hundreds of students in designing, implementing and revising classroom activities where students create and craft a series of individual and collaborative electronic textiles. Situated within the larger framework of the *Exploring Computer Science* curriculum, we illustrate how these guiding principles fostered an equity- and inquiry-oriented pedagogy through which teachers can support students’ learning.

Keywords

curriculum; coding; computer science education; electronic textiles, constructionism, maker movement

Introduction

In constructionist approaches to learning and teaching, much emphasis has been placed on the design and development of construction tool kits (Resnick & Silverman, 2005). Hundreds, if not thousands, of digital, physical and hybrid construction kits have been developed to help students engage with STEM topics and express their ideas and personal interests (e.g., Blikstein, 2012). Far less attention has been given to the design and development of curricula where constructionism connects to academic content within a classroom. While curriculum design is often seen as contradictory to constructionist pedagogy because it constrains and directs student activities, we argue that it also can broaden access to both making and coding, deepen learning in those fields, and promote better diversity in what is being made. This is particularly important in coding and maker activities which have a longstanding history of inaccessibility to non-White, non-male students from working-class communities (Margolis, Estrella, Goode, Holme, & Nao, 2017).

In this paper, we report on the design of an eight-week long formal curricular unit, called “Stitching the Loop,” which facilitated students’ interest-driven projects, supported peer collaboration, and applied equity-minded teaching. Our electronic textiles (e-textiles) unit was situated within *Exploring Computer Science* (ECS), an equity-focused and inquiry-based year-long introductory computer science course taught in public high school classrooms all over the country (Goode, Chapman & Margolis, 2012). We concentrated on bringing creative making in the form of e-textiles into computer science classrooms. E-textiles are hybrid designs, using conductive thread to sew LEDs, sewable microcontrollers (e.g., LilyPad Arduino, Adafruit Circuit Playground), sensors and other actuators into fabric or similarly soft media (Buechley & Eisenberg, 2008). The unit consists of a series of four open-ended projects (see Figure 1) with creative constraints that help students learn challenging concepts in computing, electronics, and crafting three-dimensional designs while also supporting personal expression and creativity. In the following sections we review different approaches to constructionist curriculum design and articulate the guiding principles we developed in designing the e-textile activities.

Background

Arguably the first constructionist curriculum was published in 1971 as a memo titled “Twenty Things To Do With A Computer” in which Papert and Solomon suggested a variety of activities that could engage children in programming, among them: making a turtle draw images on paper by programming a pen to lift up and down; programming behaviors such as the turtle following along walls in a room; writing programs to draw geometric spirals, making an online movie by programming a change in petals on a flower, programming sounds to play a song, playing spacewar games, and many more. The memo concludes its list with the last recommendation asking the reader to come up with twenty more things to do with a computer.

This curriculum consisted of a *collection* of different projects, not necessarily organized in a sequence. These projects promoted what Papert (1980) called “powerful ideas” about computing such as recursion or repetition by situating them in visible realizations such as drawing flowers or circles. They also connected to other academic subjects providing an alternative way for children to experience geometry or mathematics. Furthermore, they sought out compelling applications such as making music or movies or playing and designing games that would resonate with students’ interest. Many of these features have become guiding principles for other constructionist efforts, for instance, the development of the recent guide to *Creative Computing* (Brennan, Balch, & Chung, 2014) for Scratch activities.

A different approach to constructionist curriculum has been software design for learning, where *one project* becomes the central focus, such as designing instructional software (Harel & Papert, 1990). Here students work on designing software such as learning tools or educational games that teach academic content like mathematics or science. In this context, the learning of programming is connected with the learning of academic content by teaching subject areas to others through designed software. For instance, software games designed by students (Kafai, 1995) included multiple problems in words, graphics, and often provide stories and animations. Furthermore, students created a whole product by including package design and advertisements for their games. A key distinction to the previous efforts is that in designing software applications, students work on one complex, long term project rather than a collection of several smaller projects. Another dimension is that students journal about their ongoing design process in notebooks outlining project ideas and reflecting on their challenges.

More recent efforts have combined elements from these two approaches by providing an explicit sequence of projects, such as game design as in Repenning and colleagues’ (2015) “Scalable Game Design,” which engaged students in a series of game design projects with Agentsheets. Likewise, the “Globaloria” platform (Caperton & Reynolds, 2011) included a year-long game design curriculum with a supporting social network where students can post and comment on games. Building on studio design pedagogy (Hetland, Winner, Veenema, & Sheridan, 2013), Fields and colleagues (2016) also created a weeklong “Scratch Camp” where students engaged in an intentional series of projects with creative constraints interjected with mini-lessons, gallery walks, and a final interactive event sharing with families and friends. Our curricular e-textile unit combines the collection of projects and the one-project approaches by creating an intentional series of projects with creative constraints, culminating in a final project with a reflective portfolio. Below we share the guiding principles for designing the e-textile curricular unit.

Guiding Principles for *Stitching the Loop*

The *Stitching the Loop* curriculum contains big ideas, recommended lesson plans, and sample rubrics, with much room for students and teachers to interpret and bring in their own style, evidenced in the successful and nuanced ways it has been implemented by teachers over the past three years. It is accompanied by a 60+ page technical guide with fine-grained tutorials about crafting, circuit design, coding, and troubleshooting. By consciously combining traditionally masculine activities such as engineering and computing with traditionally feminine activities such as crafting and sewing, e-textiles can disrupt preconceptions about who can do computing, engineering, and crafting (Kafai, Fields & Searle, 2014). With this background, we brought together experts in e-textiles educational activities and the ECS development and implementation team. The curriculum was co-developed to combine

best practices of teaching and creating e-textiles based on a constructionist philosophy alongside ECS principles (inquiry, equity, and computing) and style.

Over the past three years, we have written and piloted the curriculum with two teachers (Year 1—Spring 2016), four teachers (Year 2—Spring 2017), and now 15 teachers (Year 3—Spring 2018) in one of the largest and most diverse school districts in the United States. All teachers participated in 3-4 days of professional development, focusing on creating the core projects (see Table 1 and also Figure 1) and reflecting on principles of the unit like valuing aesthetics, personalization, mistakes, and audience. Between implementations we used analysis of observations and interviews with teachers and students to revise the unit. Our experiences in creating, revising, and implementing this unit highlight the possibilities in introducing making to computer science in ways that promote equity, imagination, and personalization in classrooms.

Table 1. Overview of projects in *Stitching the Loop* curricular unit

Project	Content	Description
#1 Paper Circuit (~1-2 hrs)	Single circuit project design: Create a simple paper circuit greeting card that includes one LED. Introduce the concept of aesthetic design and personalization.	<ul style="list-style-type: none"> • Simple circuit • Polarity • Materials: LEDs, copper tape (wire), paper
#2 Wristband (~5-6 hrs)	Simple wearable project: Create a wristband with three LEDs in parallel and a switch that turns on the project when the ends of the wristband are snapped together.	<ul style="list-style-type: none"> • Parallel circuit, switch • Reading circuit diagrams • Three-dimensional project • Deconstruction • Materials: Conductive thread, LEDs, fabric
#3 Collaborative Mural Project (~10 hrs)	Collaborative project: As a class create a mural, with each panel made by two students. Each panel must have five independently programmable LEDs and two switches, allowing for four blinking light patterns.	<ul style="list-style-type: none"> • Programming: Sequences, conditionals, embedded conditionals or Boolean statements • Collaborative work & division of labor • Materials: Conductive thread, LEDs, fabric
#4 Human Sensor Project (~10-14 hrs)	Capstone project: Create a project with two aluminium foil patches that act as a sensor when both are touched by a person. Program four+ lighting patterns based on different sensor readings.	<ul style="list-style-type: none"> • Sensor design (handcrafted) • Programming: operators, sensor range, Boolean statements • Materials: Conductive aluminium foil, human body, LEDs, fabric

We made several changes as we developed the unit. One of the most important ones involved a change in assessment as we shifted from pre/post-tests to reflective portfolios where students summarized their final projects, shared challenges that came up, and wrote about their progression during the e-textiles unit (Lui et al, 2018). The portfolio served to support student meta-reflection on their learning and to emphasize the *process* of making as much as the final *product* of making. The leadership model of the PD also changed as the two teachers from the first year of implementation took over nearly all training activities in the third year, bringing their hands-on expertise about managing students' creative making, teaching students how to sew, organizing materials, and handling classroom management in the e-textiles unit.

Designing for Engagement

We designed the e-textiles unit for engagement by keeping all projects open-ended, allowing students to express their interests, hobbies, and personal relationships in their artifacts. This was demonstrated by the vast diversity of students' projects and in students' own consistent expressions of creative freedom in the projects. As one student related, "I was able to make something that I wanted, anything, and I just created that and I liked it. It was fun." In some instances, students also displayed their engagement and relationships by designing their projects for others such as making a touch-sensitive soft toy for a little sister or making a blanket throw for a brother's birthday.

Designing for Expression

The unit emphasized “aesthetics first” through a personal design or sketch at the beginning of every project. In prior studies we found that starting with instruction (i.e., ways to design circuits) instead of with design resulted in poor engagement by students (Kafai et al., 2014). In contrast, by having students sketch out what they wanted their projects to look like, even if ideas were technically or practically infeasible, encouraged personal ownership from the beginning and set up students up to persevere through challenges (Kafai, Lee, et al., 2014).

Designing for Depth

We also wanted students to learn deeply through providing increasingly complex projects. Too often constructionist projects stay at what Blikstein and Worsley (2016) call the “keychain phenomenon” where students enjoy “low floors” to design but do not continue onward to the “high ceilings” possible with more advanced ideas and skills (Resnick & Silverman, 2005). To this end, the unit had introductory and complex e-textiles projects (Figure 1 and Table 1) that built on design, crafting, circuitry, and coding skills, each increasing in both difficulty and open-endedness.

Designing for Multiple Experiences

The design of the e-textile unit also provided multiple experiences in learning about design, crafting, circuitry, and coding skills through having students conceptualize and then implement more than just one complex project. Initially we had six projects but worked with teachers to provide repeated opportunities for students to engage in practices such as debugging, revising, testing, collaborating, and designing for other users in just four projects.

Designing for Audience

We further supported project designs with authentic audiences by having teachers display their own and students’ projects at the beginning of the unit, during the unit as a way to share peer knowledge (see next section), and at the end of the unit as a form of collaborative show and tell. Some teachers went further to encourage students to show their projects to other teachers at the school or put projects on display in school hallways and display areas. This made students’ projects transparent to each other for idea generation and also provided authentic audiences for students, a common principle in studio design education (Hetland et al., 2013).

Designing for Collaboration

Peer pedagogy, or students teaching each other, was another design principle of the unit in both intentional and emergent ways. We deliberately used pair programming during coding instruction moments and chose to make one project (the mural) collaborative at both a classroom and partner level. But we were also surprised to find from our research how often students’ helped each other in unstructured ways. When reporting on challenges on their individual final projects in their Year 2 portfolios, nearly one-fourth of students explicitly mentioned peer help as key to resolving bugs (see Jayathirtha, Fields & Kafai, 2018). Observations show even more frequent peer-to-peer help, encouragement, and support. Two things support this unstructured peer pedagogy in the unit. First, the physical structure of the classroom with students at small tables with shared supplies (scissors, thread, alligator clips, etc.) encouraged unstructured student collaboration. Student work (including errors) is visible and frustration is audible by sheer proximity. Second, teachers developed practices that support peer pedagogy, such as providing help to one student so that student could help others, connecting students to others who have expertise, and allowing student mobility in the classrooms, permitting them to get up and down and move around the room.

Designing for Productive Failure

Another goal of the unit was to support students in valuing the *process* of making projects, not just the final product. This meant finding ways to highlight mistakes and make them into learning opportunities rather than learning barriers. The teachers themselves developed several practices in this regard that we now explicitly model and name in professional development workshops (Fields, Kafai, Nakajima, Goode, & Margolis, in press). For instance, the teachers highlighted their own iterative practices of

creation, including their own mistakes, errors, and less-than-perfect projects in front of the classroom. This allowed the teachers to self-deprecatingly model practices of revision and iteration and coach students on tips for dealing with this process. The teachers encouraged students to think that it was okay not to be perfect the first (or the second, third, fourth) time.

The teachers also showcased students' challenges, mistakes, and in-process projects. They did this in multiple ways. First, teachers would highlight mistakes for the entire class during project time. One teacher created a tradition of saying, "This is my favorite mistake of the day!" and then would show the mistake and ask the rest of the class for help in identifying what was wrong and why. Second, teachers had students highlight mistakes through personal journal entries, some of which we adopted formally in later versions of the curriculum. For instance, one teacher added a journal question after the completion of the wristband project that solicited challenges that students had faced: "Think about this week's project, what was the biggest challenge?" Students wrote their own reflections before sharing out ideas. These methods made students' mistakes into a form of shared classroom knowledge, foregrounding students as experts in the classroom, a key practice of equity-based and constructionist teaching principles that situates knowledge in the hands of learners and not just teachers.

Designing for Reflection

Honouring the role of reflection in constructionist learning settings, another key element of the curriculum involves supporting students in consciously thinking about their processes of learning and honouring mistakes and challenges that occur during that process. Drawing on practices of reflection already present in the larger ECS curriculum (namely short journal entries and class discussion), we intentionally expanded these by including *design notebooks* and *portfolios*. Students responded very positively to the portfolio, saying that it helped them to see how much they had learned and appreciating that they were graded not just on the final product but also on their process of learning (Lui et al., 2018). In Year 3 we increased the supports for the portfolio by encouraging more frequent practices of documenting and reflecting on mistakes throughout the entire unit and not just on the final project. We included taking regular photographs of unfinished projects as "exit tickets" on crafting days, and made more "design notebook" entries where students suggested tips for others or noted changes that they made.

Designing for Practicalities

We also designed for the practicalities of managing materials and students, improving this each year. Teachers of *Stitching the Loop* taught in regular computer science classrooms not set up for the messiness of crafting. Early on the teachers developed practices to *manage materials and set-up* including 1) Using lidded boxes to contain table supplies; 2) Initiating set-up and clean-up practices where one student per table would pick up a craft box and later take it back; and 3) Organizing student work in individual Ziploc bags and storing these within the craft boxes. We also worked with *manufacturers* to make it easier to purchase materials for the e-textiles unit. This involved simplifying the number of merchants to order from (to satisfy school administrations) and lowering costs. The development of new microcontrollers like the Adafruit Circuit Playground that already had multiple switches and sensors onboard further allowed us to steeply lower the per-student cost of the unit to about \$40/student (instead of \$60+/student). We also negotiated with the manufacturer to create student and classroom kits that were easy to order and organize, with the added benefit of offering teachers of the unit bulk pricing for additional classes they needed supplies for.

Design for Iterations

Finally, as with any constructionist venture be it tools or activities, we iterated through the various aspects listed above. For instance, in the first year of the unit we had six projects, but this became overwhelming to fit into the limited 8-week window available as an ECS unit. We identified overlapping skills between projects and cut two projects that did not significantly add to students' skills. In Year 2 implementation we found that the four projects were sufficient for students developing the knowledge needed to carry out the final project and the time required was much more manageable. Furthermore, we shifted from a test-based assessment to a portfolio-based assessment where students shared

summaries of their projects, challenges and revisions that happened as they made them, and reflections on their learning overall.

Conclusion and Discussion

In *Mindstorms* Papert (1980) outlined a bold vision of how computers could help children learn, launching the development of numerous programming languages for learners, the design of various computational construction kits, and the creation of learning communities. Nearly forty years later this vision is making a comeback around the globe, promoting coding and making inside and outside of schools. Success stories of the Scratch platform and community and the Maker Movement have demonstrated that millions of kids can be interested in programming and in making electronics together in afterschool and online spaces. What does it mean for these activities to move back into the classroom with its focus on standards, curricula, and assessments within limited time periods and limited staffing?

Our guiding principles for designing the e-textile curriculum unit embrace constructionist ideas and approaches in creating anew the conditions where personal projects can flourish, students can support each other, teachers can become members of the learning community, and failure is seen as part of the process. In many ways, constructionist-oriented teachers and researchers have adopted these principles for a long time. In designing “Stitching the Loop,” we hope we have made them explicit so that other teachers and designers can adopt them for bringing constructionist activities to classrooms and promoting more equitable teaching and learning opportunities for students. Though afterschool, out-of-school, and online constructionist experiences have much to offer, we believe that classrooms provide unique opportunities to reach out to broader numbers of children and youth who may not take the initiative to step into those more informal experiences. Further, classrooms furnish circumstances that can support greater rigor and depth because of consistent attendance and dedicated time to projects.

In return we must of course consider the constraints of classrooms themselves, with the need to promote certain academic content and practices, limited staff, and physical and material constraints. Teachers report that “Stitching the Loop” has been a tremendous success with student engagement and preparation for more advanced computing courses. It provides a proven example that one teacher can work to support personalized project-creation with 25, 35 and even 40+ students. Projects with creative constraints, peer pedagogy, and process-based reflection all support depth of learning while legitimizing learners’ expertise and supporting interest-driven engagement. Robust professional development, building on the ECS model and educating in the way we hope teachers will educate their own students, is a key factor in ensuring the *design principles* are implemented fully.

Of course, challenges remain. Supply costs for the unit have become more reasonable but not all schools can afford them consistently. Inevitably, not all teachers will embrace the principles of the unit equally, resulting in inconsistent implementation. This means we need to support teachers beyond the first year or two of implementation now that the curriculum is ready for national release. In developing “Stitching the Loop,” we illustrated how guiding principles need to apply to the design of construction tools and kits as well as to the constructionist projects and activities in which they are employed. Only then can we provide personally meaningful and equity-minded experiences to all learners.

Acknowledgment

This work was supported by a grant #1509245 from the National Science Foundation to Yasmin Kafai, Jane Margolis, and Joanna Goode. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, the University of Pennsylvania, or Utah State University. Special thanks to Tomoko Nakajima, Debora Lui, Justice Walker, Gayithri Jayathirtha, and Mia Shaw for their help with data collection and analysis.

References

- Blikstein, P. & Worsley, M. (2016). Children are not hackers: Building a culture of powerful ideas, deep learning, and equity in the Maker Movement. In K. Peppler, E. Halverson, & Y.B. Kafai (Eds.), *Makeology: Makerspaces as learning environments* (pp. 64-79). New York, NY: Routledge.
- Brennan, K., Balch, C. & Chung, M. (2014). *Creative Computing*. Harvard Graduate School of Education. Retrieved February 21, 2018 at <http://scratched.gse.harvard.edu/guide/download.html>
- Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*, 7(2), 12-15.
- Reynolds, R., & Caperton, I. H. (2011). Contrasts in student engagement, meaning-making, dislikes, and challenges in a discovery-based program of game design learning. *Educational Technology Research and Development*, 59(2), 267-289.
- Fields, D. A., Quirke, L., Horton, T., Velasquez, X., Amely, J. & Pantic, K. (2016). Working toward equity in a constructionist Scratch camp: Lessons learned in applying a studio design model. In A. Sipitakiat & N. Tutiyaiphuengprasert (Eds.), *Proceedings of Constructionism 2016* (pp. 290-297). Bangkok, Thailand: Suksapattana Foundation.
- Goode, J., Chapman, G., Margolis, J. (2012). Beyond curriculum: The Exploring Computer Science program. *ACM Inroads*, 3(2), 47-53.
- Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, 1(1), 1-32.
- Hetland, L., Winner, E., Veenema, S., and Sheridan, K. (2013). *Studio thinking 2: The real benefits of visual arts education*. New York, NY: Teachers College Press.
- Jayathirtha, G., Fields, D. A., & Kafai, Y.B. (2018). Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects. *Conference Proceedings of International Conference on Computational Thinking Education 2018*, Hong Kong: The Education University of Hong Kong.
- Kafai, Y. B. (1995). *Minds in play: Computer game design as a context for children's learning*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Kafai, Y. B. Fields, D. A., & Searle, K. A. (2014). Electronic textiles as disruptive designs in schools: Supporting and challenging maker activities for learning. *Harvard Educational Review*, 84(4), 532-556.
- Kafai, Y. B., Lee, E., Searle, K. S., Fields, D. A., Kaplan, E., & Lui, D. (2014). A crafts-oriented approach to computing in high school. *ACM Transactions of Computing Education*, 14(1). 1-20.
- Lui, D., Walker, J. T., Hanna, S., Kafai, Y. B., Jayathirtha, G., & Fields, D. A. (2018). Communicating computational concepts and practices within high school students' portfolios of making electronic textiles. In proceedings of the *International Conference of the Learning Sciences*, London, UK.
- Margolis, J., Estrella, R., Goode, J. & Holme, J. & Nao, K. (2017). *Towards the Shallow End (revised edition)*. Cambridge, MA: The MIT Press.
- Papert, S. (1980). *Mindstorms*. New York, NY: Basic Books.
- Papert, S., & Solomon, C. (1971). *Twenty things to do with a computer*. Artificial Intelligence Memo 248. Cambridge, MA: MIT AI Laboratory.
- Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Horses, I. H. M., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K. & Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education*, 16(2), Article 11. DOI=<http://dx.doi.org/10.1145/2700517>
- Resnick, M. & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children* (pp. 117-122). New York, NY: ACM.

Learning Analytics in Education: Objectives, Application Possibilities and Challenges

Aleksandra Klašnja-Milićević, *akm@dmi.uns.ac.rs*

Faculty of Sciences, University of Novi Sad, Serbia

Mirjana Ivanović, *mira@dmi.uns.ac.rs*

Faculty of Sciences, University of Novi Sad, Serbia

Abstract

With an increased amount of educational data in all domains of human activities, Learning Analytics (LA) become a powerful mechanism for supporting learners, instructors, teachers, learning system designers and developers to better understand and predict learners' needs and performances. In this paper, we analyse the significant dimensions and objectives of LA, application possibilities and some challenges to the beneficial exploitation of educational data. We identify the required skills and capabilities that make meaningful use of LA techniques and technologies in this domain. They can act as a useful guide for setting up LA services in support of educational practice and learner guidance, in quality assurance, curriculum development, and in improving learning process effectiveness and efficiency. Furthermore, this paper proposes the most important constraints that affect LA technologies in education.

Keywords (style: Keywords)

learning analytics; education; application; learning environments; effective learning system

Introduction

During the last years, developments of new learning technologies, devices and environments, such as digital learner records, learner cards, mobile and eye-tracker devices, sensors, flexible classroom design, and Massive Open Online Course (MOOC) are completely transforming the approach of learning and teaching (Sharple, et. Al, 2014). Higher education institutions are collecting more data than ever before. Management of this vast amount of data, named "big data", should offer valuable comprehension about the learning process, insights about risk of learner's dropping out, and support for increasing learners' success. In order to comprehend the patterns of value that exist within the large amount of data, new or innovative approaches are required. Lot of exploration and researches aim to handle the data with the proper techniques and new tools to produce real time solutions and predictions in this certain area. The utilities of proper techniques and new tools could be: operative self-learning, useful peer groups, available class time for creativeness, and possibilities for problem solving (Papamitsiou and Economides, 2014).

Analysis techniques that extract information from "big data" such as discovering patterns and applying them to the education process are named Learning Analytics (LA). In its initial steps of evolving, there has been numerous definitions used for LA. Siemens defines it as "the use of intelligent data, learner product data and analysis models to discover information and social connections, and to predict and advise on learning" (Siemens, 2012). Elias described it as "an emerging field, in which sophisticated analytic tools are used to improve learning and education" (Elias, 2011). Learners and teachers leave many traces behind them in which LA can convert them to be beneficial for the education sector (Duval, 2011). The most cited definition of LA came from an open online course on learning and knowledge analytics (LAK11) and was adopted by the associated First International Conference on LA and Knowledge in 2011 (Long, 2011): "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs".

Many researchers and developers study the use of LA in different fields related to the data availability, the competence, applicability, the cost, the privacy, the relevance and the ownership (Adams Becker,

2017; xxx et al., 2017). In this paper we will concentrate on contemporary study examines the applicability of LA in higher education institutions.

The rest of the paper is organized as follow. The second section gives an overview of key trends in education environments, dataset and analytics. The third section is devoted to educational LA applications and their role in contemporary education. The challenges of implementation of LA in education environments are pointed out in the section 4. The fifth section of the paper provides the concluding remarks and possible future directions.

Educational data - benefits for Learning environments

LA can obtain advantages of available educational datasets from different Learning Management System (LMS), Content Management System (CMS), Tutoring system and other learning proposed systems (Davies, et al. 2017). Educational Institutions already have a large amount of student data and use these for different purposes. Administering student progress and reporting to receive funding from the public authorities are the most commonly known purpose. Linking such available datasets would facilitate the development of mash-up applications that can lead to more learner-oriented services and therefore improved personalization. LA strongly relies on data about learners and one of the major challenges LA researchers are facing is the accessibility of publicly available datasets to evaluate their LA methods. Most of the data produced in institutions is protected, and the protection of student data and created learning artefacts is a high priority for IT services departments. Nevertheless, analogous to Open Access publishing and related movements, calls for more openness of educational datasets have already been brought forward (Drachler et al., 2010).

Greller and Drachler (2012) developed a framework that includes six critical dimensions related to an LA initiative. Each of the dimensions must be addressed to institutionalize an LA initiative successfully. Figure 1 illustrates critical dimensions according to educational environments.

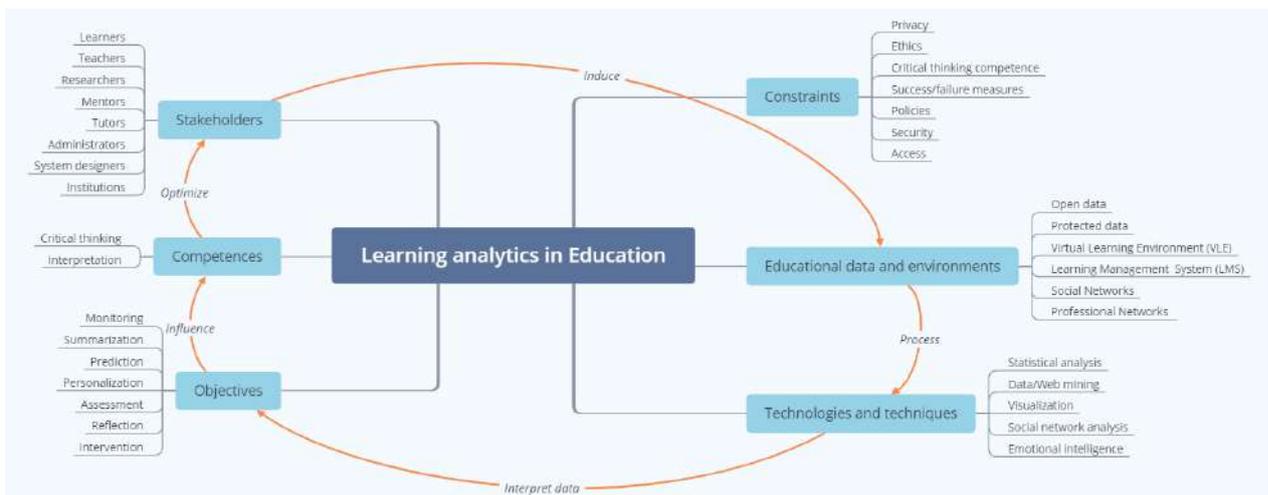


Figure 1. Critical dimensions of LA

Some researchers specified that higher education has to include the analytics tool into the system in order to improve productivity (Daniel 2016; Hrabowski et al. 2011; Becker 2013). First, data analytics software can provide feedback to learners and teachers about educational performances:

1. Feedback: Learner often might fail at a subject but not know what the reason was. It becomes valuable when the learner can look not just at himself, but at other people who have had the same experience. (S)he can get an insight either that would describe it so (s)he is not frustrated or that (s)he could use to correct it so that (s)he could be successful again. The improvement of electronic learning modules supports evaluation of learners in logical, real-time ways. In order to predict learner outcomes such as dropping out, needing extra help, or being capable of more demanding assignments, this

approach can analyze underlying patterns. Pedagogic approaches that seem most effective with particular learners could be identified.

2. **Tracking:** In order to understand the real patterns of learners more effectively tracker devices can be used for teachers, by allowing them to track a learner's experience in an e-learning course. In observing the digital paths learners leave overdue. Teachers can track learners' passage during the whole learning experience.

3. **Efficiency:** LA can save many hours of time and effort, when it comes to the achievement of our goals and strategies that we need to reach them.

4. **Understanding the learning process:** By using LA in e-learning, teachers can see which parts of a course were too easy and which parts were so difficult that the learner has failed to solve. Other parts of the learner's path teachers can analyse after that and consider pages re-entered often, preferred learning styles, sections recommended to peers, and the time of day, when learning operates at its best.

5. **Collaboration:** Experts from many different fields have to come together to retain a Learning Management System function at its best. This encourages cooperation, teamwork, and interdisciplinary thought processes.

6. **Personalization:** LA can be successful in the way we approach e-learning design by allowing designers to personalize courses to adjust their learners' individual needs. This will allow e-learning developers to promote the standard for effective and exceptional e-learning courses.

Regardless of the educational performances, many faculties, and universities have confirmed that analytics can support significant improvement of an institution, including resource distribution, learner achievement, administration, and finance. Some important features that LA offers to institutions are listed below (Daniel, 2015; Siemens, 2011).

- Assisting in creating common sense of complex topics through the combination of social networks and technical and information networks. Algorithms can recognize and provide insight into data and at-risk challenges.
- Innovating and transforming the college and university system, in addition to educational models and pedagogical approaches.
- Improving administrative decision-making and organizational resource provision.
- Helping leaders transition to holistic decision-making through analyses of "what-if" scenarios.
- Exploring how different components within a complex discipline (e.g., remembering learners, decreasing costs) connects and discovers the influence of varying essential components.
- Increasing administrative efficiency and productivity by providing latest information and allowing fast reaction to challenges.
- Testing and evaluation of curricula.
- Helping official leaders to control the hard (e.g., research, patent) and easy (e.g., quality of teaching, reputation, profile,) value created by faculty activities.
- Evaluating typical grading techniques and instruments (i.e., departmental and licensing exams).

Potential Application Areas of Educational Learning Analytics

Nowadays, researchers in educational LA field tries to answer increasingly important and complex questions: what a student knowledge is and whether a student is engaged in learning process. Scientists and researchers have analysed and experimented with new techniques and new kinds of learning system data that have shown promise for predicting student learning outcomes (Jivet et. al. 2018, Adams Becker et al, 2017). In this section we present review of potential application areas for Educational LA. These areas represent the broad categories in which LA can be applied regarding online activities. Figure 2 presents a summary of selected areas and appropriate data types needed for analysis. In summary, LA systems apply models to answer such questions as:

- When are students ready to move on to the next topic?
- When are students falling behind in a course?
- When is a student at risk for not completing a course?

- What grade is a student likely to get without intervention?
- What is the best next course for a given student?
- Should a student be referred to a teacher for help?

Responding to these questions involves the collection of students' entry and correctness of the answer, student activities on learning systems over time, when and to which group is specific for learning strategy, and performance of students on pre / post-tests.

Research on the various components of the learning system can contribute to the design of better learning systems as it has strong implications for student learning. This area represents a key point for educational data processing and LA (Baker, 2010). Haixiang et al. (2017) proposed learning decomposition as an alternative and effective method. Learning decomposition method involves exponential learning curves to performance data and relating student success to the amount of each type of pedagogical support. The weights indicate how effective each type of pedagogical support is for improving learning.

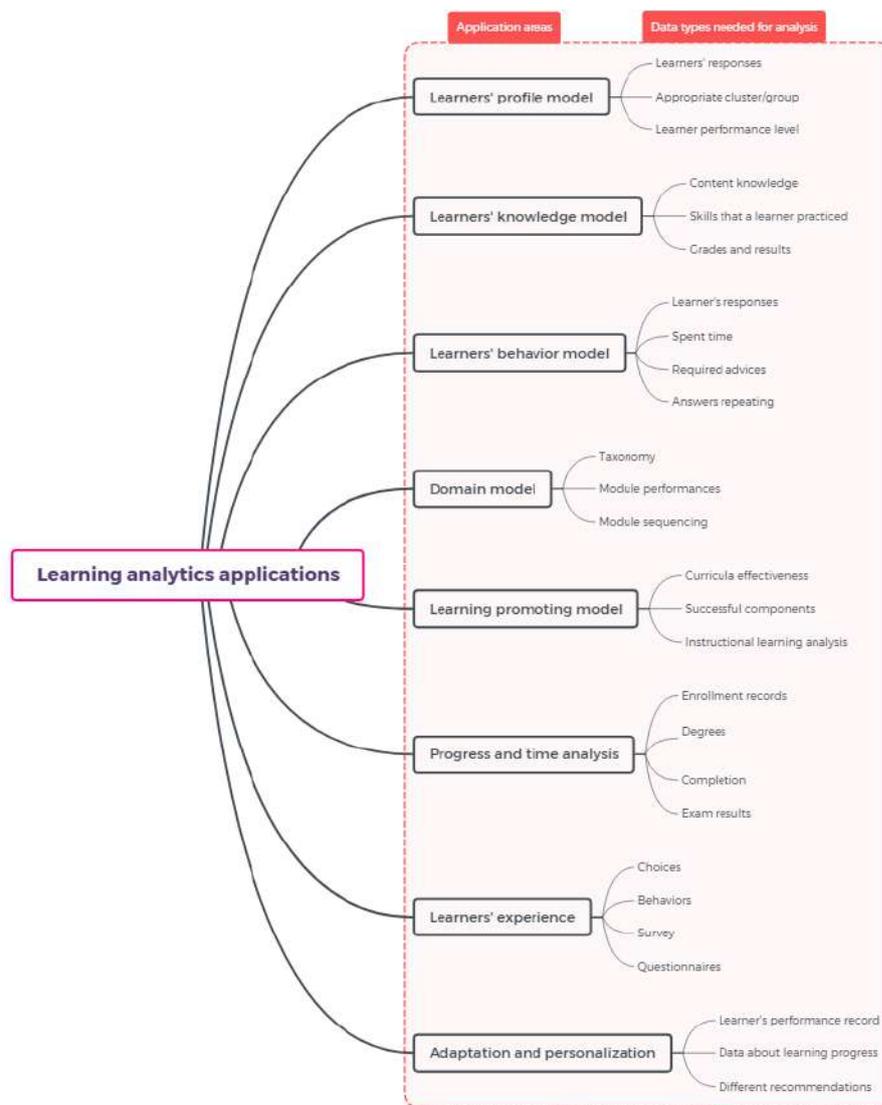


Figure 2. Application areas for Learning Analytics and data types needed for analysis

Scientists from Carnegie Mellon University have built cognitive models of mathematics, which have become the basis for high school programs that include intelligent tutorials (Ritter et al., 2007). In these systems, complex tasks are divided into individual components of knowledge, and the model is used to

improve the learning process and the problem-solving strategy. Each student activity is associated with one or more skills. In this way, researchers can use data from the tutorial system to dynamically assess the teaching efficiency. Evaluations and improvements to this model have been carried out over the last 15 years.

Challenges of Learning Analytics Technologies in Higher Education

Regardless of the fast development of applications that support implementation of big data techniques in higher education, there are also several apprehensions that affect LA technologies. The main issues that can be observed in the application of LA in education are related to data profiling, privacy, and the rights of learners with respect to their individual behavior recording (Boyd, 2010). For example, some important issues must be taken into consideration (Picciano, 2012):

- Should learners be informed that their activity is being followed?
- How much information is necessary for faculty, students, parents, scholarships, and other issuers?
- In what manner should faculty affiliates act in response?
- Do learners have a requirement to look for support?

Protection should be undertaken to confirm that the well-known collections of personal data of learner transactions are not hurt individuals. One encouraging approach to resolving these issues could be masking the data at its source (Barlow, 2013). Masking is one type of creative approaches that will make large-scale applications of data possible while still protecting the confidence of students' and teachers' information. New performances and competences for software applications permit sensitive data to be masked at the database level, when brought into a data warehouse. It can be concluded that even if someone has acquired physical access to database, delicate information like social security numbers will still be confused.

With the adoption of LA in the educational fields, institutions are required to adjust their policies with legislative framework. Many institutional policies failed to fully reflect the ethical and privacy implications of LA (Prinsloo & Slade, 2013). In the rest of the section, we specify several possible principles that an ethical LA policy should describe:

1. Collection of personal information: sex, date of birth, address, ethnicity, occupational status, qualifications and study records.
2. The usage of personal information, if it is for the benefit of the students, such as predicting students' behavior and advices based on LA, or if it is for research reasons to achieve LA objectives.
3. Methodology of data collection either by the student's input him/herself or by other services, such as browser cookies.
4. Security principles for keeping the data protected.
5. A description of the time period of keeping learners' data and rules for deletion process. (Singer, 2014)

All LA tools should follow convenient security principles to keep the analysis results and the students' records safe from any threat (Anciaux et al., 2006).

- *The confidentiality property* promises that the data can never be accessed by an unauthorized access.
- *The integrity property* promises that the data cannot be watched, changed, or transformed.
- *The availability property* guarantees that the data should be available for authorized parties to access when needed.

A key component of protecting learners' information is encrypting their data in order to achieve the confidentiality concept. Encryption guarantees that only authorized people can use the data. Moreover, assuring confidentiality can include: invoking file permissions and granting a secure operating environment, while cryptographic hashing of datasets can assure the integrity property of students' records (Chen & Wang, 2008).

As LA is an emerging research topic in the field of Technology Enhanced Learning and a forthcoming trend (Ebner & Schön, 2013), accuracy and validity of information is highly questionable. Faults related to selection a wrong dataset, or not recognizing the component relevant to data will negatively affect the accuracy of the outcome (Krasnow Waterman & Bruening, 2014). Therefore, a wrong selection of educational dataset will lead to inaccurate results. The questions we can ask here are: What if LA results were wrong? And what if the predications or the interventions went wrong? Accordingly, LA would aim to provide guarantees that it's analyzing, and picking the data, fit quality criteria and produce an agreed level of accuracy.

Data protection and copyright laws are legal restrictions that limit the beneficial use of LA. Such legal restrictions are: limitations of keeping the data for longer than a specific period, which are regulated differently in each country; the data should be kept secure and safe from internal and external threats; data should be used for specific purposes and the results of any process should be as accurate as possible. The restrictions could be stronger when it relates to personal information. Applying social network analysis as a method of LA causes the adoption of personal information.

There are two main perspectives about who own the data: students and institutions. Jones et al. (2014) concluded that neither the students nor the institutions should win the ownership of the data. They suggested a hybrid module that merges both perspectives. Institutions can invest the students' data in analytics, develop new personalized learning platforms and benchmark their learning management system. While students want to enhance their learning and maintain their performance, they would like to ensure that their information is kept confidential. An uprising question we like to address here is: What if LA methods have to modify the students' data for prediction purpose.

Conclusion

Learning Analytics is a promising research field, which provides tools and platforms that influence researchers in Technology Enhanced Learning. Higher education institutions are applying LA to improve the facilities they provide and to improve observable and measurable learning outcomes (e.g. marks, success and awards, as well setback and obstruction) (Cooper, 2013). Nonetheless, this emerging new field lacks an approach that explains a complete overview of its processes. This research study revised the definitions of LA and the aspects that encourage its extension. This provides an entire overview consisting of: learning environment, dataset, analytics, constraints and the interventions which are interpreted to achieve the main goals of LA. Based on this approach, we identified the stakeholders, introduced examples of usage, presented methodologies and discussed the objectives. After that, we covered the way of determining the challenges that surround LA and shed the light on the privacy, security and ethical issues and anticipated questions that need a further research in near future.

In the future, educational institutions should try to balance the institutions own philosophy of learner development on the one side and various federal privacy laws on the other side. It is significant that organizations comprehend the dynamic nature of educational success and retaining, offer surroundings for open dialogue, and enhance practices and approaches to solve these issues.

We hope that research achievements presented in this paper and the implications derived from them will advance the discussion about building effective learning systems for learners, instructors, course designers, and similar activities in educational institutions.

References

- Adams Becker, S., Cummins, M., Davis, A., Freeman, A., Hall Giesinger, C., & Ananthanarayanan, V. (2017). NMC Horizon Report: 2017 Higher Education Edition.
- Anciaux, N., Bouganim, L., & Pucheral, P. (2006). Data confidentiality: to which extent cryptography and secured hardware can help. In *Annales des télécommunications* (Vol. 61, No. 3-4, pp. 267-283). Springer-Verlag.
- Baker, R. (2010). Data mining for education. *International encyclopedia of education*, 112-118.
- Barlow, M. (2013). *Real-time big data analytics: Emerging architecture*. " O'Reilly Media, Inc."

- Becker B., (2013). Learning analytics: insights into the natural learning behavior of our students, *Behav. Soc. Sci. Librarian*, 32,63–67.
- Boyd, D. (2010). Privacy and publicity in the context of big data. In *Keynote Talk of The 19th Int'l Conf. on World Wide Web*.
- Chen, L. & Wang, G. (2008). An efficient piecewise hashing method for computer forensics. In *Knowledge Discovery and Data Mining, 2008. WKDD 2008*. (pp. 635-638). IEEE.
- Daniel B. (2015). Big data and analytics in higher education: Opportunities and challenges, *Brit. J. Educ. Technol*, 46 904–920.
- Daniel, B. K. (Ed.). (2016). *Big data and learning analytics in higher education: current theory and practice*. Springer.
- Davies, R., Nyland, R., Bodily, R., Chapman, J., Jones, B., & Young, J. (2017). Designing technology-enabled instruction to utilize learning analytics. *TechTrends*, 61(2), 155-161.
- Drachsler, H., Bogers, T., Vuorikari, R., Verbert, K., Duval, E., Manouselis, N., & Wolpers, M. (2010). Issues and considerations regarding sharable data sets for recommender systems in technology enhanced learning. *Procedia Computer Science*, 1(2), 2849-2858.
- Ebner, M., & Schön, M. (2013). Why learning analytics in primary education matters. *Bulletin of the Technical Committee on Learning Technology*, 15(2), 14-17.
- Elias, T. (2011). Learning Analytics: Definitions, Processes and Potential. Retrieved 10 March 2018 from <http://learninganalytics.net/LearningAnalyticsDefinitionsProcessesPotential.pdf>
- Greller, W. and Drachsler, H. 2012. Translating Learning into Numbers: A Generic Framework for Learning Analytics. *Educational Technology & Society*. 15, 3 (2012), 42–57.
- Haixiang, G., Yijing, L., Shang, J., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73, 220-239.
- Hrabowski F. A., Suess, and J. Fritz, (2011). Assessment and analytics in institutional transformation, *Assess. Analyt. Institut. Transform.*46, 14–28.
- Jivet, I., Scheffel, M., Specht, M., & Drachsler, H. (2018). License to evaluate: Preparing learning analytics dashboards for educational practice.
- Jones, K., Thomson, J., and Arnold, K. (2014). Questions of Data Ownership on Campus. *EDUCASE Review*. Retrieved 2nd January 2018 from <http://www.educause.edu/ero/article/questions-data-ownership-campus>
- Xxx, (2017).
- Krasnow Waterman, K., & Bruening, P. J. (2014). Big Data analytics: risks and responsibilities. *International Data Privacy Law*, 4(2), 89-95.
- Long, P. (2011). LAK'11: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, February 27-March 1, 2011, Banff, Alberta, Canada. ACM.
- Picciano, A. G. (2012). The evolution of big data and learning analytics in American higher education. *Journal of Asynchronous Learning Networks*, 16(3), 9-20.
- Ritter, S., J. Anderson, K. Koedinger, and A. Corbett. 2007. "Cognitive Tutor: Applied Research in Mathematics Education." *Psychonomic Bulletin & Review* 14 (2): 249–255.
- Sharples, M. et al. *Innovating Pedagogy*, (2014), 1–37.
- Siemens, G. (2012). Learning Analytics: Envisioning a Research Discipline and a Domain of Practice. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK 2012)* pp. 04-08, New York, USA: ACM.
- Slade, S., & Prinsloo, P. (2013). Learning analytics: Ethical issues and dilemmas. *American Behavioral Scientist*, 57(10), 1510-1529.

Assessing Learning through Exploratory Projects in Constructionist R-based Statistics Courses for Environmental Science Students

Maite Mascaró, *mmm@ciencias.unam.mx*

UMDI-Sisal, National Autonomous University of Mexico (UNAM), Mexico

Ana Isabel Sacristán, *asacrist@cinvestav.mx*

Centre for Research and Advanced Studies (Cinvestav), Mexico

Abstract

For years we have been developing sequences of constructionist and collaborative activities where university environmental science students engage in computational programming tasks (using R code –<http://www.r-project.org/>) for exploring and learning statistical concepts and ideas. In this practice paper, we present a project-based method that we use for assessing students' learning and appropriation of those statistical concepts.

The exploratory projects that we use for assessment, are open-ended realistic problems in environmental contexts. In these projects, students, working collaboratively in teams of up to 4 members, are expected to make choices of the most appropriate statistical procedures and graphic representations to be used in portraying relevant statistical features in the data, and construct statistical models using any tool they choose, such as R code. They then have to defend the line of reasoning underlying their choices, through a formal presentation using their own communication resources, and where other students question and discuss the methods, results and interpretations presented. Such projects can be considered to be thought-revealing activities; i.e. they serve as a window into students' thinking, and thus are means for assessing their learning. They also constitute an approach that is more in accordance with constructionist principles, since students explore and construct freely. Such projects thus, not only serve as a means for us to assess students' learning, but can be, in themselves, learning experiences that help reinforce what has already been learned.

Keywords

statistics education; assessment; exploratory project; R-programming; constructionism

Introduction

For approximately 8 years, we have been researching how to enhance the teaching and learning of statistics and experimental analysis for environmental and biological sciences students at university level, using the constructionist approach (Papert & Harel, 1991). As we expressed in Mascaró & Sacristán (2016, p. 102):

our concern is that environmental scientists require probability and statistical knowledge for experimental data analysis, decision-making, evidence to support theory, and communication of scientific results, so it is important for future researchers to have adequate understanding of these topics. However, learners in university programs in biology and related sciences tend to have difficulties in understanding and/or applying the concepts, and even researchers have a poor understanding and use incorrectly many statistics concepts (Batanero, 2001). Many teaching programs have had unsuccessful results (Bishop & Talbot, 2001) and there tends to be an aversion (including general 'mathophobia' – Papert, 1980) towards learning statistics.

In order to address this concern, we have developed sequences of constructionist and collaborative activities where students engage in computational programming tasks (using R code –<http://www.r-project.org/>) for learning and exploring statistical concepts. However, we then face the issue of how to assess the learning that takes place through our approach, without reverting fully to more traditional schooling methods, but rather utilizing means that are more in accordance with constructionist

principles. In this practice paper, we focus on a project-based method that we use for assessment purposes. Before that, we summarize the approach we take in our courses.

The R-based statistics course approach

Using as framework the constructionist philosophy (Papert & Harel, 1991) –which suggests that learning can be facilitated if students engage in exploring through construction, such as programming– as stated above, we have been developing and refining a series of approximately 35 student-centred computer programming activities for students to explore, implement and develop meaning for statistical concepts and ideas (see Mascaró, Sacristán & Rufino, 2016; Mascaró & Sacristán, 2016). The activities also attempt to integrate the following aspects: (i) the use of concrete examples with data from real research situations; (ii) emphasis on the use of diagrams, since graphic representations are essential in data organisation, statistical reasoning and analysis (Wild & Pfannkuch, 1999).

All the activities are based and presented in the R programming language through R-code “worksheets” with instructions, examples, programming tasks, questions for reflection, and comments. We chose R, “because it is a transparent and powerful expressive ‘Logo-like’ language commonly used by statisticians, but simple to use (using a set of intuitive and relatively simple commands)” (Mascaró & Sacristán, 2016, p. 102). The aims are for students to apply and explore statistical functions and concepts, and develop meanings for them, through the R-programming tasks; as well as develop competency in a statistical analysis tool (R) that they can use in their future research activities.

The activities are usually carried out in teams of 2-3 students and moderated by a teacher. They are of two types: those for introducing the basics of R; and those on various topics of statistics courses (e.g. activities that deal with frequency distributions; activities on binomial, Poisson, and normal distributions; activities on ANOVA and comparisons between means; activities on linear regression; etc.). A sample task in ANOVA can be seen in Mascaró et al. (2016).

The different activities have been implemented, repeatedly over the years, in at least two undergraduate courses (*Probability and Statistics; Experimental Planning and Analysis*) and three post-graduate ones (*Experimental Design and Data Analysis; Introduction to Multivariate Statistics; and Univariate Statistics*): in total, in approximately 20 courses (counting repetitions) in Mexico and Portugal.

Assessing the learning and appropriation of statistics

In order to assess students’ learning and appropriation of the statistical concepts and ideas, one approach that we have developed is through exploratory projects, carried out by teams of students, for solving realistic problems in environmental contexts. In these projects, students need to make choices of the most appropriate statistical procedures and graphic representations to be used in portraying relevant statistical features in the data, serving both exploratory and confirmatory purposes. They then have to defend the line of reasoning underlying their choices, through a formal presentation using their own communication resources, where other students question and discuss the methods, results and interpretations presented.

These projects are similar to the learning programming activities of the lessons (including the collaborative work by teams of students), except that they are completely open-ended. In that sense, they are even more constructionist than the learning activities themselves. These projects share some commonalities with what Lesh et al. (2000) called thought-revealing activities (or model-eliciting activities) in the sense that students are faced with an open-ended realistic problem which they need to explore, and for which they need to build statistical models with whatever tools and methods they consider adequate, pose questions and present their conclusion in reasoned ways. Moreover, as Noss and Hoyles (1996) proposed, computer-based activities can serve as a window onto mathematical meaning-making. In those ways, students’ thought processes are revealed; we can thus assess the way in which students are able to create meanings and apply statistical concepts in solving the problems.

Phases of the assessment exploratory projects

An assessment session centred on an exploratory project, consists of three phases:

1. Each team (maximum 4 students) is given a concise, yet informative, description of a problem to be explored and solved, the context and the data produced by the investigation stated in the problem, together with an indicative list of important aspects to be considered as a guide for its solution. Teams are given 90 minutes to work on the problem, during which they have open access to every source of information: R scripts of their own, as well as those found on the internet; online documents on statistics, blogs, books, class notes and corrected exercises and homework.
2. Teams are then given 30 minutes to prepare a 15-slide presentation of their findings. Presentations are expected to include a short introduction of the problem and research questions, a description of the exploratory and confirmatory techniques used, tables with statistical descriptors, numerical and graphic representations of relevant features in the data, and final conclusions.
3. After a short break, teams present their results in random order. After each presentation, students in the other teams have to pose at least two questions that are graded depending on the degree of complexity and pertinence. Answers by the presenting team are also graded depending on precision, extensiveness and appropriateness. Students are previously instructed to ask questions in relation only to the statistical procedures and interpretations of the results obtained (thus knowledge on other contextual aspects –biological, chemical, geographical or social concepts– whilst superficially necessary for the discussion, is not considered for the grade).

Assessment criteria

During the presentations, members of each team are graded according the following categories using a scale of 1 to 3. (Note: we need to score students' work, since we still are working in a formal school setting):

- A. How the problem and its general context are expressed and presented, and the explanation that is given on the experiment that was carried out
- B. Identification of the hypothesis of the problem (the cause-effect relationship)
- C. Identification of the factors (types, number of levels, arrays), of the response variables (units, SU, number and balance)
- D. Clear identification of the statistical model to be used and translation of the hypothesis of the problem, as expected from the different variation sources.
- E. Exploration of the data (trends of means, variances, internal distributions, extraordinary data)
- F. Execution of the analysis (types of tests and programmatic tools)
- G. Statistical interpretation of hypothesis tests
- H. Interpretation in the terms of the problem and the scope of the results
- I. Support elements to explain the results (use of tables, diagrams, graphs, etc.)

Questions from other teams and answers during the discussion stage are also marked from 1 to 3 according to the following criteria:

1. Complexity of the question
2. Articulation of the question
3. Adequate question
4. Quality and efficiency in the response
5. Elaboration of the answer

Grades in criteria A – I, are worth 60%; whereas those in criteria 1 – 5, 40% of the total score.

Sample assessment related to Factorial ANOVA

We now present an example of an exploratory evaluation project used to assess the way in which students apply statistical concepts related to factorial ANOVA for solving an environmental problem. As

can be seen in the text given to students (Figure 1) for the sample project discussed here, there are no specific questions but rather some guiding aspects to consider; so the project was open-ended and was specifically directed to assess whether students were competent in:

- Identifying situations where the estimation and prediction of variance components and interaction terms between categorical explanatory variables (factors in nested and orthogonal ANOVA designs) is required.
- Identifying the cause-effect relations of all explanatory and response variables involved, and the corresponding test hypotheses that need to be specified to confirm such relations.
- Distinguishing between random and fixed effects in factorial designs, the computation of variance components and its interpretation in the context of the problem.
- The application of test hypotheses using the F statistic, and the identification and interpretation of different elements in the numerical output of such tests i.e. degrees of freedom, residual standard error, probability of F value, coefficient of determination, etc.
- Assessing the magnitude of estimates in the statistical model, particularly those related to nested factors and interaction terms.
- The construction of graphic representations used to portray relevant statistical features in the data that will serve both exploratory and confirmatory purposes.

Programming objectives were to learn, or improve, the use of the following R functions and libraries:

- R-package GAD to adjust balanced models with different combinations of fixed and random factors in nested and factorial ANOVA designs.
- The 'anova' function and compare its performance to that of 'gad'.
- The F-Distribution 'pf', 'qf', 'df' and 'rf'.
- The 'interaction.plot' and 'coplot' functions used to build working graphs that help assess interactions between explanatory variables.
- Functions in the R-package 'ggplot2' to build graphs with different elements used often for final visualizations of statistical models.

Specific example of a problem to be explored for the assessment

Below (Figure 1), we present the text of a problem that guided one of the assessment projects.

River San Pedro is the main body of superficial water in the Mexican state of Aguascalientes. It runs in an almost straight line from North to South through 90 km, until it meets River Santiago in the state of Jalisco. Prior to the construction and launching of the Industrial Park of the Valley of Aguascalientes, clustered in the southernmost part of the state, water pollution in the river, the aquifer and the soil in the river banks was negligible. Today, however, contamination by toxic organic substances, such as phenols, anilines and sulfacetamide (SAAM) has been reported as dramatic.

In the attempt of confirming the environmental impact of organic toxins in the water, researchers from the Laboratory of Environmental Studies of University of Aguascalientes conducted a study repeating the same sampling design that their colleagues had used before the region's industrialization (1975). This consisted of taking 15 random water samples in 5 stations along the river, covering from the far north end of the industrial park (Sit1), an intermediate (Sit2), one near the industrial park (Sit3), one within the park (Sit4), and finally, one where most industries are located (Sit5). The concentrations of anilines (mg L⁻¹) were determined in each water sample following the same procedures used previously, despite the idea that the old determinations were less precise than modern ones would be.

The data can be found in sheet 'anilines' of file 'ex 2.10.xlsx'.

Guide of important aspects to consider when solving the problem.

- a) Cause-effect relations and derived hypotheses.*
- b) Type of statistical model, categorical factors included, level per factor and arrangement in the experimental design.*
- c) Types of variables, experimental units, replication and balance in the model.*

- d) Strategy for the statistical analysis, including exploratory data analysis and the type of comparisons amongst means necessary to fully respond to the research questions posed in problem.
- e) Interpretation of results, both statistic and in the context of the problem.
- g) Presentation of results (format, graphs, tables, relevant statistical values, etc.), including final remarks and recommendations for future investigations.

Figure 1. The problem given to students for an assessment related to Factorial ANOVA

Sample answer by a team of students

We discuss here the presentation created by a 4-member student team, after their exploration of the above problem, in one of courses in the Sustainable Coastal Zone Management undergraduate degree at the National Autonomous University of Mexico. This team got an overall score of 88/100 – the highest of the group. The team unequivocally identified the cause-effect relationship, the corresponding explanatory and response variables, and the fixed nature of the former; the experimental unit, number of replicates and levels in both factors (Figure 2).

Factores			
Tiempo	a=3	l=1,2,3	Fijo
Estación	b=5	j=1,2,3,4	Fijo
Estación *Tiempo	ab=15	k=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15	Fijo

- ▶ Variable de respuesta=Concentración de anilina mg/L
- ▶ Unidad experimental=muestra de agua
- ▶ N=15
- ▶ Modelo balanceado

Figure 2. The factors of the problem from the student’s team slide

Hipótesis

En términos generales.

- ▶ $H_m = \mu + A_i + B_j + A*B(i,j) + E_{ijkl}$
- ▶ $H_o = \mu + e_{ijk}$

Existen 3 fuentes de variación: Tiempo, Sitio, Interacción TS y la residual.
En realidad lo que plantea el problema es que el tiempo y el sitio afectan la concentración de anilina de manera dependiente (con diferente razón de cambio).

SITIOS

- ▶ $H_{me1} = X_{t1} > X_{t2}$
- ▶ $H_{me2} = X_{t1} > X_{t2}$
- ▶ $H_{me3} = X_{t1} > X_{t2}$
- ▶ $H_{me4} = X_{t1} > X_{t2}$
- ▶ $H_{me5} = X_{t1} > X_{t2}$

- $H_{oe1} = X_{t1} = X_{t2}$
- $H_{oe2} = X_{t1} = X_{t2}$
- $H_{oe3} = X_{t1} = X_{t2}$
- $H_{oe4} = X_{t1} = X_{t2}$
- $H_{oe5} = X_{t1} = X_{t2}$

MODELO ORTOGONAL- BIFACTORIA

Para el tiempo1 se busca ACEPTAR la nula

- ▶ $H_{mt1} = X_{s1} = X_{s2} = X_{s3} = X_{s4} = X_{s5}$
- ▶ $H_{ot1} = X_{s1} = X_{s2} = X_{s3} = X_{s4} = X_{s5}$

Para el tiempo2 se busca RECHAZAR la nula

- ▶ $H_{mt2} = X_{s1} = X_{s2} = X_{s3} = X_{s4} = X_{s5}$
- ▶ $H_{ot2} = X_{s1} = X_{s2} = X_{s3} = X_{s4} = X_{s5}$

Figure 3. The hypothesis of the problem as presented in the students team’s slides

The team was also clear and resourceful in the description of the problem, supporting the introduction with a before-and-after diagram to explain how the impact assessment experiment had been conducted. They carried out an exploration for comparing, as an appropriate control, the data amongst years and the use of time, something that was straightforward; this allowed them to express a general hypothesis (in this case of differences in aniline concentration between years), (see Figure 3). Whilst stating that “both time and site would affect aniline concentration in a dependant way (with differences in rate of change), there was some confusion in terms of which mean values would differ from each other. Despite some contradiction, a final relation amongst means per sites and years was appropriate, and the type of statistical model was correctly chosen, showing sufficient understanding of the need of an interaction

term to be assessed. The team applied several exploratory techniques to reveal features relevant to the problem and the data, including elements such as the magnitude and trend in the difference between means, size of internal dispersion, absence of outliers, etc. An important part of that exploratory analysis, is the construction of adequate graphical representations. In this case, the students used R to construct a variety of graphical resources (Figure 4), which they used effectively for describing adequately the elements of the problem and their interactions.

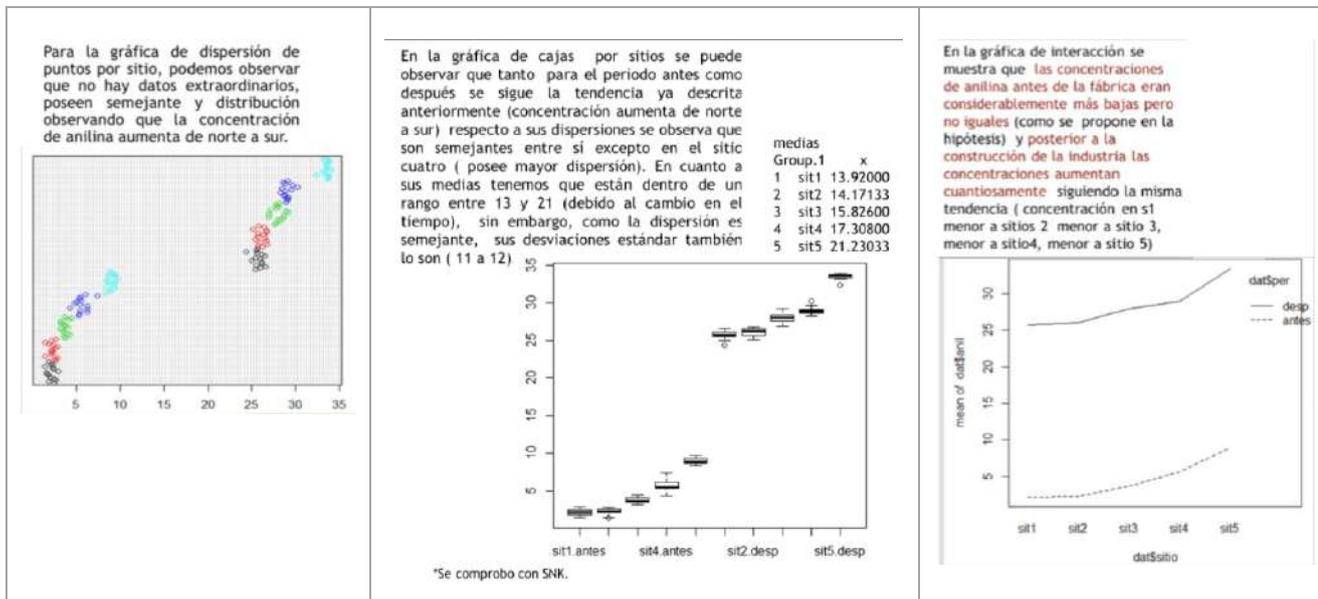


Figure 4. The students team created different graphical representations to present the data of the problem: a) a scatter plot, b) a box plot, and c) line graphs for the sample interaction.

This team of students carried out several statistical tests, such as ANOVA (see Figure 5) complemented and verified through a SNK test (Figure 6), which were correctly interpreted. They confirmed the hypothesis that the aniline concentration is dependent on both time and site. Nevertheless, whilst being able to recognize the need to assess the interaction term from the beginning, their strategy lacked consistence when they insisted on maintaining the analysis of the two main terms in the model. Also, in the final visualization of the model, this team didn't portray the important conclusive elements regarding the interaction term drawn by them in their analysis, showing only differences amongst years; however, the model was correctly executed and included a measure of dispersion (Figure 7).

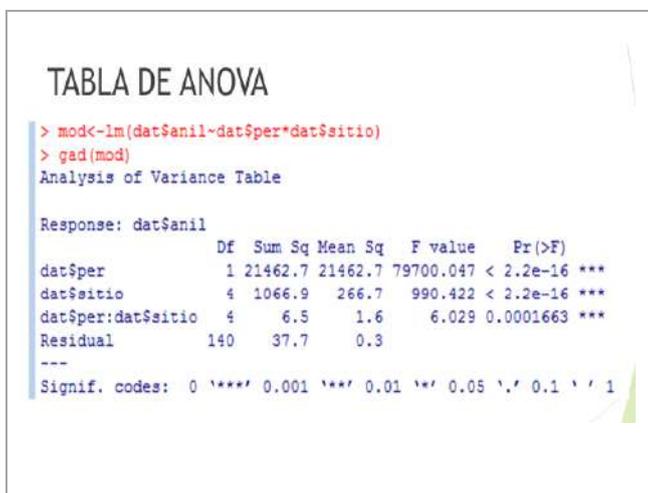


Figure 5. ANOVA table created by the team of students using R code

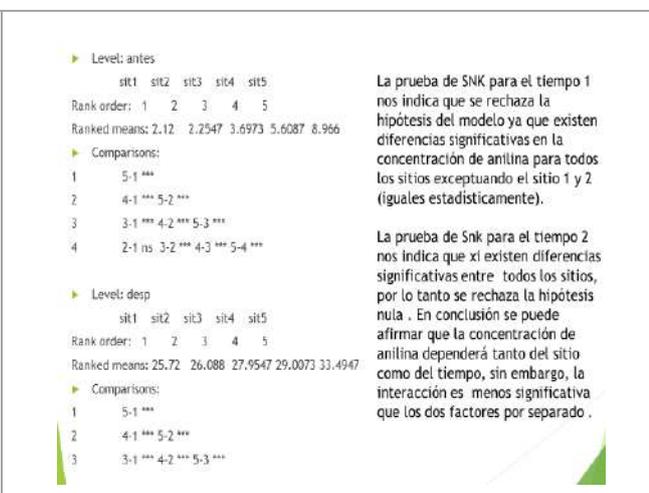


Figure 6. The SNK test carried out by the students team and their interpretation.

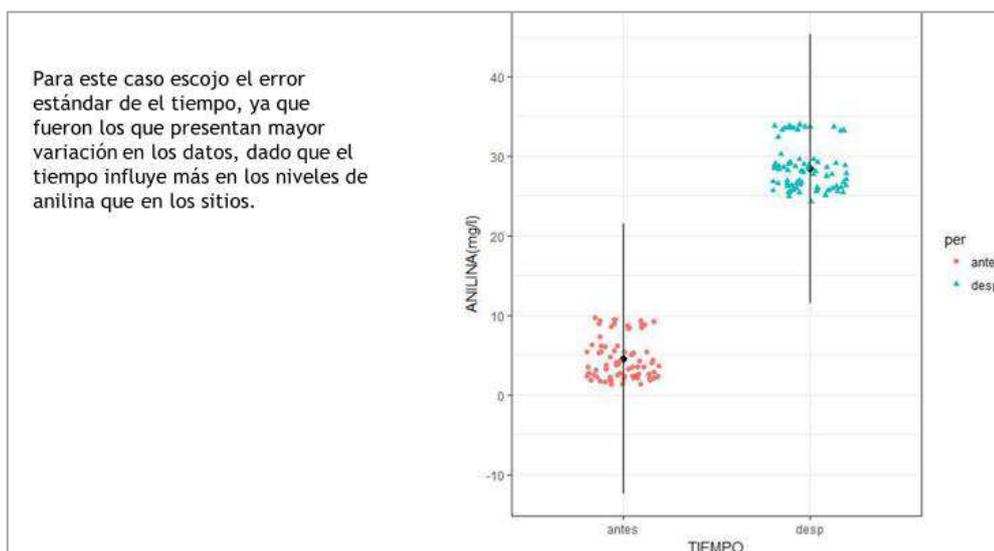


Figure 7. The team's choice for the final visualisation showing the differences in concentration over time with measures of dispersion.

So, in general, this team of students carried out a fairly successful exploration where they showed an understanding of the problem, were able to apply statistical concepts and generated adequate graphical representations and conclusions. This shows a good appropriation and learning of the concepts. They also used R programming for their statistical explorations, which shows, as well, that they appropriated that tool for their explorations. This example, although of the highest scored students, illustrates the potential of this type of approach for assessing students learning.

We did not present here examples of the questions posed by individual students; but it is another important dimension of that assessment approach, since it reveals individual students thinking versus that of the team.

Concluding remarks

In this paper we showed how we use exploratory problem-solving projects in realistic contexts, as well as using a collaborative approach, in order for students to reveal their appropriation of the statistical concepts and ideas as they apply them in exploring, solving and presenting their interpretations, statistical model and conclusions. We consider this assessment approach more in accordance with constructionist principles, since students explore and construct freely, in a way that is also more realistic: where they actually have to think and work, more as they will have to, in their future careers. These projects thus, not only serve as a means for us to assess their learning, but are, in themselves, learning experiences that help reinforce what has already been learned.

Acknowledgement

The work presented in this paper has been financed in part by the PAPIME PE204614 and PE207416 grants from DGAPA-UNAM.

References

- Batanero, C. (2001). Main research problems in the training of researchers. In C. Batanero (Ed.), *Training Researchers in the Use of Statistics* (pp. 385-396). Granada, Spain: International Association for Statistical Education and International Statistical Institute.
- Bishop, G. & Talbot, M. (2001). Statistical thinking for novice researchers in the biological sciences. In C. Batanero (Ed.), *Training Researchers in the Use of Statistics* (pp. 215-226). Granada, Spain: Intl. Association for Statistical Education and International Statistical Institute.

Lesh, R., Hoover, M., Hole, B., Kelly, A., & Post, T. (2000). Principles for developing thought-revealing activities for students and teachers. In A. Kelly & R. Lesh (Eds.), *Handbook of research design in mathematics and science education* (pp. 591-645). Mahwah, NJ: L Erlbaum.

Mascaró, M., & Sacristán, A. I. (2016). Exploring randomness and variability in statistics through R based programming tasks. In A. Sipitakiat & N. Tutiya-phuengprasert (Eds.), *Constructionism in Action: Conference Proceedings Constructionism 2016* (pp. 101–108). Bangkok, Thailand: Suksapattana Foundation. <http://e-school.kmutt.ac.th/constructionism2016/?p=772>

Mascaró, M., Sacristán, A. I., & Rufino, M. M. (2016). For the love of statistics: appreciating and learning to apply experimental analysis and statistics through computer programming activities. *Teaching Mathematics and Its Applications*, 35(2), 74–87. doi: 10.1093/teamat/hrw006

Noss, R. and Hoyles, C. (1996). *Windows on Mathematical Meanings*. Dordrecht: Kluwer.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books

Papert, S. & Harel, I. (1991). Situating Constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood., NJ: Ablex Publishing Corporation. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>

Wild, C. & Pfannkuch, M. (1999). Statistical thinking in empirical enquiry. *International Statistical Review* 67 (3), 223-265.

How Students Struggled with Preparation of Activities for a Leisure Time Robotic Workshop

Karolína Mayerová, *mayerova@fmph.uniba.sk*
Comenius University, FMFI, Bratislava, Slovakia

Michaela Veselovská, *veselovska@fmph.uniba.sk*
Comenius University, FMFI, Bratislava, Slovakia

Abstract

In this article we describe university compulsory elective course, which we taught in the winter semester of 2017. We briefly explain specific conditions, which encourage transformation of the organization of the course. The aim of this article is to provide an overview of the process and results of a case study on integrating authentic learning into mentioned course. Not only from point of view of usage robotic kits during course lessons but also from the point of view of creating specific activities for leisure time workshop. This workshop was taught by our student (let's call her Jane), who also conducted observations there. In our research we used qualitative methods of data collection and data analysis including observations, recorded video, students' work process and outcomes (created activities). We introduce several activities with remarks from conducted observations. These activities were created with constructionist approach by our students during the course. We present several findings from Jane, from the teacher' point of view and also from pupils' point of view. She identified several requirements for content of activities for robotic workshop, which should include solutions of all the tasks in the activities and methodological guidelines for teaching.

Based on Jane observations, we found out that number of pupils in the workshop was growing especially because of attractive nature of created activities. We hope that such increased internal motivation could help pupils to acquire a positive relationship to the compulsory school subject Informatics and to direct them in career growth.

Keywords

university elective course within teacher's training; educational robotics; after school activities for primary and lower secondary school pupils; authentic learning

Introduction

Teacher education in the field of robotics can have a positive impact on teaching practice, such as, i.e. teaching focused on students (Bers, 2007). However, there are only few studies that include the training of future teachers with educational robotics (Kim et al., 2015). Nonetheless in these studies there is a serious lack of information and systematic evaluation about teacher education using robotics in teaching (Kim et al., 2015). Spolaôr and Benitti (2017) conducted a systematic review of studies about educational robotics that is being used in higher education institutions (tertiary institutions) and found out that studies do not focus on practical experience embedded in learning theories such as constructivism (Strommen, Lincoln, 1992) and project-based learning (Blumenfeld et al., 1991).

For the several past years, we have been teaching course at the Comenius University in Bratislava that is engaged in educational robotics. This course is taught both in winter and summer semester and it aims to provide students experience with educational robotic kits (Kabátová, 2010). We try to lead the lessons in a way that allow students to get knowledge that cannot be obtained only by lecturing. They have the opportunity to get acquainted with the advantages and disadvantages of robotic kits in the context of teaching. This year, we had opportunity to apply the elements of authentic learning (Pasch, 1991) in the natural way. During the winter semester of the academic year 2017/2018, in the course of Robotic kits in Education 2 (hereinafter RKiE2) we had a student (future teacher, we will call her Jane) whom started to run a robotic workshop at a leisure centre. This workshop had different Robotics kits than we have at our faculty. This was one of the reasons, why we decided to prepare opportunity for

students to try to work on an authentic problem, which was the creation of assignments with different robotic kits for real and diverse pupils at robotic workshop. So, we created activities that would be as useful as possible for our students and prepared them for situations in which beginning teachers can find them self.

Constructionism in robotics

In the creation of the syllabuses of our course, we were inspired by Papert et al. (1991), he states that simplest definition of constructivism evokes the idea of learning-by-making therefore, the whole content of the course is formed in a way that students acquire knowledge through active work. Therefore, students at our course are not only familiarizing themselves with robotic kits but they are actively exploring them and creating activities with them. This semester students also received feedback from their classmate, whom taught activities created during the course lessons. Students then adapted new activities, based on received feedback and new requirements. Papert et al. (1991) further states that: *"Constructionism shares constructivism's view of learning as "building knowledge structures" through progressive internalization of actions. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in construction a public entity, whether it's a sand castle on the beach or a theory of the universe"*. In our case, it was about creating learning content for a leisure time robotic workshop. The main principles of constructivism include rich user-oriented interaction, the use of authentic problem situations, learning collaboration and learning experience in the process of constructing knowledge (Papert et al., 1991). Based on these principles, we have also changed the role of the student which verified the activity. She has become a helper for pupils in the process of actively construction their own concepts, structures of knowledge and skills. Thus, we have implemented constructionism as a learning theory and educational strategy, which also takes into account the importance of the enthusiasm for learning, as we confirmed by the results of our research. This theory states that the pupils will be more actively involved in the learning process by working on what is of individual significance (Kafai et al., 1996). In our case, it was mainly the student whom tested activities that was created during that week. Similarly, Papert (1993) argues that the construction of knowledge is more effective when pupils are involved in designing meaningful projects and creating artefacts.

Papert et al. (1991) further states that digital technologies are perfect means of actively working on something real, when we construct and learn in a natural way (Kabátová, 2010). Work on something real takes place in authentic learning.

Authentic learning in robotics

In authentic learning, learning is organized with emphasis on the meaningful use of the learning material, which we have set as our primary goal in teaching our course. This way of learning reflects the process through which most of the knowledge and skills have been acquired through human history. It allows pupils to build on learning context and to use acquired knowledge in a meaningful way (Pasch, 1991). The basic characteristics of an authentic problem based on the Three-Step Model of Extension (Pasch, 1991) are:

- The real problem **should be based on the interests of pupil** or group of pupils, because it has personal value and is interesting for them. We have already seen this at the beginning of the semester, when we agreed with the students on the course details.
- The real problem has **no predetermined correct answer**. However, while solving real problems, pupils should make the most by using authentic methods – they should approach the problem as well as experts. Each aspect of the project provides an opportunity to lead pupils to high-level work.
- While solving the real problem, the pupils will in the end present gathered information to **the real audience**. This audience depends to a great extent on the pupils age and the complexity of the problem and should have and strong in interest in the problem. In our case, the audience was composed of real pupils from leisure time workshop.

Authentic teaching activities can be the main goal, during preparation of the course lessons. The ultimate goal in any course should be that pupils make meaningful use of acquired knowledge (Pasch, 1991).

University robotic course within teacher training

RKiE2 is an elective course of master's degree program in Teacher's training during the winter semester. It is intended for all combinations of student of teacher's training that we have at our faculty, i.e.: mathematics, physics, chemistry, biology, physical education, geography. It continues the course from bachelor degree program entitled Robotic kits in Education 1. The content of this course was developed by our colleague a few years ago (Kabátová, 2010). We still use some of her findings, but we adjust content of the course according to the number of students and their needs. The aim of our course was to prepare opportunity for students to try to work on an authentic problem, which was the creation of activities with different robotic kits for real and diverse pupils at robotic workshop. Our students were exploring the elements that would be included in the curriculum for leisure robotic workshop. Experience with creating this kind of curriculum should be beneficial, because in Slovakia educational robotics is most often appearing in leisure workshops.

Our school own robotic kits LEGO WeDo 1.0, WeDo 2.0 and LEGO NXT. From her own initiative, Jane brought robotic kits used at the workshop from leisure centre. Course RKiE2 at our faculty lasted 13 weeks (one lesson per week = 90 minutes) and we were dealing with following topics:

- **Simulation of an inexperienced teacher:** The students simulated a teacher who does not have any prior experience with robotics and is about to begin teaching it. They should search for suitable materials, tutorials and blogs on the internet to inspire them or direct them how to teach with robotic kit LEGO NXT / Ev3.
- **Getting to know the LEGO NXT and Sphero:** Students have tried to work with the LEGO NXT and the Sphero robot. At the end of the lesson they evaluated the advantages, difficulties and predict how pupils could react to these robotic kits.
- **Getting to know BeeBot, Albi Robot and Code-a-Pillar:** At the beginning of the lesson students get acquainted with three simple robotic kits. Subsequently, they should design a series of activities with these kits to be used at robotic workshop in leisure centre.
- **Getting to know LEGO WeDo 1.0 and WeDo 2.0:** The students experienced work with both robotic kits by getting to know the programming environment and building a few simple models according to the instructions.
- **Getting to know Ozobot:** The students were getting acquainted with the possibilities of programming the Ozobot robot and creating simple activities for this robot.
- **Creation of activities for robotic kits LEGO WeDo 1.0 and WeDo 2.0:** The students choose one tutorial for each robotic kit and created a story that incorporated both. For the first time, they were given instructions that the created activities for workshop should include story, be focused on design and program, develop interdisciplinarity and wrote down the learning objectives of the activity.
- **Web searching and testing specific activity for the robotic kit LEGO Mindstorms NXT:** Modification of activity for workshop needs.
- **Characterizing and defining recommendations and experiences from whole process of creation of activities**

Constructionist design of the course

During the design of the course we applied eight Papert's big ideas (Papert, 1999):

- **Learning by doing** - we have implemented by the exploration of different robots, robotic kits and preparation of assignments, that were used for real pupils at robotic workshop in leisure centre.

- **Using technology as a building material** - students worked with robots but also created assignments for pupils using a text editor. At the beginning of the course assignments contained mostly text, but later students also included e.g. images, various web links and samples of programs for robot control
- **Hard fun** - robots are basically toys but solving complex tasks with them could be difficult (XXX, 2010). Likewise, creating interesting and appropriate assignments that follow educational goals for robotic workshop at leisure centre is demanding.
- **Learning to learn** - students learned what assignments or types of tasks with different robots are appropriate for specific groups of pupils.
- **Taking time – proper time for the job** – syllabus of the course was flexible, so students had enough space to explore robots and to create assignments. If needed, some of the assignments were finished at home or took two lessons.
- **The opportunity to make mistakes and learn from them** - students had enough space to create their own solutions where they committed many mistakes. We did not point them out immediately but we tried to engage in dialogue with students to find out where the problem was and with the help of questions direct them to the solution.
- **Teachers also learn** - teachers cannot be fully prepared for any problems that may arise, but they can take them as opportunity for further learning and so they learn with students. In our case, we did not know most of the new robotic kits.
- **The use of digital technologies for further learning** - robot exploration and creating assignments for pupils created opportunity for students to discover important information technology concepts.

Methodology

The aim of this article is to provide an overview and results of a case study on integrating authentic learning into university course about educational robotics. Not only from point of view of usage robotic kits during course lessons but also from the point of view of creating specific activities for leisure time workshop, that take place each week. In the lessons we worked with robots and robotic kits BeeBot, Albi Robot, Code-a-Pillar, Ozobot Evo, Sphero, LEGO WeDo 1.0, LEGO WeDo 2.0 a LEGO Mindstorms NXT. The students' work process was reported in the moodle system, where they at the end of each lesson submitted the assignments – the design of activities they created. This was the data that we analysed by **qualitative methods** (Creswell, 2002). In addition, students shared all activities through google documents. We also recorded some audio from course lessons and final lesson was recorded on the video. The student who taught these activities at the workshop informed us about results and feedback at the next lesson. Further she provided her field notes with the photographs for each workshop lesson.

Participants

Our course was attended by five college students, one male and four females. Three students were enrolled in the first year of master's degree program in Teacher's Training and other two students did not attend any didactically oriented subjects or pedagogical-psychological subjects. They were students of master's degree program in Applied Informatics and another in Mathematics of Economy, Finance and Modelling. The activities created by the students of the course were verified in the workshop at leisure time centre which was taught by one of our students Jane. The number of pupils in the workshop at leisure time centre has changed over the weeks. There were two groups of pupils. The pupils were aged between 8 and 14 (mostly boys) and in one group there were from 2 to 10 pupils.

Activities created by students

In this chapter, we provide preview of the activities that students have created during the course. In the figures below, we can see how length and form of the proposed activities have changed during the

semester. After getting acquainted with robotic kits, students tried to create activities to be implemented by two groups of pupils in a row and to be interesting. They tried to take into account the time constraints and area in which the robotic workshop was running. They created a series of four activities that took place during five lessons within workshop. The aim of activities created by students was to develop pupil's interest in the workshop theme and to explore pupil's potential. The educational goals of activities were defined by our students. During our course, students discovered why well defined educational goals for teachers are needed. They did not define goals based on National educational curriculum, because it did not exist for leisure workshops. Students focused mostly on increasing of motivation of pupils. We did not edit the activities to see change in form, amount of text used, grammatical or formatting errors. To each activity we added a comment from the student who tested it in the workshop at leisure centre.

BeeBot, Albi Robot a Code-a-Pillar

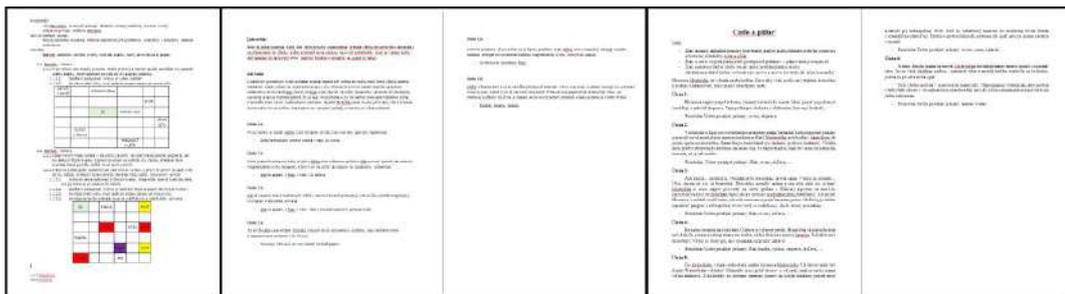


Figure 1. Activities with BeeBot, Albi Robot and Code-a-Pillar

Comment to activity Bee-Bot (left part of Figure 1): *"In this assignment, we had to improvise on the lesson because of the lack of props. Some pupils had experience with this robot from the nursery. In general, Bee-Bot was least attractive for pupils."*

Comment to activity Albi robot (middle part of Figure 1): *"Pupils were most familiar with this robot because many have it at home or their friends have it. Pupils also highlighted an error in the assignment and stated that assignment could be shorter."*

Comment to activity Code-a-Pillar (right part of Figure 1): *"Activity was longer than expected. Assignment could be shorter. The pupils very accurately measured the route and tested it several times, so it was time consuming."*

Ozobot

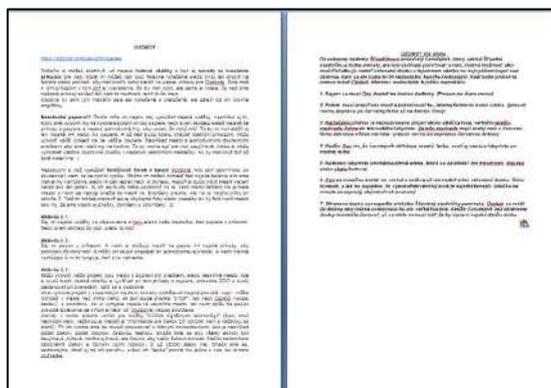


Figure 2. Activities with Ozobot

Comment to activity Ozobot painting (Figure 2): "The pupils were very fond of drawing a trail for Ozobot. Difficulties in this activity were caused by having only one robot available for all pupils. But the pupils were very creative and drawn a lot of trails for the robot."

Comment to activity Ozobot programming with the app: "The assignment was inaccurate, I had troubles to see the aim of the students that created it. So, the pupils improvised but they really liked results of how the robot lighted up."

LEGO WeDo 1.0 and LEGO WeDo 2.0

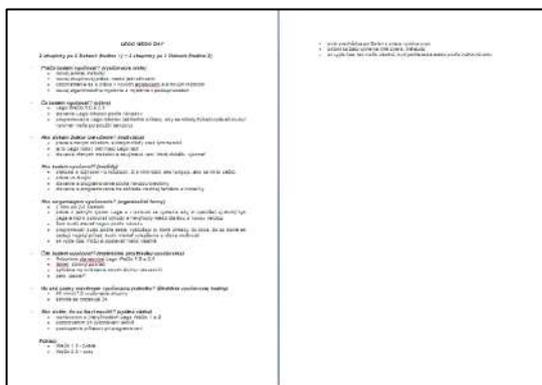


Figure 3. Activities with LEGO WeDo 1.0 and LEGO WeDo 2.0

Comment to activity Lego WeDo (Figure 3): "This was very interesting for kids, only a pity that there was only one computer. It would be better if we could have had more robotic kits, because there were a lot of pupils in one group."

LEGO Mindstorms NXT

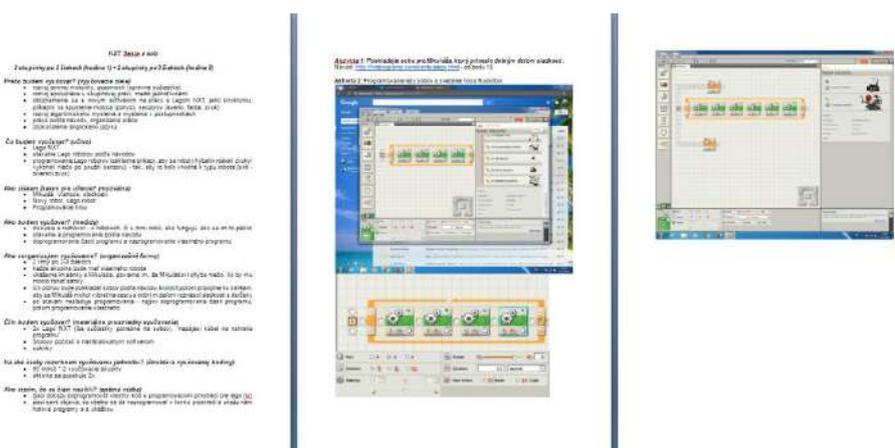


Figure 4. Activities with LEGO Mindstorms NXT

Comment to activity LEGO Mindstorms NXT (Figure 4): "The construction was not difficult, but programming was unrealistic for pupils this young. But they enjoyed moving of robot."

Experiences from authentic learning

Jane was most engaged in authentic learning, so she summed up conclusions from the course. She comments content, formal implementation, positive and negative aspects of the whole course. Most positive comments are directly related to authentic learning e. i.: "I did not have to invent my own activities.", "I really enjoyed the joy of the pupils. They relished the workshop mainly due to the activities

that we have created." In a descriptive manner, she stated that methodical materials need to be prepared alongside created activities. Emphasis should be placed not only on correct content but also on motivation aspect: *"Attention should be focused on process of activity, not only on activity goals."* The differences between students of Teacher' training and other students were summarized as follows: *"The group made up of the future teachers cared too much about the pupils and their age, so they created too simple activities – pupil could manage even more challenging tasks. Other students invented awesome activities but lacked greater focus on pupils' abilities."*

In addition, she stated, that from **pupils' point of view**, they were the most fascinated by the results of what they were doing – the movement of the robot and its interaction. The pupils were incredibly happy when Ozobot began to shine as traffic light, when the Bee-Bot managed to get through the challenging route, when Code-a-Pillar simultaneously sang and move to the marked spot. They were also very happy when they could choose what to do with the LEGO building blocks and already knew how to control robot, so on the basis of their new knowledge, they could create something completely alone.

From the **teacher's point of view**, it was demanding that the assignments were created mainly for the pupils. There were no instructions for the teacher, no expected results, no pre-programmed codes that were expected to be done by pupils. Furthermore, there were quite a lot of pupils on each lesson, so they worked in pairs but each pair had their own assignment. This created a lot of work, because teacher needed to help each group on different tasks.

Conclusion and Discussion

In this article we presented activities, which our students created for robotic workshop at leisure time centre. The content and the form of activities were developed based on observations of our student, who taught these activities. Although we have been teaching this course for several years, we have experienced for the first time that students actively used their acquired knowledge directly in practice - in real robotic workshop with diverse pupils. Over the past few years, only few students used the acquired knowledge from this course in practice as participation in the robotic competition Istrobot. Based on the data analysis, we found out that during one semester students are able to get deeper knowledge with a certain number of robotic kits - we used eight robotic kits and it was too many for our students. Based on the observations from the workshop, we found out that number of pupils in the workshop was growing especially because of attractive nature of created activities. We hope that such increased internal motivation could help pupils to acquire a positive relationship to the compulsory school subject Informatics and to direct them in their career growth. Based on the experience from this semester, we managed to buy other types of robotic kits within our department. In the current summer semester of the academic year 2017/2018, our students have already been able to use some of these new robotic kits with pupils during another leisure time robotic workshop.

References

- Bers, M. U. (2007) Blocks to robots: Learning with technology in the early childhood classroom.
- Blumenfeld, P. C. et al. (1991) Motivating project-based learning: Sustaining the doing, supporting the learning. Educational psychologist, Vol. 26, No 3-4, p. 369–398.
- Creswell, J. W. (2002) Educational research: Planning, conducting, and evaluating quantitative. Upper Saddle River, New Jersey.
- Kabátová, M. (2010) Constructionist approach at teaching pre-service teachers educational robotics. Ph.D. thesis, Comenius University, Bratislava.
- Kabátová, M. and Pekárová, J. (2010) Learning how to teach robotics. In Proceedings: Constructionism 2010 conference.
- Kafai, Y. and Resnick, M. (1996) Constructionism in practice. Mahwah. Lawrence Erlbaum Associates, New Jersey. ISBN 0-8058-1985-1.

Kim, Ch. M., et al. (2015) Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers & Education* 91, p. 14–31.

Papert, S. (1999) The Eight Big Ideas of the Constructionist Learning Laboratory. Unpublished internal document. South Portland: Maine (1999) Cited in STAGER, G. Papertian Constructionism and the Design of Productive Contexts for Learning. In *Proceedings: EuroLogo X 2005*. <http://www.stager.org/articles/eurologo2005.pdf> [Accessed: 2018-05-02]

Papert, S. (1993) *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). Basic Books, New York.

Papert, S. E. and Harel, I. E. (1991) *Constructionism*. Ablex Publishing.

Pasch, M. (1991) *Teaching as Decision Making: Instructional Practices for the Successful Teacher*. Addison Wesley Publishing Company.

Spolaôr, N. and Benitti, F. B. V. (2017) Robotics applications grounded in learning theories on tertiary education: A systematic review. *Computers & Education*, Vol. 112, p. 97–107.

Strommen, E. F. and Lincoln, B. (1992) Constructivism, technology, and the future of classroom learning. *Education and Urban Society*, Vol. 24, p. 466–476.

Constructionism as an Epistemological Option in Courses of Youth Center for Science and Culture – Bahia – Brazil

Elmara Pereira de Souza, elmarasouza@gmail.com

Youth Center for Science and Culture – Vitória da Conquista – Bahia, Brazil

Luísa Souza Moura, luisasouzamoura@gmail.com

University of São Paulo (USP) – São Paulo, Brazil

Abstract

The goal of this paper is to report on the practice of the Youth Center for Science and Culture, Vitória da Conquista, Bahia, Brazil, a public school for secondary school students. The Youth Center uses a constructionist perspective as an epistemological basis in their courses. The students are the main characters in the learning process and they use the coding languages Logo and Scratch in their projects' development.

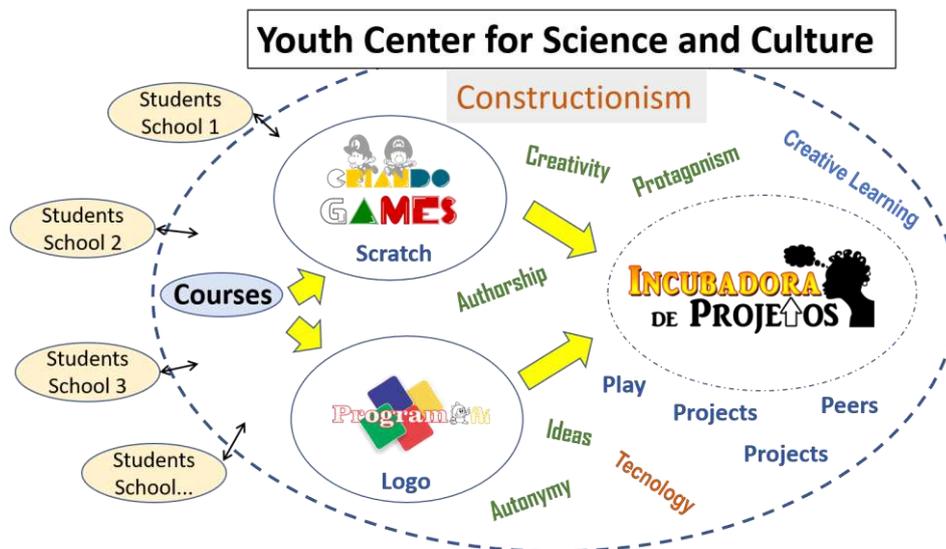


Figure 1. Pedagogical structure of the courses of Youth Center for Science and Culture – Vitória da Conquista – Bahia - Brazil

This experience shows that it is possible to create spaces of authorship so that the public school's students don't turn out to be just users of the technologies, but authors of socially relevant content, that can change their lives and that can provide future perspectives.

Keywords

constructionism; authorship; protagonism of young people

Introduction

We undergo, in Brazil, an educational crisis that has worsened with public political impositions and methodological proposals that are unlinked to the students' realities; pedagogies that reinforce the status quo and that, most of the time, are insufficient with regard to the diversity, to the multiplicity, to the singularity and don't favor the autonomy, the authorship of our students. In the Brazilian educational system, the problems are innumerable.

These problems can be seen by analyzing the indexes of the students' evaluations, the indexes related to the teachers' training; also our own teaching practice in the basic education and, specially, in the secondary school.

The Program for International Student Assessment (PISA, 2015), for example, was applied in 70 countries, and Brazil got the 63rd position in sciences, 59th in reading and 66th in math. Analyzing the Basic Education Development Index (BEDI, 2015), it is verified that, in a scale from 0 to 10, Brazil got a 3.7 average.

When we analyze the data related to the teachers' training, we verify that, from the state public secondary school teachers, 56.2% don't have postgraduate studies and 74.9% don't participate in continuous training (Universidade Federal do Rio Grande do Sul, 2018).

We understand that there are a lot of variables to analyze Brazilian education, only the indexes are not enough to present the real situation of our education, but they can be used to cause a search for innovational educational proposals, that are close to the interest and to the reality of the students and that can deal with the youths' desire for knowledge. The effective usage of information and communication technologies (ICT), of the cyberspace's resources in the pedagogical process and of the insertion of creative learning in the classrooms can be made possible, since the young have a close relation with technology.

The usage of the ICT, of the mobility, of the ubiquitous resources in education can favor the agency of the student's desire to build new knowledge. With pedagogical practices based on constructionism (Papert, 1994 and 1986), on dialogism (Bakhtin, 2000), on affection-joy (Deleuze, 2008; Spinoza, 2009) subjective singularities can be produced in education favoring the training of our young people.

We present, in this paper, the experience of the Youth Center for Science and Culture, a public school of the State of Bahia - Brazil, specially, the experience of the "ProgramAI" and "Criando Games" courses, developed for secondary school students, in addition to the Project Incubator, having constructionism as an epistemological option.

Youth Centers for Science and Culture

The Youth Centers for Science and Culture are public schools created by the Education Department of the State of Bahia and instituted through the decree nº 12.829, of May 4th, 2011. They have the goal of offering complementary education and diversifying the public school curriculum, as well as promoting the students' access to contemporary thematics, through studies and interdisciplinary activities. The main pillars of the pedagogic activities developed in the Youth Centers are: (1) the student is the author of his own journey; (2) the school's connection; (3) the knowledge is transmedia; (4) learning is fun.

The Youth Centers are interschool spaces and bring as a feature the logic "one-lots", which means, they are after-school programs offering courses, workshops and activities to students of any state public school of the region covered by the Center. This feature multiplies the attendance and, at the same time, promotes the interaction of students from different schools, ages and degrees, as in every workshop, there may be students from elementary school to secondary school and from multiple schools.

The Centers are a pedagogical innovation laboratory, they are active learning spaces, and they enhance the usage of digital technologies in the creative learning perspective (Resnick, 2017). The proposal is to provoke the students' curiosity and encourage them to create a new relation with the act of learning, motivated by the pleasure of discovery. No Youth Centers activity is required. The students choose if, when and how they'll participate, making possible the effective protagonism of their training.

There are five Youth Centers for Science and Culture in Bahia in the following cities: Salvador, Senhor do Bonfim, Itabuna, Barreiras and Vitória da Conquista.

In this paper we present the experience of the Youth Center from Vitória da Conquista, specially the experience of the "ProgramAI" and "Criando Games" courses and the Project Incubator.

The courses “ProgramAI”, “Criando Games” and the Project incubator

The “ProgramAI” is a course of introduction to programming logic and it was created to address the need of implementing courses that are close to the students’ interest. It has the objective of providing to the secondary school students the initial contact with robotics systems control and coding, and also stimulate the development of logical thinking, creativity and teamwork for problem solving.

The course “Criando Games” aims to provide the students with the knowledge of Scratch programming language to the development of games and animations. The activities are created in a practical way so that the students can be authors of their projects and the gaming and animation development can be a fun and attractive way of learning. The students are part of the Scratch community, sharing their products and interacting with other youth.

The first experiences happened in 2016 with two classes of 20 students from 1st to 3rd years of secondary school. The workload of the courses was 30 hours each, with two weekly face-to-face meetings and distance activities in virtual learning environments (cjccvc.org⁶⁰).

After one year of performance in the courses “ProgramAI” and “Criando Games”, we identified the potential of many students in the digital technologies fields and we noticed that they had innovative ideas, but they couldn’t keep and/or finish their projects because the courses time was limited. After that observation, we created the Project Incubator, a space of authorship so the students can deepen their projects or develop research projects and innovative products after their own ideas. There is one weekly meeting and what determines the time that each student will be in the Incubator is the project he/she is developing. Any Youth Center student can integrate the Incubator at any time of the year; he/she only needs to develop a project in the digital technologies field, join a project that is being developed or just be welcomed in the group and learn with the others. The objective is to be an open and productive space in which the students can exercise creativity and produce knowledges through research projects and products development.

In all these courses and in the Project Incubator we follow the constructionism as epistemological basis.

Constructionist as epistemological option

The Youth Center is a public school opened to pedagogical innovation, so we chose constructionism as an epistemological basis in the courses “ProgramAI” and “Criando Games”, and also in the Project incubator.

In his investigations, Papert (1994) looked for different ways to learn, in which kids could be producers of knowledge, leaving their spot as just users to become active in the knowledge construction process. As an author, the students should take control of their own development along with the school as a learning place. So, inspired in Jean Piaget’s psychogenic theory, he created a set of ideas called constructionism.

According to Papert (1986 and 1994), with the use of computers, the student visualizes and verifies his/her mental constructions related to the concrete as well as the abstract, following an interactive process that favors knowledge production.

The creation of active learning environments is one of the constructionism principles and, in Center’s experiences, is one of the most important actions for the students to be able to develop creativity and produce knowledge. The active environments allow the students to test their ideas, theories or hypotheses and try the creation and implementation of projects. Papert (1993) realized that with the computers, there was the possibility of creating conditions for significant changes in the developing process of the children and, along with a group of researchers from MIT, developed the Logo programming language. Logo is a language that is considered simple, allowing the development of projects, and at the same time as it has the power of a professional programming language.

⁶⁰ This is the address of virtual learning environment of Youth Center.

Recently, the Lifelong Kindergarten Group of the MIT Media Lab, led by Mitchel Resnick, inspired in Logo, created Scratch, a free programming language and an online community in which you can create your own interactive stories, games and animations. Scratch has an easier and more friendly graphical interface that is free and it's available on and offline. By using Logo and Scratch, the children and teenagers have the opportunity to be in control of the knowledge production and the development of their own ideas through the exercise of creative learning.

Creative learning experience is based on four P's: Projects, Peers, Passion, and Play. Resnick (2017) describes each P:

- Projects. People learn best when they are actively working on meaningful projects – generating new ideas, designing prototypes, refining iteratively.
- Peers. Learning flourishes as a social activity, with people sharing ideas, collaborating on projects, and building on one another's work.
- Passion. When people work on projects they care about, they work longer and harder, persist in the face of challenges, and learn more in the process.
- Play. Learning involves playful experimentation – trying new things, tinkering with materials, testing boundaries, taking risks, iterating again and again. (Resnick, 2017).

In Youth Center, Logo is used in the course "ProgramAI" and Scratch in the course "Criando Games". Both languages promote the development of creative thinking.

Some projects developed in Project Incubator

In the Youth Center's Project Incubator, many projects were produced using Scratch, among them we can highlight the game "Choices", the interactive animation "Are your behaviors ecologically sustainable?" and the game "Aedes Adventure". All projects have the objective of being educational actions and social relevant proposals.

In Brazil, 8.4% of the teenagers from 12 to 17 years old are obese and 25.5% are above the ideal weight. Identifying these indexes and knowing that the teenagers like using digital games, the project "Choices" proposed the creation of a fun and interesting game, that could be an educational channel to make the youth aware of the importance of a healthy diet, physical activity and the consequence of the choices (individual and collective) to life quality. (figure 2).



Figure 2. The first screen of the game "Choices". Player can measure the body mass index - BMI (2017).

In the project "Are your behaviors ecologically sustainable?" the students developed an interactive animation with Scratch that was used as data production methodology (variables were created to keep

the users' answers) to know if the students from a public school had sustainable attitudes related to water, energy, consumption, waste disposal and transportation usage.

This project was also an educational action, as the subjects of the research, by having fun with the animation and answering the proposed questions, gave feedback about each theme related to sustainability and could reflect about their own attitudes (figure 3).



Figure 3. A girl using the animation “Are your behaviors ecologically sustainable?” developed in Scratch (2017)

The game “Aedes Adventure” was created so people could think about the importance of combating the bug *Aedes Aegypti*'s breeder. In this game, we used augmented reality and, with a computer's camera, the players could use their bodies to “kick” the breeders and the bugs.

Besides providing the students' authorship, we had exciting results with this project that have been changing these boys' and girls' lives, as well as their families'. (1) The game “Choices” has won 3rd place in the Edital Technologies for Education from FAPESB and the students got a R\$ 5,000,00 prize; (2) we wrote an article about this experience in the development of the game “Choices” that was presented in the Meeting of the Brazilian Society for the Progress of Science (boys who had never left their hometown had the opportunity of being in one of the greatest events on the science field from Latin America, of presenting a paper among ungraduate and postgraduation students); (3) we received honored mention in this event, for the merit of the work; (4) we received a motion of Applause at the Vitória de Conquista City Council because of the developed work; (5) the project of the interactive animation “Are your behaviors ecologically sustainable?” was finalist of the Brazilian Science and Engineering Fair 2018; (6) the students presented their Scratch developed projects in the Campus Party 2017 and in the Meeting of Students that took place in Salvador - Bahia.

In the Project Incubator, there are no proofs or traditional learning measurements. The evaluation is effectively developed in the process of knowledge construction. We evaluate every stage of the project, the involvement of each student; we reflect about all the steps taken and we identify which path to follow, which subjects to deepen and to study. The partnerships with teachers and Universities are fundamental to the Incubator's projects development.

In the Incubator, everything is built in partnerships, after the students' ideas, who are protagonists in this project. Each one learns and contributes according to his rhythm and knowledge. In the end, we all learn about contents and about coding, about how to relate with the other, about group work, about overcoming our own limits and going beyond the imagined, about trusting our capacity. The Project Incubator brings secondary school students in contact with University education, providing them with future perspectives.

The projects of the Project Incubator are available in the repository of learning objects cjccvc.org.

Conclusion

Considering that the contemporary subjectivity is anchored in capitalistic devices that try to standardize, form consumer subjects and social, economic and cultural products users, developing educational

proposals that allow the students' authorship, the youth's protagonism is relevant, especially in an unequal country such as Brazil.

Constructionism as epistemological option in the courses and activities in Youth Center for Science and Culture from Vitória da Conquista have shown themselves very productive, since, in the courses "ProgramAI" and "Criando Games" as well as in the Project Incubator, the students express their feelings and ideas through the project development; they, are authors, share their productions, work in groups and learn with interaction and technological artifacts.

The experience of projects, games and digital animation production shows that it is possible to create authorship spaces so that public school youth don't turn out to be reproducers and users of what's already made, but producers and authors of socially relevant educational content.

References

Bakhtin, M. (2000). *Estética da Criação Verbal*. 4. ed. São Paulo: Martins Fontes.

Deleuze, G. *Em médio de Spinoza*. Buenos Aires: Cactus, 2008.

Papert, S. (1993). *The Children's Machine: Rethinking School in the Age of the Computer*. New York: Basic Books.

Papert, S. (1986) *LOGO: Computadores e Educação*. São Paulo: Brasiliense.

Resnick, M. (2017). *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. MIT Media Lab. MIT Press.

Spinoza, B. (1996) *Ethics*. E. Curley (trans.). London: Penguin.

Universidade Federal do Rio Grande do Sul (2018). Carvalho, M. J. S, Neves, B. G. B, Melo, R. S. *Cultiveduca*. Brasil no. BR512014001340-5, 18 mai. 2014, 25 jan. 2016. Source: <http://cultiveduca.ufrgs.br/pg.index.html>.

Female Teenagers and Coding: Create Gender Sensitive and Creative Learning Environments

Bernadette Spieler, bernadette.spieler@ist.tugraz.at

Institute of Software Technology, Graz University of Technology, Austria

Wolfgang Slany, wolfgang.slany@tugraz.at

Institute of Software Technology, Graz University of Technology, Austria

Abstract

The number of women in technical fields is far below the average number of males, especially in developed countries. Gender differences in STEM are already present in secondary schools in students aged between 12 to 15 years. It is during this intermediate female adolescence that girls begin to make critical career choices, which therefore makes this a key age to reinforce them and reduce the gender disparities in ICT. Acquiring computational thinking (CT) skills, particularly coding, is important for building a positive economic, developmental, and innovative future. To address the gender bias in schools, one of the goals of the European H2020 project No One Left Behind (NOLB) included integrating Pocket Code, a free open source app developed by the non-profit project Catrobat, into different school lessons. Through game design, Pocket Code allows teenage girls to incorporate diversity and inclusiveness, as well as the ability to reflect their cultural identity, their likes, and their ways of interacting and thinking. To evaluate the impact of the use of the app in these courses, we captured the results on engaging girls in design and coding activities. For this paper, the authors present the data of surveys via a qualitative content analysis during the second cycle of the project. The results let the researchers conclude that the organization and the setting of the coding courses (for example, guidance and supporting material, freedom of choice) had much more influence on female students' engagement than the coding aspects or the app itself. In contrast, male students more frequently mentioned missing features in the app, and stated that they liked the coding. With a focus on female teenagers, the results allow us to conclude that a suitable classroom setting is significantly more important for them than the coding tool itself.



Figure 1. Providing inclusive coding environments for female teenagers

Keywords

pocket code; game design; gender inclusion; coding; mobile learning; STEM; social inclusion; constructionism; girls; learning environment

Introduction

Secondary school is the place where students make the critical choices which decide their future careers, develop a more realistic picture of their future jobs, and assess their career-relevant abilities. Researchers observe that girls' interest in IT drops significantly from the age of 12 to 15 (Tsan, Boyer, and Lynch, 2016). To address this gender bias at an early stage, a goal of the European project No One Left Behind (NOLB) was to integrate Pocket Code, an app developed at Graz University of Technology, into different school subjects, thus making coding more accessible and attractive to female students. Although promoting gender equality is a longstanding policy which all European countries place on their agendas (EC, 2014), a gender-based inequity in ICT still poses barriers for women. Thus, the acquisition of digital skills is more important than ever and represents a key professional qualification (Balanskat and Engelhardt, 2015, Kahn, 2017, Tedre and Denning, 2016). For the NOLB project, the authors assumed that according to related literature it is possible to spark girls' interests by getting them engaged in computational thinking through collaborative, creative, and engaging coding activities (Khan and Luxton-Reilly, 2016). Thus, the team studied whether the design of mobile games has an impact on inclusion and satisfaction of female students. Thus, the following research question has been defined: How can we organize coding activities to reinforce female teenagers in computer science?

The paper is organized as follows: First, we describe the challenges of teaching CS in Europe and discuss several learning theories, with a focus on constructionism. Second, we present solutions on how to provide suitable learning environments for girls, followed by a brief description of the NOLB project and our educational apps for coding. Third, we present the results of the quantitative and qualitative surveys which show insights for framing suitable classroom settings for coding activities for girls. Finally, the last sections conclude the paper and describe our future work.

Learn Coding: A Worldwide Challenge

The European Commission states, "*All of Europe's citizens need to be educated in both digital literacy and informatics*" (Informatics Europe/ACM Europe, 2013, EC, 2016). Thus, IT education is seen as an interdisciplinary field that bridges the gap between the use of digital media and information-processing technology as well as basic concepts and fundamental ideas of computer science. The EC recommend that students should benefit from computer literacy at an early stage and that it is important to find a standardized definition of the informatics curricula through all countries. However, in Austria and in many other European countries, computer science topics are underrepresented in school curricula, hence, teaching time for these topics is limited (Informatics Europe/ACM Europe, 2016). From primary through secondary school, only a few opportunities exist for students to explore coding and CS topics.

Learning theories from the past serve as an organized set of principles, clarifying how people acquire, retain, and recall knowledge (Schunk, 2016). These theories helped researchers to gain a better understanding of how learning occurs and help us to select appropriate techniques, tools, and strategies to support learning and teaching how to code. Three basic types of learning theory exist: Behaviorism, Cognitivism, and Constructivism, and some subtypes or variations, e.g., Instructionism, and Constructionism.

Figure 2 provides an overview of important findings of each theory, how each theory handles students' motivations, and shows details on how teaching and learning should be done.

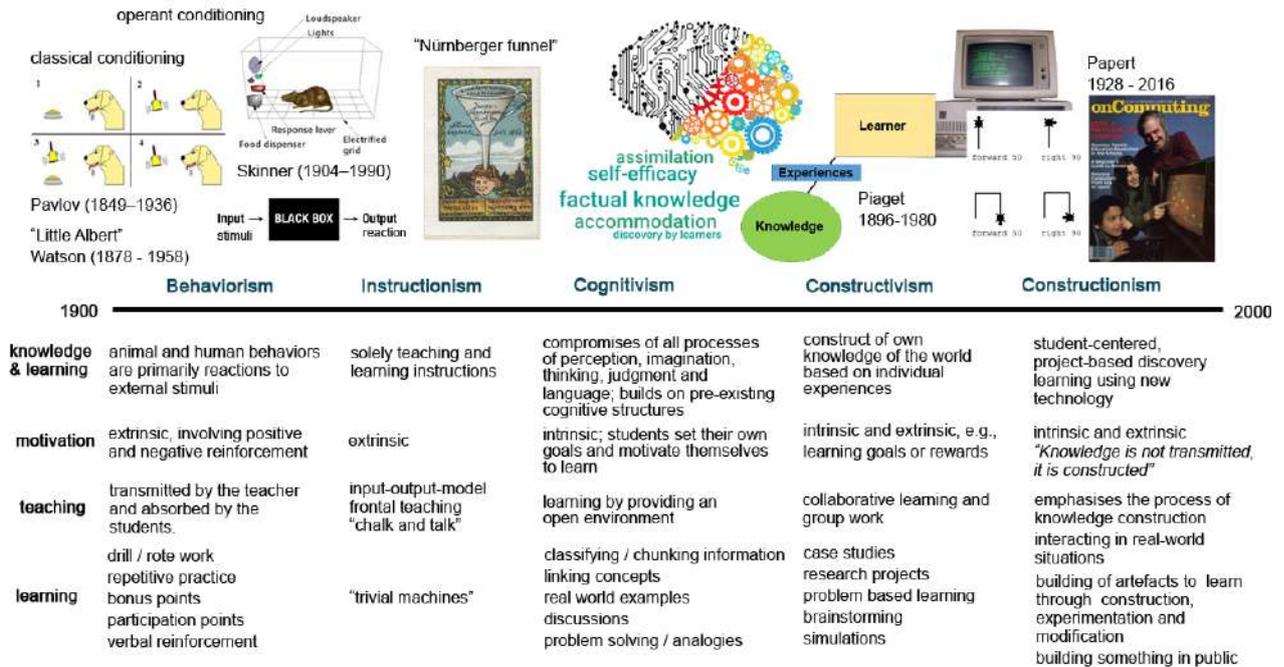


Figure 2. Learning theories of the 20th century: Behaviorism/Instructionism (Pavlov, 1927, Skinner, 1976, von Foerster et al., 2009), Cognitivism (Piaget, 1968, Perry, 1999), Constructivism (Piaget, 1968, Vygotsky, 1978), Constructionism (Papert, 1980, Paper, 1991).

Constructionism

The Constructionist approach (Papert, 1980) is interested in building knowledge through active engagement and personal experience. Papert noted that individual learning occurred more effectively when students understood the world around them and were creating something that was meaningful to them. This experiential and discovery learning by challenges should inspire creativity, and project work allows for independent thinking and new ways of constructing information. The iterative process of self-directed learning underlines that humans learn most effectively when they are actively involved in the learning process and build their own structures of knowledge. In this theory, communication between students about the work, and the process of learning with peers, teachers, and collaborators, is seen an indispensable part of a students' learning (Papert, 1993, Papert, 1991).

"The construction of knowledge through experience and the creation of personally relevant products. The theory proposes that whatever the product, e.g. a birdhouse, computer program, or robot, the design and implementation of products are meaningful to those creating and that learning becomes active and self-directed through the construction of artefacts." [Papert, 1980, p.2]

Thus, Papert described the huge potential of bringing new technology into the classroom (Papert, 1993). For this reason, he co-invented the LOGO programming language in the late 1960s at the MIT. LOGO was designed to have a "low threshold and no ceiling" and was indeed used to help novice programmers, and to support complex explorations and the creation of sophisticated projects (Tinker and Papert, 1989). LOGO set the basis for later visual programming tools, such as Etoys (Kay et al., 1997) or Scratch (Resnick et al., 2009). Such block based visually oriented tools made programming accessible for a large number of people and taught new skills such as engineering, design, and coding (Blikstein and Krannich, 2013). They allow students to recognize blocks instead of recalling syntax. They are broadly integrated in schools, or even at universities all over the world (Meerbaum-Salant et al., 2010).

To conclude, psychologists and pedagogues from today following the constructionist approach state three main goals. First, they wish to rethink traditional education without step-by-step guidance and to create new social and open environments. Second, they strive to allow students to engage in meaningful

and relevant problem-solving activities, and third, they want to integrate new tools, media, and technologies in school lessons (Neo and Neo, 2009).

Jeannette Wing, 2006 shaped the term “Computational Thinking” (CT):

“Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (Wing, 2006)

Wing’s idea that children who are introduced to CS learn more than just programming opened a new way of thinking, e.g., it showed the benefits of learning to think like a technician (Wing, 2008). Wing’s findings have been incorporated into the CS curriculum of many countries (Kahn, 2017) and into K-12 movements (Mannila et al., 2014). However, CT is just a small subset of Seymour Papert’s ideas in the 80s (see previous section). Papert was the first who used the phrase computational thinking and defined it in a much broader way. For instance, Wing focuses mainly on computer programs, whereas Papert stated that there are more kinds of constructionist projects, and computational ideas could serve learning in a broad variety of subjects, this *“can change the way [children] learn everything else”* (Papert, 1980, p. 8). In comparison, CT concepts lack creativity and student-directed projects (Kahn, 2017). To summarize, CT concentrates on the importance of coding and computer science activities, thus delivering concepts that are more applicable and successful on a number of levels (Tedre and Denning, 2016). However, critics argue that coding should not be seen as a unitary skill but instead as a meta-skill for a complex network of other skills.

Creative Environments to Reinforce Female Teenagers in Coding

Teaching good game design and development skills is especially important for girls because they are not that likely to play games (Krieger, Allen and Rawn, 2015). In addition, the literature argues that even if the number of female students who plays video games increased in recent years, male students have a greater interest in playing games (Jenson, Castell and Fisher, 2007). However, a recent study which examined American female players’ experiences showed that 65% of women play mobile games, compared to 2011 when only 31% of women played mobile games (Google and NewZoo, 2017). Furthermore, 64% of women prefer smartphones to other platforms (38% of the men do so). Thus, to use smartphones to design mobile games seems to be a very promising approach to attract young females. Furthermore, framing a supportive classroom setting for coding activities is a critical factor, and literature suggests that interventions should specifically target the classroom climate to strengthen teenage girls’ confidence to motivate female students extrinsically as well (Beyer *et al.*, 2003). Such first positive experiences in coding may direct their future career choices towards STEM fields. If they become game designers and creators of their own learning content through a constructionist, creative, and engaging learning environment, this can significantly contribute to closing the divide and participation gap in digital culture (Veilleux *et al.*, 2013, Allison, Cheryan, and Meltzoff, 2016). Thus, the literature states that for young women, creativity and interest in STEM professions are often related, but there is a lack of practical relevance in a lot of these subjects. A European-wide study in which 11,500 young women between 11 and 30 were interviewed (Microsoft, 2017) showed that girls between the ages of 12 and 16 are the most creative. Approximately every third women that has been asked (33%) criticized how scientific topics were explained in schools and that those subjects are taught from a more “male perspective”. Thus, it is important to make CS more attractive for girls in order to sustain a balance in this very strongly male-dominated IT labor market. It is therefore important to eliminate the prejudice that STEM professions are not creative. Through game design activities, creative environments, and customization, a broader spectrum of girls can be reached (Subsol, 2005). If facilitators promote especially IT careers that are more driven by creative thinking and design, more female students will expressed their interest in IT careers (Wong and Kemp, 2017).

The European No One Left Behind (NOLB) Project

The focus in the subsequent sections lies on the European No One Left Behind⁶¹ (NOLB) project and the Catrobat⁶² learning apps, Pocket Code and Create@School. The NOLB project has been funded by the Horizon 2020 framework and involved partners from Germany, Spain, the UK, and Austria. The vision of the NOLB project was to unlock inclusive game creation and to construct experiences in formal and informal learning situations from primary to secondary level, particularly for students at risk of social exclusion. This project started in January 2015 and reached its conclusion in June 2017 by validating its outcome in different phases: preparation phase, feasibility study, first, and second cycle. To limit the scope of topics to those relevant to this paper, the remainder of this section focuses on the results of the *second cycle* and on the *Austrian pilot*. In the past the authors' work concentrated on the feasibility study (Petri *et al.*, 2016), evaluation of performed Pocket Code Game Jams (Spieler *et al.*, 2016), and analyzing female teenagers' performance in regard to the learning goal achievement of submitted programs (Spieler, 2018).

Educational Coding Apps

Our app Pocket Code (Slany, 2014), is a visual programming language environment that allows the creation of games, stories, animations, and many types of other apps directly on smartphones or tablets, thereby teaching fundamental programming skills. Programs in Pocket Code follow a similar syntax to the one used in Scratch. During NOLB an enhanced version of the app has been developed and adapted for use in schools. This new version is called Create@School and integrated the results of the observations during the pilot studies as well as feedback from teachers and students. Furthermore, a web based Project Management Dashboard (PMD) for teachers was developed. The app was released in October 2016 for first test runs during the second cycle of the project. Important components of the new Create@School flavor of Pocket Code were the gathering of analytics data, the integration of accessibility preferences for children with special needs, and pre-coded game design templates. In previous work the authors focused on the evaluation of the Create@School features (Spieler *et al.*, 2017 [1]) and the teachers' perspective during NOLB (Spieler *et al.*, 2017 [2]). This paper focuses on the evaluation of the NOLB coding environments, thus the apps will not be explained in more detail.

In Austria, the NOLB project piloted in three different schools, in total, 478 students participated in Austria (281 female students and 197 male students). Altogether, 22 coding courses were conducted, and the coding apps were integrated in the curricula of English, computer science, physics, fine arts, and music (Spieler *et al.*, 2017 [2]).

Gender Inclusiveness in NOLB

The idea of NOLB included that game creation challenges in classes should enhance female students' abilities across all academic subjects, including logical reasoning, creativity, and the development of social and computational thinking skills (see previous section). Moreover, students had the opportunity to socialize with their peers during the game making process by working in teams. To address the gender bias in coding classes, the goal of the Austrian study included how to make the coding environment more suitable for female teenagers. From the literature, the assumption for NOLB was to attract girls by:

- Increasing their personal attachment to programming by improving our services with appropriate example games/templates, new assets, and themed tutorials/templates.
- Increasing their involvement by encouraging them to become active members of the Pocket Code community by providing them a safe and interesting environment to join, with featured games from female users for female users.
- Asking them to design their own games rather than just to code programs.

⁶¹ <http://no1leftbehind.eu/>

⁶² <https://www.catrobat.org/>

The intent was to discover how to organize the coding units in ways that specifically empower girls by engaging them with playful and creative activities.

Evaluation of NOLB Activities in Austria

In this section, the NOLB results from the Austrian case study are evaluated to investigate female students' experiences, behaviors, and outcomes when using our tools and services in different courses. With the help of a quantitative and qualitative survey, students' opinions about the user experience (UX) of the Create@School app and the courses in which the app was used has been collected to gain a deeper understanding of the experience evaluation. Overall, a total of 131 students filled out feedback forms in Austria during the second cycle (63 male and 68 female students). Based on the research design, a t-test was performed which compares whether two groups (female/male students) have different average values. The evaluation is part of the NOLB Delivery 5.4 (Spieler and Mashkina, 2017). Results are collected by answering like/dislike questions: *How was your experience with Create@School? (very good – very bad)*. Figure 3 illustrates students' opinions per answer in percentage. The percentage of female students (in orange) who rated the experience as "good" was 29% (38), and the percentage of male students (in cyan) account for 21% (27). The answer option "bad" was chosen by 13% of the girls (17) and 14% of the boys (19). In general, girls rated the app experience more positively (mean female = 2.63) as their male colleagues did (mean male = 2.37, but not significantly: $p = .130$, $\alpha = .05$).

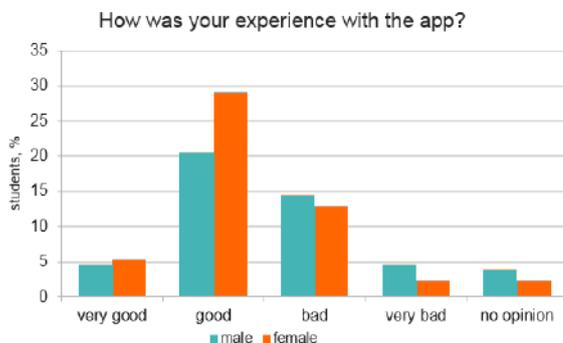


Figure 3. Distribution of answers about the experience with the Create@School app.

To clarify the motivation for these answers, it is necessary to take a closer look at the analysis of the open-ended questions: *What did you like the most?* The answers of the participants describe their positive impressions about the Create@School experience. Answers could be classified into five different categories: "working process", "the app", "the results (their game)", "organization", or "others". The distribution of the answers among the categories can be seen in Figure 4. This time a Mann-Whitney U test was performed to analyse whether the central trends of the two independent samples are different.

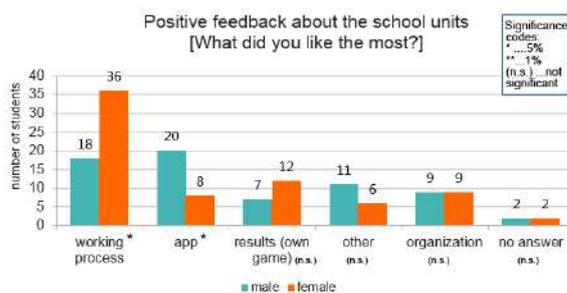


Figure 4. Categorization of the positive impressions during NOLB.

Note that one answer could contribute to two categories, for instance the feedback "the finished products + the facilitators" contributed to both, results and organization categories.

The category “working process” contains feedback about preferred actions (e.g., I like to program, to play the game, to design), or properties of the school units (e.g., having a freedom of choice, that you could be very creative). The largest statistically significant difference between the responses of male and female participants was observed for this category, namely 26% of the girls (36) and 13% of the boys (18) provided positive working process related feedback ($z = 2.402$; $p = .016$, $\alpha = .05$). The category “app” contains the answers mentioning the experiences directly connected with the app itself, e.g., “the simplicity of the app”, or “the different effects and backgrounds”. Male students seemed to be significantly more positively impressed by the app than girls were ($z = -1.969$; $p = .049$, $\alpha = .05$). A number of 8 girls and 20 boys evaluated the app qualities as positive. Some students were especially satisfied with the results of their work or the concept of game creation, and the category “results (own game)” contains this type of feedback. For instance, “the results and how everything turned out”, or “I liked the idea of creating a game” demonstrates this feedback. Of the girls, 9% (12) and 11% of the boys (7) were satisfied with the results or their personal game. The answers from the category “organization” highlights feelings about how the unit was structured. This includes if students were enjoying teamwork or solving the problems on their own, the presence of external people (facilitators of the workshop), usage of tablets during the school units, etc. It was noticeable that students in the younger age group of 12 - 14 years old were very excited about using the tablets during the school units. Typical responses for this category were, e.g., “to work in a team with your friends”, “when you (the facilitators) explained to us how the app really worked”, “that we had our own tablets”. The same number of the male and female participants’ feedback falls into the category of “organization” with nine responses from boys and from girls. The replies that could not be clearly classified into any of the described categories above were summarized into the “other” category. For example, answers with no clear message, or containing feedback connected to a particular game or classroom setting, as well as such responses as “nothing” or “everything”.

Next, we will take a closer look on the categories “working process” and “app”. The category “working process” was divided into six subcategories: “designing”, “creativity”, “playing”, “freedom of choice”, “programming”, and “other”. Figure 5 shows the distribution of the responses.

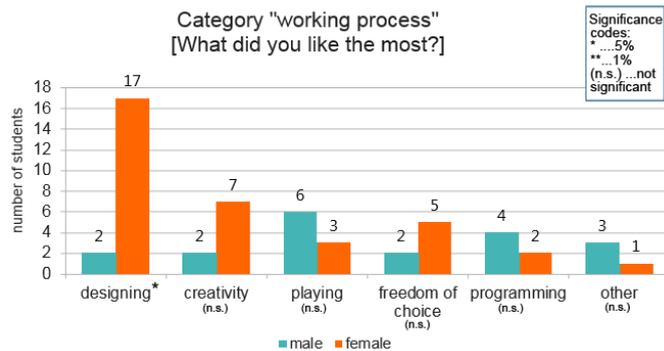


Figure 5. Detailed overview of positive impressions category “working process”

The subcategory with the name “designing” summarizes the feedback related to drawing, taking pictures, personalization of the characters, etc. For instance, “drawing our pictures” or “creating the characters” fall into this subcategory. There were 32% of the girls (17) and only two boys who identified design related activities as the essence of their positive experience (significant: $z = 2.151$; $p = .032$, $\alpha = .05$). The subcategory “creativity” represents the feedback praising the creative side of the school units, for instance, “you could be very creative”, “you could do almost any game you wanted”. A total of seven female students and two male students evaluated creativity as the positive aspect of the Create@School experience. A total of six boys and three girls stated that they considered “playing” as a positive experience. Answers like “you were allowed to play” were typical for this subcategory. No constraints in choice or actions were valued by two boys and five girls. These responses were summarized within the subcategory “freedom of choice”. A representative statement for this subcategory was “That you could do whatever you want”. There were four boys and two girls who enjoyed the programming process itself.

Typical responses for the subcategory “programming” were “programming the rocket” or “programming by ourselves”.

The category “app” was divided into four subcategories: “LEGO®-style bricks”, “features”, and “design”. The distribution of the feedback can be seen in Figure 6.

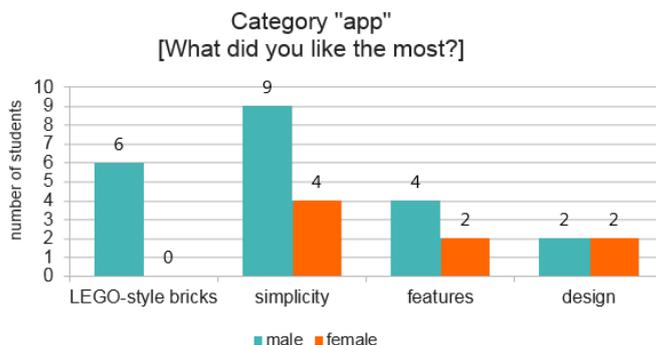


Figure 6. Detailed overview of positive impressions of the category “app”.

None of the girls but six boys stated that they were pleased about the LEGO-style bricks in Create@School. Although the number of answers is too small to be significant, the insights are interesting. The response “brick system” was representative for this subcategory. A total of nine boys and four girls liked the “simplicity” of the app. Feedback included “It was relatively self-explaining!”, or “components are easy to understand”. The subcategory “features” consists of answers like “the different effects and backgrounds” or “variables were available”. Four boys and two girls contributed to this subcategory. Equally, two of each gender liked the design of the app.

What did you like the least? Any suggestions for app improvement? The answers to the questions were the basis for the negative feedback evaluation. Note that some students responded with everything was fine (6) and some did not give any answer to this question (10). Figure 7 categorizes the negative impressions about the app and the school units.

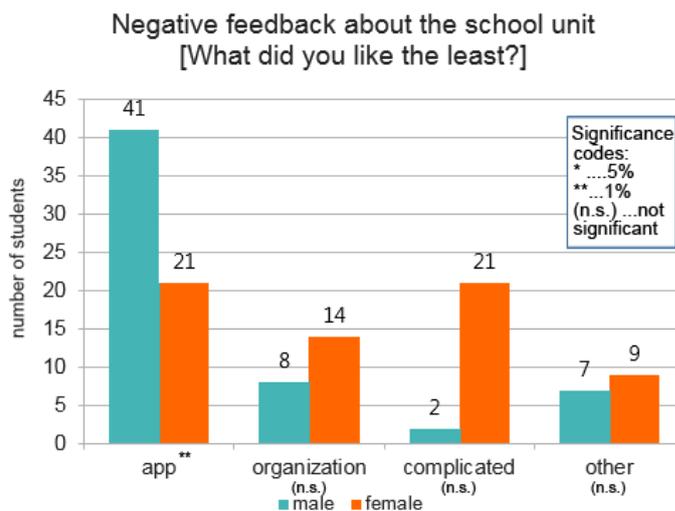


Figure 7. Categorization of the negative impressions about the app and the school unit.

Note that one answer can contribute to two categories, for instance the feedback “It was too complicated and it took too long to finish the game” contributed to both the “organization” and “complicated” categories.

The category “app” consists of the feedback related only to the Create@School app, for example, “it looked kind of tacky, complicated, confusing”, “the axis of the screen”, or “it was buggy”. There were 33% of the boys (41) and 17% of the girls (21) who provided this kind of feedback (significant: $z = -3.372$; $p = .0007$, $\alpha = .01$). Responses such as “it took too long to finish up the game”, “introduction of

the app was boring”, or “The instructors were not able to explain everything to us” about the unit were summarized into the category “organization”. There were 8 boys and 14 girls who gave this kind of responses. The feedback of the contents of the type “I found some things complicated”, “make it simpler for people that are not technical”, or “you needed help a lot” were summarized under the category “complicated”. There were 21 girls (17%) and only 2 boys who gave this type of feedback about their experience with Create@School. The replies that could not be clearly classified into any of the categories described above were summarized into “other”, for example, the answers with no clear message, or complaints about the devices and other apps. A breakdown of the category “app” into subcategories can be seen in Figure 8.

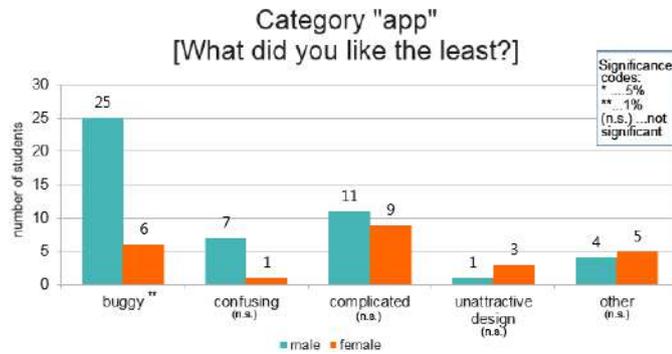


Figure 8. Detailed overview of negative impressions of the category “app”.

The subcategory “buggy” contains the feedback about the app behavior that students considered as bugs, responses about slow performance, and mentions of app crashes. 32% of the boys (23) and only 8% of the girls (6) gave this type of feedback about the app (significant: $z = -2.73$; $p = .006$, $\alpha = .01$). The representative answers for this subcategory were “Create@School crashes ALL the time”, “It sometimes stopped, didn’t always work smoothly”, or “it took AGES to load”. Seven boys and one girl stated that the app lacks structural clarity, and is confusing, e.g. “the app totally lacks the structure”, or “some things should be easier to find”. These answers were summarized within the subcategory “confusing”. Nine girls and eleven boys complained about the complexity of the app (subcategory “complicated”). Comments characterizing the subcategory “unattractive design” were, for example “It looked kind of tacky” or “the design is rather boring”. The subcategory “other” contains all other responses that were related to the app, e.g., missing features (“the axes of the screen should be like in Scratch”).

Conclusion and Discussion

The results of the UX experiences with Create@School showed many significant responses and helped the authors to shape future workshops. Overall, the answers about likes and dislikes can be clearly separated by gender. On the one hand, female students significantly preferred the aspects of the Create@School units (“working process”) and mostly related to design activities, but did not mention the programming aspect explicitly. Negative impressions from female students concentrated more on the organization of the units or on the level of complexity, and less on the app itself. On the other hand, male students mentioned app related aspects, like programming and Create@School’s simplicity, as positive impressions significantly much more often than the units. Their dislikes significantly concentrated on the app, e.g., that it was buggy. This reflects the complaints of only six girls. Surprisingly, answers from girls also included that they did not like coding at all, whereas this was not an answer given by boys. The low performance and high error rate of the app is still a serious issue, but seems to have been more problematic for the male students. Possible reasons are again that boys are more used to utility apps (“tools”) and game engines than girls are (Krieger, Allen, and Rawn, 2015).

To conclude, there are statistically significant different aspects which are more important for girls than for boys. In the literature review, the authors already described that CS lessons are mostly constructed to suit the interests of males. However, this evaluation shows that the working process and the sequence of the units are particularly important for engaging female students. Thus, for them not only is the tool essential but the learning environment as a whole, as well as the ability to express their own

interests, e.g., through designing and creative activities. Based on these statistically significant results and our analysis of the existing literature, and after the completion of the NOLB project, we started to design more suitable learning environments for girls, particularly focussing on aspects of gender sensibility and awareness (McLean and Harlow, 2017). We shortly describe them in the next section and will publish a more detailed description in a future paper.

Outlook

One result of NOLB, the Create@School app, was a great opportunity to provide schools with a tailored package of tools and services to help them integrate coding in their classrooms and to apply the app for interdisciplinary project work. This is important for the future to support teachers in Austria with the upcoming challenge to integrate a basic set of digital literacy education⁶³ in secondary schools. After NOLB, the authors concentrated their research on how to apply the NOLB results to tailor the Catrobat services to female needs. Therefore, we have developed a new course model that includes extrinsic and intrinsic motivators as well as four key fields which have been considered as important for coding activities: playfulness, engagement, creativity, and coding. This model has already been tested with two classes of 23 students (one mixed gender class and one girls-only class). This sample clearly was very small, and thus we plan further larger studies in the same and in different contexts (outside school). As a first step, an intensive “Girls Coding Week” will be performed later in 2018 to test the model’s efficiency.

References

- Allison, M., Cheryan, S., and Meltzoff, A.N. (2016) Computing Whether She Belongs: Stereotypes Undermine Girls’ Interest and Sense of Belonging in Computer Science. In: *Journal of Educational Psychology*, Vol. 108, No. 3, p. 424–437.
- Balanskat, A., and Engelhardt, K. (2015) Computing our future. Computer programming and coding Priorities, school curricula and initiatives across Europe. *European Schoolnet*.
- Blikstein, P., and Krannich, D. (2013) The Makers’ Movement and FabLabs in Education: Experiences, Technologies, and Research. In Proceedings: *12th International Conference on Interaction Design and Children*. New York. June, p. 613-616.
- Beyer, S., Rynes, K., Perraul, J., Hay, K., and Haller, S. (2003) Gender differences in computer science students. In: *SIGCSE Bull*, Vol. 35, No. 1, p. 49–53.
- European Commission (2016) A new skills agenda for Europe. Working together to strengthen human capital, employability and competitiveness, [online] <http://ec.europa.eu/social/main.jsp?catId=1223>, accessed: 1.3.2018.
- European Commission (2014) Key priorities, [online]: <https://ec.europa.eu/digital-single-market/key-priorities-grand-coalition>, accessed: 1.3.2018.
- Google and NewZoo (2017) Change the Game. THE WORLD OF WOMEN AND MOBILE GAMING. A White Paper. [online] http://services.google.com/fh/files/misc/changethethegame_white_paper.pdf, accessed: 8.3.2018.
- Informatics Europe/ACM Europe (2016) Informatics Education in Europe: Are We All in the Same Boat? Report by The Committee on European Computing Education (CECE) Jointly established by Informatics Europe & ACM Europe. [online] <http://www.informatics-europe.org/news/382-informatics-education-in-europe-are-we-on-the-same-boat.html>, accessed: 1.3.2018.
- Informatics Europe/ACM Europe (2013) Informatics education: Europe cannot afford to miss the boat. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education, [online] <http://www.informatics-europe.org/images/documents/informatics-education-acm-ie.pdf>, accessed: 1.3.2018.

⁶³ Mandatory exercise “Digital literacy” in secondary education 1, Content for piloting in the school year 2017/18, [online] <https://tinyurl.com/y78wov7a>, accessed: 8.3.2018.

- Jenson, J., de Castell, S., and Fisher, S. (2007) Girls playing games: Rethinking stereotypes. In *Proceedings: 2007 conference on Future Play (Future Play '07)*. ACM, New York, USA, p. 9-16.
- Khan, N.Z., and Luxton-Reilly, A. (2016) Is computing for social good the solution to closing the gender gap in computer science? In *Proceedings: Australasian Computer Science Week Multiconference*, New York, USA, Article 17, 5 pages.
- Kahn, K. (2017) A half-century perspective on Computational Thinking, In: *The Future of Learning, Music Learning with Massive Open Online Courses (MOOCs)*, Vol. 6, No 14 , p. 213-224.
- Kay, A., Rose, K., Ingalls, D., Kaehler, T., Maloney, J., and Wallace S. (1997) Etoys & SimStories. In: *ImagiLearning Group, Walt Disney Imagineering*
- Krieger, S., Allen, M., and Rawn, C. (2015) Are females disinclined to tinker in computer science? In *Proceedings: 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, p. 102-107.
- Neo, M., and Neo, T.K. (2009) Engaging students in multimedia-mediated Constructivist learning – Students' perceptions. In: *Educational Technology & Society*, Vol. 2, p. 254–266.
- Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., and Settle, A. (2014) Computational thinking in K–9 education. In *Proceedings: Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, ITiCSE-WGR '14*, p. 1-29.
- McLean, M., and Harlow, D. (2017) Designing Inclusive STEM Activities: A Comparison of Playful Interactive Experiences Across Gender. In *Proceedings of the 2017 Conference on Interaction Design and Children (IDC '17)*, p. 567-574.
- Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. (2010) Learning computer science concepts with scratch. In *Proceedings of the Sixth international workshop on Computing education research*, p. 69-76.
- Microsoft (2017) Why don't European girls like science or technology?, [online] <https://news.microsoft.com/de-de/microsoft-studie-mehr-frauen-mint-berufen/>, accessed: 1.3.2018.
- Papert, S. (1993) *The Children's Machine: Rethinking School In The Age Of The Computer: Bringing the Computer Revolution to Our Schools*. Basic Books.
- Papert, S., and Harel, I. (1991) *Constructionism*. New Jersey: Ablex Publishing Corporation.
- Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Pavlov, I.P. (1927). *Conditioned reflexes* (G. V. Anrep, Trans.). London: Oxford University Press.
- Perry, W.G. (1999) *Forms of Ethical and Intellectual Development in the College Years*. San Francisco: Jossey-Bass Publishers.
- Petri, A., Slany, W., Schindler, C., and Spieler, B. (2016) Game Design with Pocket Code: Providing a Constructionist Environment for Girls in the School Context. In *Proceedings: Constructionism 2016*, Bangkok, Thailand, p. 109-116.
- Piaget, J. (1968) *Six Psychological Studies*. Anita Tenzer (Trans.), New York: Vintage Books.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009) Scratch: programming for all. *Commun.* November, Vol. 52, No 11, p. 60-67.
- Schunk, D.H. (2016) *Learning Theories: An Educational Perspective*, Pearson eText, 7th edition.
- Skinner, B.F. (1976) *About Behaviorism*. New York: Vintage Books.
- Slany, W. (2014) Tinkering with Pocket Code, a Scratch-like programming app for your smartphone. In *Proceedings: Constructionism 2014*, Vienna, August 2014.

- Subsol, G. (2005) Virtual Storytelling – using virtual reality technologies for storytelling, In Proceedings: 3rd international conference, ICVS 2005, Strasbourg, France, p. 251-259, Springer Verlag Berlin Heideberg.
- Spieler, B. (2018) Reinforcing Gender Equality by Analysing Female Teenagers' Performances in Coding Activities: A Lesson Learned, In Proceedings: *Conference on Gender IT 2018 / GEWINN-Konferenz 2018*, Heilbronn, Germany, May 2018, 10 pages.
- Spieler, B., and Mashkina, O. (2017) D5.4 – Report and findings from experimental pilot in Austria. Agreement 645215.
- Spieler, B., Schindler, C., Slany, W., Mashkina, O., Beltrán, M.E., Boulton, H., and Brown D. (2017 [1]) Evaluation of Game Templates to support Programming Activities in Schools, In Proceedings: *11th European Conference on Games Based Learning*, 5-6 October 2017, Graz, Austria. p. 600-609.
- Spieler, B., Schindler, C., Slany, W., and Mashkina, O. (2017 [2]) App Creation in Schools for different Curricula Subjects - Lessons Learned, In *Proceedings 9th International Conference on Education and New Learning Technologies* (Edulearn17), 3-5.July 2017, Barcelona, Spain, p. 5814-5824.
- Spieler, B., Petri, A., Slany, W., Schindler, C., Beltrán, M.E., and Boulton H. (2016) Pocket Code: A Mobile App for Game Jams to facilitate Classroom Learning through Game Creation. In Proceedings: *The Irish Conference on Game-Based Learning (iGBL)*. Dublin. Ireland, p. 61-79.
- Tedre, M., and Denning, P.J. (2016) The Long Quest for Computational Thinking. In Proceedings: *16th Koli Calling Conference on Computing Education Research*, November 24-27, Finland. p. 120-129.
- Tinker, R.F., and Papert, S. (1989) Tools for science education. In J. D. Ellis, editor, *Information Technology and Science Education. 1988 AETS Yearbook*, Chapter 1, p. 1-23. Columbus: SMEAC Information Reference Center (SMEAC/IRC), The Ohio State University, 1989.
- Tsan, J., Boyer, K.E., and Lynch, C.F. (2016) How Early Does the CS Gender Gap Emerge?: A Study of Collaborative project Solving in 5th Grade Computer Science. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). ACM, p. 388–393.
- Veilleux, N., Bates, R., Allendoerfer, C., Jones, D., Crawford, J., and Floyd Smith, T. (2013) The relationship between belonging and ability in computer science. In Proceeding: 44th ACM technical symposium on computer science education (SIGCSE '13). ACM, New York, NY, USA, p. 65-70.
- Vygotsky, L. (1978) *Mind in Society*. London: *Harvard University Press*.
- von Foerster, H., von Glasersfeld, E., Heijl, B.M., Schmidt, S.J., and Watzlawick, P. (2009) *Einführung in den Konstruktivismus*, 11th Edition, München: *Piper*.
- Wing, J.M. (2008) Computational thinking and thinking about computing. In: *Philosophical Transactions of the Royal Society A*, Vol. 36, No 1881, p. 3717–3725.
- Wing, J.M. (2006) Computational thinking. In *Commun. ACM*, Vol. 49, No 3, p. 33-35.
- Wong, B., and Kemp, P.E.J. (2017) Technical boys and creative girls: the career aspirations of digitally-skilled youths. *The Cambridge Journal of Education*. ISSN 1469-3577, Available [online] <http://centaur.reading.ac.uk/70247/>

Visualizing Mathematics with the MathBot: a Constructionist Activity to Explore Mathematical Concepts through Robotics

Christina Todorova, *tina@esicenter.bg*

European Software Institute – Center Eastern Europe, Sofia, Bulgaria

Carina Girvan, *girvanc@cardiff.ac.uk*

School of Social Sciences, Cardiff University, Wales, UK

Nikoleta Yiannoutsou, *nyiannoutsou@ppp.uoa.gr*

Marianthi Grizioti, *mgriziot@ppp.uoa.gr*

UoA ETL, National and Kapodistrian University of Athens, Athens, Greece

Ivaylo Gueorguiev, *ivo@esicenter.bg*

Pavel Varbanov, *pavel@esicenter.bg*

George Sharkov, *gesha@esicenter.bg*

European Software Institute – Center Eastern Europe, Sofia, Bulgaria

Abstract

In this practice paper, we aim to share our experience with the design and implementation of constructionist educational robotics activities tailored to primary school students (4th grade, age 9-11 years) implemented in a series of robotics workshops, which took place within a real school setting in Sofia, Bulgaria.

Through this contribution, we will further present an activity plan, which involves student engagement with mathematical concepts (angle measuring and properties of the circle) in order to program the behaviour of a robot. Our paper reports insights on the implementation of the activity plan focusing students' evaluation of their experience during the workshop. These insights are drawn from quantitative data from 131 participants (63 boys and 68 girls), capturing the overall student attitude.

The activity plan behind this set of educational robotics workshops was designed, adapted and piloted in alignment to the guidelines of the Bulgarian national curriculum for mathematics for the 4th grade. However, it could function as a practical example, which could be adapted, enriched and modified to benefit other age groups, nationalities and desired learning outcomes.

Keywords

educational robotics; mathematics; programming; Scratch; primary education

Introduction

The increasing difficulty related to maintaining students' positive attitude and motivation towards the educational process is not a new topic neither for the Bulgarian education scene, nor internationally. Furthermore, against the backdrop of the rapidly changing technological landscape, which demands the continuous improvement of computer science curricula in general education schools, providing students with a functional understanding of how technology works becomes a paramount and a challenge.

The ICT general education in Bulgaria is already undergoing a renaissance of a sort with subjects such as computer modelling⁶⁴ with Scratch for primary school students being gradually included within the official school curriculum. Furthermore, curriculum strategies are being developed in order to include a multidisciplinary approach towards learning, an example being the Ministry of Education's "Innovative Schools"⁶⁷ initiative granting schools a certain level of curriculum flexibility and allowing them to include innovative learning approaches and multidisciplinary subjects.

At the same time, there is a worldwide turn to robotics and STEM education alike, with a big number of robotics kits launched during the last few years. According to the International Federation of Robotics in October 2016, from 2014 to 2015, unit sales of entertainment robots, including educational robotics kits jumped by 29 percent to around 1.7 million units. The entire segment is projected to increase to a total of 11 million units (2016-2019). The sales value adds up to around US\$ 9 billion in the same period.⁷⁰ Robotics activities for STEM have been considered as interdisciplinary with a strong emphasis on collaborative learning, and robots have been perceived as a potentially powerful "vehicle" providing opportunities for construction, experimentation and collaboration between learners (Alimissis 2013). However, most of the robotics kits and activities tend to focus mainly on Programming and Engineering leaving aside the Mathematics and Science elements of STEM education. Additionally, in many educational robotics activities the element of construction and active exploration is limited as they are based on instructional, "step-by-step" type of exercises and closed quizzes. Nevertheless, the field of robotics bares an enormous educational potential as children's fascination for robots, along with the variety of fields and topics covered by robotics (Johnson, 2003), make robots a powerful idea to engage with, including gears and computers (Papert, 1980). At the same time, robotics is an excellent tool for teaching science and technology (Mataric, 2004; Mead 2012), therefore many educational robotics activities already focus on STEM (Baretto, 2012), which unfortunately attracts predominantly children who have not lost their interest in these subjects.

We do believe, however, that robots are a powerful tool for teaching STEM disciplines not only to the children who are already interested in those subjects. Thus, inspired by the spirit of the undergoing educational reform but also by the current state of educational robotics, involving mainly students with established interest in STEM, under the ER4STEM project, we undertook an exploration on how robotics activity plans could be designed to support teaching concepts, stipulated within an official mathematics curriculum. With the term activity plans, we refer to structured descriptions of robotics activities, which are built on the ER4STEM Activity Plan Template. This template provides a generic design tool that identifies critical elements of teaching and learning with robotics based in theory and practice and is expected to contribute to the description of effective learning and teaching with robotics (Yiannoutsou et al., 2017).

In this practice paper, we begin by introducing an activity plan, applied to a set of educational robotics workshops carried out between November and December 2016, within one public general education school in Sofia, Bulgaria. Each of the six 4th grade classes went through one educational workshop which comprised two 4-hour sessions. By involving experimentation with virtual and physical constructions within the activity, serving as tools of learning, we attempted to design an activity plan, which helps expand student's access to powerful ideas, in the sense of being immediately useful to the learner, by association to other productive ideas (Papert, 1980, 2000). We believe that this activity plan

⁶⁴ Ordinance No 5 of 30.11.2015 on general education in force from 08.12.2015 issued by the Minister of Education and Science, obtained online in Bulgarian on March 10, 2018 at http://zareformata.mon.bg/documents/nrdb5_30.11.2015_obshtoobr_podgotovka.pdf

⁶⁵ Educational curriculum for computer modelling for the third grade issued by the Minister of Education and Science, obtained online in Bulgarian on March 10, 2018 at https://www.mon.bg/upload/12205/UP_KM_3kl.pdf

⁶⁶ Educational curriculum for computer modelling for the fourth grade issued by the Minister of Education and Science, obtained online in Bulgarian on March 10, 2018 at https://www.mon.bg/upload/13660/pr_UP_KM_4kl.pdf

⁶⁷ Again in Ordinance No 5 of 30.11.2015 on general education in force from 08.12.2015 issued by the Minister of Education and Science, obtained online in Bulgarian on March 10, 2018 at http://zareformata.mon.bg/documents/nrdb5_30.11.2015_obshtoobr_podgotovka.pdf

⁶⁸ Guidelines for application for inclusion in the network of innovative schools in Bulgaria for the academic year 2018/2019, obtained online in Bulgarian on March 10, 2018 at https://www.mon.bg/upload/12192/Nasoki_2018_2019_iSchools.pdf

⁶⁹ COUNCIL DECISION No 391 from 17 July 2017 on adopting a list of innovative schools for the academic year 2017/2018 (promulgated, SG No. 60/25 July 2017), obtained online in Bulgarian on March 10, 2018 at https://www.mon.bg/upload/12193/rms_391_17072017.pdf

⁷⁰ International Federation of Robotics, Press Release, Seoul, Oct 12, 2016, obtained online in English on March 10, 2018 at <https://ifr.org/ifr-press-releases/news/service-robotics>

could be adapted to fit a variety of educational and cultural contexts and might be of use to educators looking for strategies to improve their student's involvement in subjects such as mathematics.

In addition to the pedagogic goals of this set of workshops, qualitative and quantitative data were collected through pre- and post-activity questionnaires completed by students (63 boys and 68 girls), and 6 groups of learners participated in semi-structured interviews to explore students' interactions and feedback. For the purposes of this paper, we will focus on data collected through questionnaires at the end of the workshops. The specific question we aimed to answer was what are Bulgarian students' attitudes towards constructionist robotics activities.

Finally, in this contribution we further share our experience with integrating collaborative work, aimed at developing an activity that could support the cultivation of some important 21st century skills within students (Dede, 2010), namely problem solving, collaboration, flexibility and adaptability.

Implementation Context

Overview

This activity plan was developed for boys and girls (age 9-11 years) studying at the 4th grade of one public general education school in Sofia, Bulgaria in 2016. The students have previously participated in educational robotics activities carried out by the implementation team and are expected to have previous experience programming with Scratch (however, previous programming knowledge was not required). They were also familiar with the collaboration requirements of the activity as well as the research protocol. This activity was developed following the guidelines of the Bulgarian national curriculum for mathematics for the 4th grade, thus knowledge of third grade mathematics was a prerequisite for the participation.

The educational robotics activities took place within regular school time. For each 4th grade class a separate workshop was organized, thus resulting in six workshops in total. Every workshop was of an eight-hour duration, with breaks, and was divided into two sessions of equal duration, with not more than one-week time between sessions. A regular school class in a public general education school in Bulgaria consists of anywhere within 24 and 28 students. Each session was led by 2-3 tutors.

Following our strong belief that the collaborative work plays a very important role for the effective engagement of students in the educational process, we apply the engagement theory (Kearsley & Shneiderman, 1998) as a significant part of the Constructionist teaching methods. Based also on their empirical conclusions that "technology can facilitate engagement in ways, difficult to achieve otherwise" (Kearsley & Shneiderman, 1998), we also asked students to assess their feelings on problem-based learning, working with robots and collaborative work.

Bulgaria's Mathematics Curriculum for the 4th Grade for General Education Schools

The fourth-grade curriculum in mathematics of Bulgaria⁷¹ is designed by the Ministry of Education in accordance to the State Educational Requirements and is developed in accordance to a set of principles, several of which served as a common ground for the design of the activity plan for the educational robotics workshops:

- Arithmetic operations with one and two-digit numbers and the properties of these actions;
- Deepening and expanding knowledge on the units of measurement by introducing angle measurement;
- Deepening and expanding the knowledge of the basic geometric shapes, types of triangles, types of angles, rectangle, square, knowledge of the circle and its elements
- Making evident how mathematics connects to subjects of other cultural and educational fields;

⁷¹ Educational curriculum for mathematics for the fourth grade issued by the Minister of Education and Science, obtained online in Bulgarian on March 10, 2018 at https://www.mon.bg/upload/2645/matematika_4kl.pdf

- Improving collaboration, communication, formulating and expressing ideas and cultivating skills such as observance, logical thinking, comparative thinking, critical analysis, formulating statements, making conclusions, experimentation;
- Develop interest and motivation to study mathematics and form a positive attitude towards the subject;

Following these guidelines, we created a multi-layered activity plan, to support the mission of the curriculum and possibly transfer this experience to fourth-grade mathematics educators, in order to contribute to the student's confidence in the subject and to inspire their curiosity in it.

Visualizing Mathematics with the MathBot

A Constructionist Activity Plan for Mathematical Experimentation

The educational robotics activities presented here are structured around the constructionist pedagogical approach, where the activity is organized around powerful ideas, facilitated through collaborative and group-centred games. Our focus was put on promoting a social dimension of robotics, which includes sharing personal knowledge and experiences on STEM related concepts while learning from others (Kafai & Burke, 2015). In this context, robotic construction and programming are seen as expressive medium to externalize ideas and thoughts, which through the sharing are becoming objects for discussion and change (Kynigos, 1995). Thus, all the activities are designed to take place in a common space and integrate virtual and physical constructions (Papert, 1980, 2000).

The pedagogical approach, as well as the background for the elaboration of the educational robotics workshops within the ER4STEM project are coordinated and structured with the means of an Activity Plan Template. This template was developed in the project as a design tool for planning robotics activities and depicts what we have identified as essential and transferrable elements of learning with robotics.

The Activity plan template, addresses the following aspects: a) Focus and resources: reference to the different domains involved, different types of objectives, duration and necessary material; b) contextual information regarding space and characteristics of the participants; c) social orchestration of the activity (i.e. group or individual work, formulation of groups etc.); d) a description of the teaching and learning procedures where the influence of the pedagogical theory is mostly demonstrated; e) expected student constructions; f) description of the sequencing and the focus of activities; g) means of evaluation (Yiannoutsou et al., 2017)

Social Orchestration and the Role of Discussion

The students were assigned to groups, each group consisting of 3-4 members. Every group was working around one table with one computer and one robot. We applied no specific grouping criteria, as we wanted the students to be able to sit together based on pre-established friendships. We wanted students to be positively influenced by the idea that the workshops they were participating in were a fun group activity, which they could share with their friends.

As the students in one general education school class are about 25 boys and girls, this resulted in the formation of about 6-7 student groups in total.

Students were given discussion opportunities to reinforce their understanding by sharing with peers their constructions and working strategies. Students were supposed to verbally, and visually present their solutions to the tutors and the other teams, which required them to communicate their learning curve and solution in an understandable and clear manner. Student groups were encouraged to invite other student groups to present their solution and could go and ask for consultation with other student groups. By answering questions from other students and student groups, the presenting group was required to be able to explain their reasons and articulate their chosen strategy. The ability to clearly identify and communicate the reasoning underlying group decisions is at the core of the "thinking about one's own thinking" (Han & Bhattacharya 2001) constructionist approach, which we aimed to integrate in our activity. Furthermore, in relation to the subject of mathematics itself, language can facilitate

reflection and internal regulation, which is of importance to realize which parts of the mathematical idea are important (Hoyles, 1985).

Furthermore, we wanted our students to remain motivated to learn and participate throughout the activity. Motivation and understanding are shown to be increased when encouraged to explain knowledge (Brown, 1988). By enabling such discussions, the robot's behaviour or the code is becoming a subject of common reflection, which empowers learning by providing the learner with an expertise to share.

To this end, apart from meaning generation we argue that constructionist robotics activities can also help the development of a number of 21st century skills including problem solving, flexibility and adaptability. With flexibility and adaptability, we refer to the abilities of a) incorporating feedback effectively, understanding, negotiating and balancing diverse views and beliefs to reach workable solutions and b) adapting to varied roles, schedules and context. In our workshops, we aim to reinforce these abilities through collaborative reflection, public discussion, role rotation and redistribution.

Implementation

For this particular set of workshops, we chose to use the Finch Robot, designed and developed by Birdbrain Technologies. Among the advantages of the Finch Robot is its cost-effectiveness, durability and the minimal amount of required knowledge or preparations in order to install all necessary software and run it. The Finch Robot is a black-box technology, equipped with many programmable sensors to support learning activities that emphasize programming the behaviour of the robot.

The Finch robot is programmable through the offline version of the visual programming language Scratch, which makes it highly accessible to students of this age group with little to no experience with programming and discrete logic. Furthermore, as Scratch allows the students to focus on tackling mathematical problems and easily create logical constructions, which becomes increasingly challenging with other, non-visual programming environments (Utting et al., 2010), where the focus falls on the code itself, thus making the programming process a purpose, instead of a tool for learning.

The activity plan was structured in seven phases of gradual complexity, apart from the first and the last one, which were introductory and evaluation-related. Every phase offered games with multiple sub-quests, which allowed for differentiation and made it possible for all groups to learn and problem-solve at their own pace. As cultural context, the activities were designed for Bulgarian students or students representing Bulgaria-based minorities and required fluency in Bulgarian language as game tasks are written in Bulgarian.

Phases 2 and 3 were more programming-oriented, allowing the students to reach a level of comfort in programming the robot and included basic command of the robot, programming it to perform a sequence of actions, programming its nose to blink in different colours while performing certain movements and others. Phase 4 included games related to drawing and measuring lines with the robot and rulers. In phase 5, students are engaged in a mathematical game of measuring the angle at which the robot turns with a protractor. Phase 6 aimed at exercising knowledge on the elements of a circle and included drawing circles with the robot and measuring its elements.

Ultimately, at the end of the workshop students gain understanding of what a robot is and know some common robotic parts. They understand that robots are programmable and gain knowledge on sensors and some different types of sensors and are aware that different sensors serve different purpose. At the end of the workshop, students would have exercised basic age-appropriate concepts from the subject of mathematics.

The full activity plan, with more information about all phases is available at repository.er4stem.com.

Activity Evaluation

In order to be able to evaluate the activities and answer our research questions data was collected from students whose parents had given informed written consent. From 6 workshops held with different classes, data was obtained from 131 participants (63 boys and 68 girls). The majority of them had

previously participated in educational robotics workshops in Bulgaria organized by the European Software Institute, under the ER4STEM project.

Quantitative data was obtained through pre and post workshop questionnaires. Interviews were conducted with one focus group from each class (6 interviews in total) and were transcribed, allowing more insight of the participants' experience and immediately after participating in the workshops. Additionally, reflections and observations were collected from tutors, along with student reflections and artefacts of learning.

This section presents the findings of the analysis of the questionnaires aims to provide insight on the participants' experience. It should be noted that some participants chose not to give an answer to some questions. Considering this, the percentage of students who left a question blank will be indicated next to the answers' report.

It is noteworthy to mention that this paper **will report on insights related to the implementation of the activity plan** focusing students' evaluation of their experience during the workshop. The activity's quantitative evaluation was carried out under the ER4STEM evaluation protocol, which is not specifically targeted at assessing student's understanding of mathematical concepts or student's attitude towards mathematic in general. The quantitative evaluation results reported below focus more, but not only on exploring boys and girls' interests in STEM, whether the approaches/activities appeal to them, on reporting on their intra-relational and interpersonal skills, and their feelings towards collaborative work experience.

Students' understanding of mathematical concepts, specifically for this activity, is qualitatively assessed through open answers in the questionnaires, focus group interviews, by collecting artefacts of learning, tutor observations and group reflections. The qualitative data obtained throughout ER4STEM workshops under analysis (May 2018) and will be reported on following the project.

What are Bulgarian students' attitude towards constructionist robotics activities?

The majority of the students who participated in the study reported the educational robotics workshop activities for mathematics as interesting, fun and not very difficult.

	Strongly disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly agree	Blank
The problems we had to solve were:						
Interesting	1%	0%	2%	9%	88%	0%
Difficult	39%	20%	28%	8%	3%	0%
Fun	2%	0%	3%	10%	83%	0%
Working with robots was:						
Interesting	1%	0%	0%	8%	89%	0%
Difficult	42%	23%	22%	8%	5%	0%
Fun	2%	1%	0%	8%	89%	0%
Working in a team was:						
Interesting	3%	0%	8%	16%	71%	0%
Difficult	44%	22%	15%	8%	7%	0%
Fun	5%	1%	7%	16%	68%	0%

Figure 1. Aggregated students' feedback related to the problems solved, their work with robots and teamwork

Most students reported working in a team as interesting, fun and not difficult (Figure 1). Tutors' observation show that regardless of group conflicts that inevitably arose in some groups, the students generally manifested positive attitudes towards group work and felt excited for the opportunity to work in a team.

	Strongly disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly agree	Blank
During the workshop...						
I solved a problem	9%	5%	23%	15%	45%	3%
I worked as part of a team	4%	2%	5%	14%	73%	2%
I programmed a robot	2%	1%	2%	4%	89%	2%
I was good at listening	2%	2%	15%	27%	50%	3%
I gave up quickly	80%	12%	3%	1%	3%	2%
I worked hard	3%	3%	8%	21%	64%	2%
I was bored	74%	12%	7%	5%	1%	2%
I helped someone	6%	2%	20%	24%	48%	1%

The students also gave a relatively high rating to the workshop they participated in with 4.93 stars out of 5.

Some students also reported increased confidence in mathematics and felt encouraged to participate in more activities such as this one.

Closing remarks

In this practice paper, we concentrated on providing insights on the design, implementation and pedagogical theory underlying an educational robotics activity, designed in support to the guidelines of a national educational curriculum in mathematics. We further provided aggregated feedback data reporting on students' experience with this activity, obtained by the primary school students who took part in the activities and we have given a brief overview on the evaluation protocol of the activities.

As practitioners in the field of educational robotics, we realize the importance of applying "powerful ideas" (Papert, 1980) as tools for learning. Through the Visualizing Mathematics with the MathBot activities, we aim to engage students in experimentation with digital and robotics artefacts to solve mathematical problems, in order to construct a better understanding of the interdisciplinary application of mathematical concepts and to further develop a collaborative mind set to problem-solving. We believe that constructionist activities such as this one, bare the potential to enhance the properties of the linked curricula, and the possibility of adapting such activities to a plethora of robotics kits, programming environments and learning contexts makes this activity an idea worth sharing.

References

- Alimissis, D. 2013. Educational robotics: Open questions and new challenges. *Themes in Science and Technology Education*, 6 (1), 63-71
- Barreto, F. and Vavassori, B. 2012. Exploring the educational potential of robotics in schools: A systematic review. *Comput. Educ.* 58, 3 (April 2012), 978-988.
- Brown, A.L., 1988. Motivation to learn and understand: On taking charge of one's own learning. *Cognition and Instruction*, 5, pp.311–322.
- Dede, C. (2010). Comparing frameworks for 21st century skills. *21st century skills: Rethinking how students learn*, 20, 51-76.
- Johnson, J. 2003. Children, robotics and education. In *Proceedings of 7th international symposium on artificial life and robotics* (Vol. 7, pp. 16-21), Oita, Japan.
- Han, S. & Bhattacharya, K., 2001. *Constructionism, Learning by Design, and Project Based Learning*. In *Emerging perspectives on learning, teaching, and technology*.
- Hoyles, C., 1985. What is the point of group discussion in mathematics? *Educational studies in mathematics*, 16(2), pp.205–214.

Kafai, Y. B., & Burke, Q. (2015). Constructionist Gaming: Understanding the Benefits of Making Games for Learning. *Educational Psychologist*, 50(4), 313–334. <https://doi.org/10.1080/00461520.2015.1124022>

Kafai, Y. B., Burke, Q., & Mote, C. (2012). What makes competitions fun to participate? The role of audience for middle school game designers, *Proceedings of the 11th International Conference on Interaction Design and Children*, 284–287.

Kynigos, Chronis. 1995. Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies. *Computers and Exploratory Learning*, diSessa, A, Hoyles, C. and Noss, R. (eds), Springer Verlag NATO ASI Series, 399-420.

Matarić, M. 2004. Robotics Education for All Ages, *Proceedings, AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, Palo Alto, CA, Mar 22-24.

Mead, R.A., Thomas, S.L., and Weinberg, J.B. 2012. From Grade School to Grad School: An Integrated STEM Pipeline Model through Robotics in Robots in K-12 Education: A New Technology for Learning, IGI Global.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic books

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, 39(3.4), 720-729.

Utting, I., Cooper, S., Kölling, M., Maloney, J., and Resnick, M. (2010) Alice, Greenfoot, and Scratch: A Discussion. *ACM Transactions on Computing Education*, Vol. 10, No. 4, Article 17, November 2010.

Ethnomathematics in Teacher Education: Analysis and Construction of Geometric Ornaments

Igor Verner, ttrigor@technion.ac.il

Technion – Israel Institute of Technology, Israel

Khayriah Massarwe, massarwe@technion.ac.il

Technion – Israel Institute of Technology, The Arab Academic College for Education, Israel

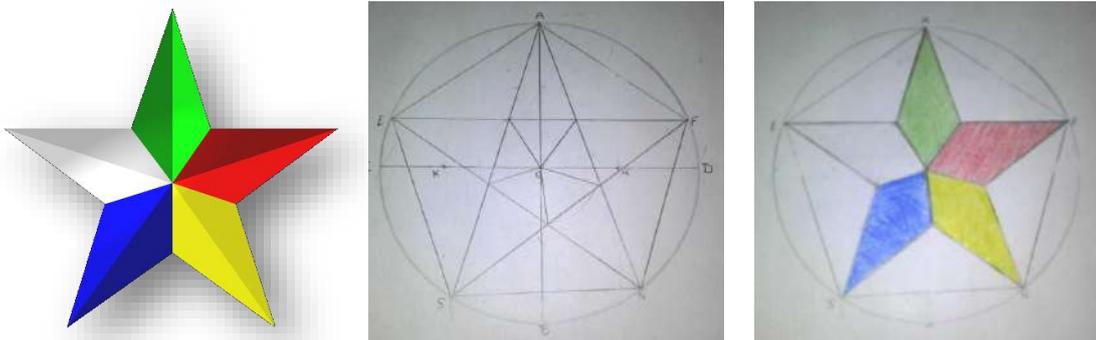
Daoud Bshouty, daoud@technion.ac.il

Technion – Israel Institute of Technology, Israel

Abstract

This paper presents a study in which we developed, implemented, and evaluated a teacher education course for teaching geometry based on the ethnomathematics approach. The participants, prospective and in-service teachers with different cultural backgrounds studied geometry through analysis and construction of geometric ornaments from diverse cultures, and acquired knowledge and skills in multicultural education. The students inquired geometrical properties and symbolism of the ornaments from the cultures they chose. In the study we analysed motivating desires observed in the course using the theory of engagement structures proposed by Goldin and colleagues. We found that some of the engagement structures that are typical for conventional mathematics classes, emerged in our course while other structures were not observed. We propose a new, additional engagement structure to embody motivational desires arising from multicultural interactions in diverse classes.

The figure below presents the Druze Star, the prominent symbol of the Druze culture. It also shows the geometric construction of the Star performed by a Druze student. She figured out the meaning of colours of the Star: green symbolizes nature; red symbolizes courage and love; yellow symbolizes enlightenment; blue indicates patience and brotherhood; white indicates reconciliation and peace.



Keywords

ethnomathematics; geometric ornaments; teacher education; learning engagement.

Introduction

Creative hands-on practices and projects are increasingly included in the school mathematics curricula. In these activities students learn mathematics with applications through practical activities in technological environments. They construct artefacts that drive their thinking, inquiry, and communication. Teachers should have knowledge of the constructionist approach and skills to apply it in choosing suitable learning environments, activities, and strategies of students' guidance.

The challenges facing the mathematics teacher in implementing the constructionist approach are to engage every student in the class, to make the activities accessible for the students, and provide them opportunity for systematic mathematical learning. To meet these challenges, research points out to the important role of affective factors on the constructionist learning behaviour (Picard et al., 2004).

Geometry underlies construction of artefacts. As such, development of geometry is rooted in the constructional practice while learning geometry is naturally embedded in the constructionist educational processes. Since ancient times, geometric constructions, in addition to functionality, possessed symbolism and cultural values. This duality stands out in geometric ornaments, decorative and symbolic patterns of cultural value, composed of basic geometric figures that are repeated under different transformations. Ornaments are complex geometric objects that carry within a variety of properties of Euclidean geometry. They are part of the world cultural heritage, created in folk art and crafts throughout the history of nations (El-Said, 1993). Historically, people constructed ornaments with compass and straightedge to express their feelings, spiritual beliefs, and cultural identity.

Our aspiration for teaching geometry with ornaments is to make geometry lovable to students who perceived it as a dry subject, as expressed by Papert (2006) "Let's stop trying to make children like the mathematics they hate. Let's make a mathematics they will love." For the past ten years, we developed and guided practices of analysis and construction of geometric ornaments in mathematics teacher education. The two theories that underlie our teaching were constructionism and ethnomathematics.

Ethnomathematics approach

D'Ambrosio (2004) defines ethnomathematics as the mathematics practiced among identifiable cultural groups, meaning that ethnomathematics deals with mathematical conceptions and techniques developed in different cultures to solve real-life problems. In education, the ethnomathematical approach generates learning processes, in which learners inquire into mathematical experiences from their own and other cultures to understand how mathematical ideas are formulated and applied (Rosa, & Shirley, 2016).

The ethnomathematical approach promotes mathematical learning by activating the factors of students' engagement related to context, culture, and ethnicity. Ethnomathematically based courses introduce mathematical concepts in such a way that they are better understood, and their power, beauty and utility are better appreciated (Achor, 2009). Affective processes in ethnomathematics courses are influenced by both cultural context and diversity. The majority of ethnomathematically based courses in schools and teacher colleges consider situations in which students in culturally homogeneous groups study mathematics in the context of their culture (Gerdes, 2001; Rosa & Orey, 2011).

Presmeg (1998) studies a different situation in which pre-service teachers with different cultural backgrounds inquired into mathematics embedded in their cultures and shared findings with the class. Her study shows that using students' cultural practices in the mathematics classroom had a positive impact on their awareness of diversity and multiculturalism. Presmeg calls for seeing cultural diversity as an asset and not a burden. Gay (2010) emphasizes the need to enhance teacher education through special training to develop the skills of teaching in a culturally diverse classroom and creating cross-cultural dialog. The teacher can reinforce students' energy and learning engagement by encouraging classroom discussions on interesting and joyful experiences and by asking questions that help maintain students' positive mood (Gay, 2002; Vomvori-Ivanovic, 2012).

Our research started with pilot experiments, in which high school students in geometry lessons performed a constructionist exercise involving analysis and construction of geometrical ornaments (Massarwe, Verner, & Bshouty, 2010). The experiments revealed that the students and teachers perceived learning geometry through ornaments as a meaningful and joyful experience. Then we took part in the program of the Israel National Council of Higher Education for development of academic courses on the theme "academy–community partnership for social change." Our Technion course "Issues in ethnomathematics" attracted interest of prospective and in-service teachers of mathematics and other subjects. It prompted us to study culturally responsive teaching of geometry and the ways to

treat issues of diversity and cultural identity. In this way we came to an understanding of the educational value of ethnomathematics and its potential for promoting constructionist mathematical learning.

Our study explored learning engagement of students participated in the ethnomathematically based teacher education course. The study applied the methodology engagement structures proposed by Goldin et al. (2011). The theory focuses on identifying typical patterns of engagement observed in the learning process and characterizing the patterns by means of engagement structures. An engagement structure is considered as a construct including the statement of the motivating desire, the scheme of the social interaction, the characteristics of situations which are likely to evoke the desire, and the behavior pattern. Goldin et al. (2011) presented patterns of learning engagement typical for conventional middle school mathematics classes and characterized them by a number of engagement structures. The study presented in this paper identified and characterized the patterns of engagement that are typical for our course.

Characteristics of learning engagement

This case study followed up the course delivered at the Technion Department of Education in Science and Technology. The goal was to investigate motivations that drive students to perform mathematical, cultural and pedagogical activities in our ethnomathematically-based teacher training course. We sought to answer the question: what are the features of student engagement in the course and how can the methodology of engagement structures be applied to analyze these features?

The course "Issues in ethnomathematics" has been developed and given to prospective and in-service teachers as part of the Technion teacher education program. The 29 students participated in the course had different academic and cultural backgrounds. The 42-hour course included the following learning activities:

- Acquisition of geometrical skills required for construction and analysis of ornaments. The students learned basic geometric constructions with compass and straightedge and applied this knowledge to analyze and construct given ornaments. They learned to define geometrical problems related to the constructed ornaments.
- Studying mathematical and pedagogical concepts for teaching geometry in cultural context. The students learned the principles of teaching mathematics with applications and cognitive mechanisms of learning in context. They inquired how geometric concepts are embedded in culturally meaningful ornaments and serve to express symbolism and beauty. They discussed the concept of diversity and the value of multicultural education.
- Development of instructional units on analysis and construction of geometric ornaments. The lectures exposed students to the common historical, mathematical, and cultural roots of geometric ornaments. They discussed applicability of the ethnomathematics approach in class. Each student developed and gave in class a presentation of the selected culture, which included a collection of ornaments, a detailed construction procedure of the chosen ornament, related geometric problems, and their solutions.
- Workshop "Joyful learning of geometry in multicultural context". The workshop "Joyful learning of geometry in multicultural context" was the culmination of the course. Each of the students guided a multicultural group of school students to make a poster. Each poster consisted of an ornament constructed by compass and straightedge, its variation made using the graphics software tool Kaledomania, geometric problems and solutions, and an essay on the chosen culture. The posters were publicly exhibited at the end of the workshop.
- Analysis and presentation of the workshop results, and a final report. In the last part of the course, the students analyzed the data collected during the workshop, and presented findings.

When teaching the course we paid attention to the strong engagement of students in the ethnomathematical learning activities and decided to investigate motivating desires that drive the engagement. We applied the methodology of engagement structures proposed by Goldin et al. (2011).

To answer the research question, observations of classroom activities were recorded and videotaped throughout the course. We collected and analyzed repeated patterns of engagement that indicated the

presence of the engagement structures proposed by Goldin et al. and of new structures that emerged through our teaching. We also analyzed the homework tasks and materials prepared by the students for the workshop, as well as in-class discussions, final reports, and post-course reflections. In these reflections, we paid attention to patterns associated with cultural awareness of students with different cultural backgrounds. We found that studying geometry in cultural context touched student's feelings of cultural identity and raised their desires for learning. This way we came to a new engagement structure that we named "Acknowledge my culture."

Below we present structures from the ones proposed by Goldin et al. (2011) that appeared in our course.

Get the Job Done. In this engagement structure "emotional satisfaction accompanies fulfilling the obligation through task completion" (ibid.). Our observations in the course showed that this engagement structure arises in the constructionist learning process. This feature was especially prominent when the students learned to construct ornaments and prove their geometric properties. In the beginning, the students had difficulties in constructing the ornament using compass and straightedge because they lacked this skill. They asked for guidance to construct the first ornament step by step and they showed commitment to accomplish the task correctly and accurately (Figure 1). The post-course reflections indicated that students' desire to "get the job done" in the task, was determined by the added value of the experience for developing construction skills and understanding geometry of ornaments.

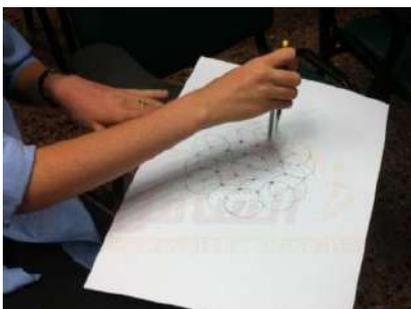


Figure 1. Student's commitment to accomplish the construction task.

Look How Smart I Am. "The student's motivating desire behind this engagement structure is to impress classmates and achieve positive self-regard by demonstrating high mathematical ability" (ibid.). We found indications of this desire in our students when they strived to perform the course assignments in the best way possible and to demonstrate their competences in geometry and culturally responsive teaching. The desire for superior achievement was indicated also by creative solutions developed in the course.



Figure 2. Students expressing positive self-regard.

Check this Out. For this engagement structure the source of motivating desire is "that solving a mathematical problem can give a benefit" (ibid.). This desire can be stimulated by extrinsic benefits, but our course used intrinsic payoffs: (a) feeling of invention aroused when the students practiced in posing

and solving non-conventional geometric problems; (b) satisfaction about coping with a real geometric problem of practical value; and (c) discovery of geometric properties, methods of creation of ornaments, and cultural meanings. An example of a geometric problem proposed by one of the students is presented below.

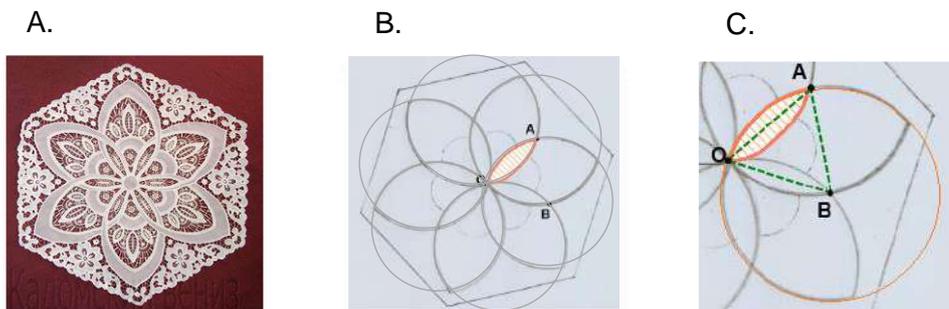


Figure 3. A. An ornament; B. problem related to the.

Problem: The ornament in Figure 3A is constructed as shown in Figures 3B and 3C. Given that the radius of the circles is R , find the area of the selected shape.

Solution: We draw a circle of radius R with the centre B . In triangle $\triangle OAB$, $\angle AOB = \frac{360^\circ}{6} = 60^\circ$

from the construction and $|BO| = |OA| = R$. So, $\triangle OAB$ is an isosceles triangle and its area equals

$S_{\triangle OAB} = \frac{\sqrt{3}}{4} R^2$. The area of the sector AOB is: $S_{AOB} = \frac{60^\circ}{360^\circ} \pi R^2 = \frac{1}{6} \pi R^2$. Then the area of the selected shape is $S = 2 \cdot (S_{AOB} - S_{\triangle OAB}) = 2 \cdot \left(\frac{\pi}{6} R^2 - \frac{\sqrt{3}}{4} R^2 \right) = \left(\frac{\pi}{3} - \frac{\sqrt{3}}{2} \right) R^2$.

I'm Really into This. The meaning of the structure is that “the student is intrigued by solving a challenging mathematical problem and tunes out from other elements of the environment (ibid.)”. Our course involved students in the geometric analysis of ornaments, they proposed problems related on the ornament and solved the applied geometric problems with enthusiasm. Being intrigued by the geometric problem, the students did not abstract from the cultural meaning of the ornament and from the way of its construction.

Let Me Teach You. This structure presents “the motivation desire of a student to mentor the classmate who has difficulties in solving mathematical problems. (ibid.)”. This role motivates the student to learn and master the subject. Satisfaction from the progress of the classmate is an additional stimulator of the motivation desire. In our course each student developed an instructional unit on geometric ornaments from the culture that she/he selected. The student taught the unit to peers with the intention of getting feedback for improving the presentation. The course created an atmosphere in which the students felt invited to share information on heritage artifacts, proposals for geometric constructions and proofs, and ideas on how to present the material to school students.

From our observations, in peer teaching the students shared proposals of alternative solutions for geometric constructions and proofs, pedagogical ideas on how to present the material to school students, and findings on traditional customs and visual artefacts in different cultures. A repeated reflection:

The discussions on geometry greatly helped me to understand the way other people looked at the same subjects and even taught me some new tricks.

Observations in the course indicated occurrence of motivating desires rooted in cultural diversity of the participants and pointed out the need to introduce a new engagement structure which is presented below.

Acknowledge My Culture. Here, "the motivating desire awakes when the student feels urged to be a representative of his or her own culture" (Goldin et al., 2011). In our course some of the students elected to investigate ornaments from their own culture and others from cultures to which their families belonged before immigration to Israel. Passion, inflamed by exposure to artefacts from different cultures, drove the students to indulge in collections of artefacts, and impress the class by distinguished mathematical features of artefacts from these cultures. A repeated reflection:

In the course I learned about my culture in a more elaborate way and tried to represent it in the best way possible.

Discussion and Conclusion

Today's students prefer hands-on, visual, joyful, and socially meaningful learning. To meet their preferences, teachers of mathematics should be able to effectively use the learning resources of context, affect, culture, and ethnicity. Therefore, teaching in a cultural context becomes a necessary subject of mathematics teacher education. Our course addressed this subject with regard to teaching geometry based on the ethnomathematical approach. This approach focuses on learning by inquiry into mathematical practices developed in different cultures to solve real-life problems.

In our course students inquired properties and symbolism of geometric ornaments from different cultures. Following the constructionist pedagogy, the course included practice in constructing ornaments by compass and straightedge, posing and solving geometric problems, creating aesthetic presentations, developing instructional units, and teaching them to diverse groups of school students.

In the course we assessed student's performance in the construction of the ornament of her/his choice using compass and straightedge and the analysis of the construction procedure. Through the construction process the students gained insight into geometric analysis that justifies the practice. In the assignment of posing and solving geometric problems, we assessed the ability to identify and formulate an interesting problem related to the complex geometric pattern, solve it formally, and present the solution. Assessment of the instructional units on geometric ornaments and the feedback from the peers and the school students indicated that the prospective teachers developed the skill of teaching geometry in cultural context to diverse groups. The experiences were perceived by the prospective teachers as interesting, joyful, and engaging.

Our research focused on students' engagement in the course. We applied the methodology of engagement structures proposed by Goldin et al. (2011). The analysis of students' motivational desires observed in our course revealed their significant differences from that typically observed in conventional mathematics classrooms:

- High level of engagement of all the students participated in the course.
- The diminished role of external motivation (rewards and sanctions), and of competition for superiority.
- The increased role of interest, enjoyment, self-realization, and perceived relevance.
- Strong impact of personally meaningful cultural context.

Based on the positive results of this study, we continue to develop and implement the ethnomathematical approach in teacher professional development as a way to facilitate more constructive, contextualized, and culturally responsive mathematics education in schools.

References

- Achor, E. E., Imoko, B. I. & Uloko, S. E. (2009). Effect of ethnomathematics teaching approach on senior secondary students' achievement and retention in Locus. *Educational Research and Review*, 4(8), p. 385-390.
- D'Ambrosio, U. (2004) Prefix. In Proceedings: *Ethnomathematics and Mathematics Education*, ICME 10 Discussion Group on Ethnomathematics, Copenhagen (p. V-X).
- El-Said I. (1993) *Islamic art and architecture. The system of geometric design*. Reading, UK: Garnet Publishing.

- Gay, G. (2002). Culturally responsive teaching in special education for ethnically diverse students: Setting the stage. *Qualitative Studies in Education*, 15(6), 613-629.
- Gay, G. (2010). *Culturally Responsive Teaching: Theory, Research, and Practice* (2nd edition). New York: Teachers College Press.
- Gerdes, P. (2001). Ethnomathematics as a new research field, illustrated by studies of mathematical ideas in African history. In J. J. Saldana (Ed.): *Science and Cultural Diversity. Filling a Gap in the History of Science* (p. 11-36). Mexico, Cuadernos de Quipu.
- Goldin, G. A., Epstein, Y. M., Schorr, R. Y. & Warner, L. B. (2011) Beliefs and engagement structures: Behind the affective dimension of mathematical learning. *ZDM*, 43(4), p. 547–560.
- Massarwe, K., Verner, I. & Bshouty, D. (2010) Fostering creativity through geometrical and cultural inquiry into ornaments. In: B. Sriraman and K.H. Lee (Eds.), *The Elements of Creativity and Giftedness in Mathematics*, p. 217-241, Sense Publishers.
- Papert, S. (2006) From math wars to the new new math. Plenary lecture at the 17th ICMI Study conference, Digital Technologies and Mathematics Teaching and Learning: Rethinking the Terrain. Hanoi, Vietnam.
- Picard, R., Papert, S., Bender, W., Blumberg, B., Breazeal, C., Cavallo, D., Machover, T., Resnick, M., Roy, D., & Strohecker, C. (2004) Affective learning – A manifesto. *BT Technology Journal*, 22(4), p. 253–269.
- Presmeg, N. C. (1998) Ethnomathematics in teacher education. *Journal of Mathematics Teacher Education*, 1(3), p. 317-339.
- Rosa, M. & Orey, D. C. (2011) Ethnomathematics: the cultural aspects of mathematics. *Revista Latinoamericana de Etnomatemática*, 4(2), p. 32-54.
- Rosa, M., & Shirley, L. (2016) In Guise of Conclusion. In: M. Rosa et al. (Eds.), *Current and Future Perspectives of Ethnomathematics as a Program*. ICME-13 Topical Surveys, pp. 39-40, Hamburg, Germany: SpringerOpen.
- Vomvoridi-Ivanovic, E. (2012) Using culture as a resource in mathematics: The case of four Mexican–American prospective teachers in a bilingual after-school program. *Journal of Mathematics Teacher Education*, 15(1), p. 53–66.

Coding to Learn – Informatics in Science Education

Michael Weigend, mw@creative-informatics.de
Holzkamp-Gesamtschule Witten, Germany

Abstract

Programming is a way to explore and elaborate scientific content. Software projects might (to some extent) replace traditional pencil-and-paper textbook problems that require algebra skills. This contribution discusses a Python workshop which has been conducted with 34 students from three German high school chemistry classes in grades 11 and 12. The participants (mostly programming novices) developed interactive programs solving quantitative Chemistry problems. They claimed to enjoy programming, the girls almost as much as the boys. A majority agreed that programming implies practising logical thinking, precise communication and creativity as well as elaborating chemistry knowledge.

Keywords

science education; programming; computational thinking

Programming and Science Education

Computational Thinking (CT) is the thought processes involved in developing computer programs. However, CT is not just relevant for professional software engineers. Like reading, writing and calculating it is considered to be a “universally applicable attitude and skill set” that everybody should learn at school (Wing 2006 p. 32). This includes abstraction, decomposition, algorithmic thinking, evaluation and generalization (Selby and Woolard 2013). CT concepts are already (silently) used in science and science education. For example, the concept of generalization is used in chemistry, when students find principles and rules for chemical reactions based on individual observations. Chemical formulas like H_2O are abstract models of real molecules. There are several suggestions how to integrate computational thinking into curricula. One of the goals of the Google “Online course on CT for educators” (<https://computationalthinkingcourse.withgoogle.com>) is to “increase the awareness of CT concepts among educators”. Sometimes elements of traditional science curricula are reinterpreted as related to CT. On the other hand, coding (an obvious application of CT) can be a way to explore and elaborate scientific content. Before discussing examples of science-related programming projects let me briefly point out a few facets of traditional learning units in textbooks. Beside general information about a topic, STEM textbooks contain elements that are meant to evoke active elaboration of content: worked out examples, exercises (tasks) and solutions to tasks. Worked out examples have proved to be an efficient way to learn (Stark et al. 2002). There are different patterns of elaboration, which correspond to different levels of mental effort. Students may for example reconstruct an example in depth and anticipate the solution by themselves or they just take the (metacognitive) information about a possible structure of a solution.

When the learning objects contain quantitative aspects, exercises usually imply algebraic transformations and calculations. A typical task from a popular German chemistry text book is this:

Calculate the pH of these solutions:

- Sodium hydroxide solution, $c_0(\text{NaOH}) = 1 \text{ mol/L}$
 - Ammonia solution, $c_0(\text{NH}_3) = 0.2 \text{ mol/L}$
- (Tausch& von Wachtendonk, 2015, p.52)

The solution of this task requires these activities:

- Find relevant formulas like $\text{pH} + \text{pOH} = 14$.
- Do algebraic transformations and solve equations.
- Look up chemical constants (like $\text{pK}_B(\text{NH}_3)$) in a table.

- Use correct wording, in particular write correct units like mol/L for concentrations.

The primary goal of these exercises (as part of chemistry education) is not to practise algebra and to solve these problems quickly and error-free but to elaborate chemistry concepts like acids, bases and chemical equilibrium. In Germany, high school students usually do not have to learn formulas by heart. But they are expected to develop the competence of using given formulas to solve problems. For this they need a general understanding of relevant chemistry facts and a more general competence of mathematical modelling and problem solving.

In contrast to conventional pencil and paper textbook exercises, a programming task requires to develop a digital artefact, that is useful for other people. That implies almost the same activities as during solving a textbook problem, but they are now part of a design process. The programmer must create verbal system responses that are precise and well understandable and uses scientific vocabular and units because they are essential, when asking for input data. Example "Input the molar weight M of the solved substance (g/mol)."

An Introductory Python Programming Workshop for Chemistry Classes

In winter 2017/2018, students from chemistry classes in grades 11 and 12 at a German high school participated at a 90-min programming workshop. Among the 34 students whose data were evaluated, there were 21 girls and 11 boys (2 did not tell the gender), the average age was 17.5 years. Most of the participants (73%) had never written a program before. They got a short introduction and then worked on their own, following a written tutorial. They were free to collaborate and ask questions. In contrast to a regular computer science class, the primary goal of this workshop was not to discover programming techniques as such but to get an idea about how to *use* programming as a tool to solve Chemistry-related problems.

Thus, the 6-page tutorial was designed to be self-explaining for absolute beginners. It consisted of instructions, Python listings, visual and verbal explanations and tasks. Except for the very first task, all examples and tasks were related to chemistry. At some points the students were asked to note the time. So it was possible to calculate the time they needed for the exercises.

Interactive Exercises

The tutorial starts with three exercises which must be performed in the interactive Python Shell (Idle). The Python Shell displays a command line starting with the prompt `>>>`. When a user enters a Python statement and hits the ENTER-key, the Python runtime system interprets the statement (for example a mathematical expression) and returns the result in the following lines (read-eval-print loop, REPL). Example:

```
>>> 23*5.2
119.60000000000001
```

The first task is to try out four expression commands and discover the little differences between Python syntax and mathematical notation. In the workshops, this took two minutes in average.

In the second exercise the students use the Python Shell as calculator for three little computational problems from a typical chemistry textbook. Example: Calculate the mass (in g) of 0.2 mol sodium chloride (NaCl). For this they needed seven minutes.

In the third activity the students are asked to follow step-by step instructions to solve a more complex problem from Chemistry using the Python Shell. In this context the tutorial introduces variables to name data and split a complex calculation into separate steps (CT concept: decomposition).

Task: Through a hole in a pipe, the amount of 100 g methane went into a cylindric building (height 20 m, radius 5 m). Calculate the concentration in mol/L.

Step 1: Calculate the volume of the cylinder (m^3).

```
>>> V = 20 * 3.14 * 5**2
```

The result of the term is assigned to the variable V. One can check the value of V by calling the `print()` function:

```
>>>print(V)
1570.0
```

Step 2: Calculate the number of mol:

```
>>> n = 100/16
```

Step 3: Calculate the concentration c in mol/m³:

```
>>> c = n/V
>>>print(c)
0.003980891719745223
```

And so on.

In cartoon-like explanations, variables are visualised as containers for data. The name of the variable in this metaphor is a label, making it possible to *refer* to the container.

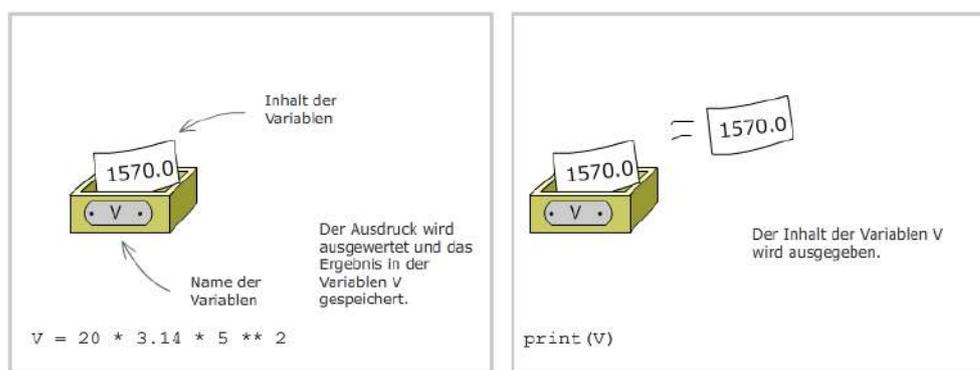


Figure 1. Two frames from a cartoon-like explanation of Python statements.

Interactive Programs

In the next exercise the students write a script consisting of six lines of code using the Idle editor. They have to figure out how to use the editor and manage to write error-free program text. In the workshops this took in average 16 minutes.

The interactive program has a simple Input-Processing-Output design. The user is asked to input the number of carbon and hydrogen atoms. The program calculates the molar weight of a hydrocarbon compound consisting of these atoms and returns the value using proper units (g/mol).

```
print("Calculate the molar weight of a hydrocarbon compound.")
inp = input("Number of C atoms: ")
n_C = int(inp)
n_H = int(input("Number of H atoms: "))
M = n_C*12 + n_H*1
print("Molar weight of the compound: ", M, "g/mol.")
```

The execution of the program is explained in a cartoon with five frames in the style of figure 1. The average self-rated comprehension of the program was 75%. Note that understanding the calculation in line 5 requires Chemistry knowledge.

The students are then asked to extend this program and add a new feature on their own. The new version should now be able to calculate the molar weight of organic compounds additionally containing oxygen atoms. This program should be tested with methanol (CH₃OH). 90% of the participants were able to do this and these students needed in average eight minutes to finish the extension.

The final challenge is to implement an interactive program that helps preparing solutions. The students have to transfer the techniques they learned from the prototype to create their own project.

Evaluation

Table 1 displays the results of the evaluation. The participants claimed to enjoy programming, the girls almost as much as the boys. The average agreement to “Programming is fun” was 4.55/5 for boys and 4.05/5 for girls. A majority said that programming implies practising logical thinking, precise communication and creativity as well as practising chemistry knowledge. There were only a few critics. Four of those seven students who stated that they had lost time doing unimportant things had the opinion that programming is useless knowledge anyway.

Table 1. Evaluation of the programming workshop (n=34).

Statement	Average level of agreement (1 to 5)	Complete and partly agreement
Programming was fun.	4.2	85%
While programming I have practised logical thinking.	4.2	79%
During programming I have practised chemical knowledge.	3.9	74%
I could imagine to do more programming projects.	3.7	62%
While programming I have practised expressing things precisely.	3.6	59%
While developing the programs I could be creative.	3.5	56%
For my projects I did some research on chemistry knowledge.	2.9	47%
While programming I lost time doing unimportant things.	2.4	21%
To me programming is useless additional knowledge.	2.3	18%

Starter Projects for Science Education: Reconstruct-Improve-Create

The Python workshop, which was discussed in the previous section follows a “Reconstruct-Improve-Create” design pattern. This can be considered as a simple version of agile programming (Beck 1999). “Stories” are implemented in a few short iterations. The tutorial contains these elements:

Reconstruct

- A story: Visual and verbal description of a useful software covering a certain topic in chemistry (e.g. molar weight),
- An easy-to-understand prototype Python program implementing the story (starter project),
- Visual and verbal explanations helping to understand the program code.

Improve

- Suggestions, how to improve the prototype program.

Create

- A new story that can be implemented using the techniques introduced in the prototype.

Note that the students are supposed to use the editor, to type a few lines of code themselves and to understand and modify the program (reconstruction). This contrasts for example with the Google “Exploring Computational Thinking” approach, which is based on copy and paste rather than on reconstructing. In the example simulating radioactive decay, learners are supposed to copy and paste

more than 40 lines of rather complex code in a VPython environment and then run the program (<https://docs.google.com/document/d/1HotPQxfK1w1SytPtJg5qqKkxFrqmZpiZrxyaIHRKZIE/edit>).

Programming Scientific Experiments

Taking advantage of the CT concepts “abstraction” and “generalization”, it is possible to create relevant software without specific programming experience. This section discusses example projects using a Raspberry Pi and a non-dispersive infrared (NDIR) CO₂ sensor. These projects are part of a workshop on sensor technology for chemistry teachers in North Rhine-Westphalia, Germany. The NDIR consists of an infrared source, a filter and a photo sensor in a tube with some holes. The IR radiation travels through the tube and is partly absorbed by the carbon dioxide molecules in the tube. A fraction of the IR with a certain wavelength passes the filter and is detected by the photosensor. A small digital device processes the detector output and creates data that are sent to the Raspberry Pi via the serial I²C bus.

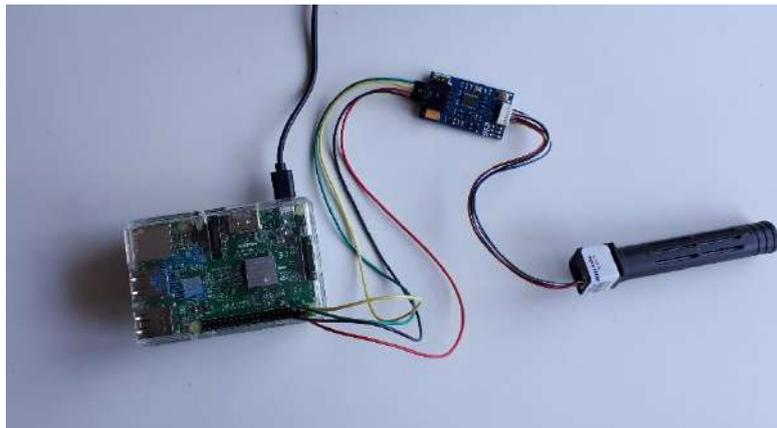


Figure 2. Raspberry Pi with NDIR CO₂-sensor module MH-Z16

Story 1: The carbon dioxide concentration in a room is an important factor of wellbeing. Connect the sensor to the GPIO of a Raspberry Pi with jumper wires and observe the carbon dioxide concentration in the room (ppm) for five seconds. The output on screen might look like this:

```
540 ppm
540 ppm
543 ppm
545 ppm
550 ppm
```

The hardware is prepared in a few minutes. This already gives an idea how the serial data transfer works. The digital sensor device has four pins: VCC, GND, CLK and SDA, which must be connected via jumper wires to the GPIO of the Raspberry Pi. The pins VCC (+5 Volts), GND (ground) are for the power supply. The data are transferred bitwise via the SDA line. The CLK line is for a clock signal that synchronizes the data transfer.

The following Python program implements the story and illustrates the CT concepts *abstraction* and *generalization*.

```
from ndir import get_ppm      #1
from time import sleep      #2
for i in range(5):          #3
    c = get_ppm()           #4
    print(c, "ppm")         #5
    sleep(1)                #6
```

All text elements at the end of a line starting with # are comments and are not part of the formal text.

#1: Import the function `get_ppm()` from the module `ndir`.

#2: Import the function `sleep()` from the standard module `time`.

#3: Repeat the following indented block 5 times.

#4: Get the current CO₂-concentration in ppm and store the value in the variable `c`.

#5: Print the value of `c` in the current line of the output window. Then start a new line.

#6: Wait for one second and continue then.

Programming implies abstraction. A computer program can be considered as a simplified model of reality. Unnecessary details are ignored. In this case, measuring the CO₂-concentration is reduced to a function call in line #4. In industrial software development, abstraction takes place on many levels - from system design to coding. Abstraction is considered as a key competence for programmers (Kramer 2007). But even a tiny project like this example illustrates the benefits of abstraction by using a formal language. From the perspective of working memory theory (Baddely 2013, Dehn 2008) abstraction is essential for problem solving. Humans can only keep a very limited number of chunks of information in mind while thinking about a solution. Unnecessary details would be disturbing. Therefore, for a human it is easier to comprehend the semantics of a concise program, than the semantics of an algorithm written informally in natural language. However, the prerequisite for understanding is that she or he masters the programming language.

Thus, all language elements in a Python starter project must be explained well to the novice learner. Some Python commands in the example can be understood almost at once, since they use familiar concepts metaphorically. The function call `sleep(1)` makes the Python interpreter to stop the execution of the program and to continue it after one second. The concept of falling asleep and being awaked after a while is well known from everyday-life.

Some other language elements are more complex and might need some active exploration and rehearsal before they are fully understood. Learners might want to try out some calls of the functions `range()` and `print()` in the Python Shell. Generalisation means solving new problems based on already-solved problems. This includes reusing software (for example classes or functions) in new contexts. In this example, the import of Python modules demonstrates the CT concept of generalisation. The module `ndir` encapsulates all the details of reading sensor values vis the I²C bus. It was designed to be useful in many projects.

However, this is only a rather weak demonstration. Learners get a deeper understanding, when they define functions, classes and modules on their own and use them at different places in their project software.

Automatic Experiments With Human Actors

Measuring the CO₂ concentration may be a part of a scientific experiment about diffusion. In special environments like space satellites, experiments may be conducted completely automatically. In the class room it is possible to involve humans as actors. The program tells the actor what to do (instead of controlling motors directly) and calculates useful information from sensor data. This way the program represents a precise description of the experiment. The program text is rather simple and concise, but one can imagine that with some effort it is possible to automatize the experiment completely.

Story 2: Create an interactive program that helps to conduct an experiment that demonstrates the diffusion of a gas.

The implementation of this story is an extension of the starter project.

```
from ndir import get_ppm
from time import sleep

print("Watch the diffusion of carbon dioxide.")
number = input("Number of measurements: ")
print("Now, breath to the NDIR sensor!")
```

```
for i in range(number):
    c = get_ppm()
    print(c, "ppm" )
    sleep(5)
```

Example Dialog

Watch the diffusion of carbon dioxide!

Number of measurements: 10

Now, breathe to the NDIR sensor!

```
722 ppm
709 ppm
935 ppm
1264 ppm
1181 ppm
1032 ppm
1012 ppm
1018 ppm
960 ppm
883 ppm
```

Summary and Conclusion

Developing interactive programs with Python may be an element of science education at high schools. When students develop interactive assistant programs or semi-automatized experiments with sensor readings, they both elaborate scientific content and apply computational thinking. During the process they are challenged to

- design man-machine dialogs using appropriate scientific terms and units
- create and explicate precisely activities that are necessary to conduct an experiment
- apply scientific laws for calculating useful information from sensor reading.

The results of a study with 34 high school students from chemistry classes suggest that young people find introductory programming projects in chemistry lessons instructive and stimulating. Using Python as programming language, even novices can write relevant programs without previous training within a regular lesson. They may get an idea of some CT concepts. But there are limitations and several reasons, why computational thinking is developed best in dedicated computer science classes:

- To experience the benefits of decomposition and generalization, students need to *create* rather complex software projects. Object oriented modelling by creating class structures requires a lot of specific instruction and cannot be learned “on the fly” in chemistry classes.
- To deal with code that is more than a few lines, learners must know debugging techniques. These include special technical knowledge about the IDE, debugging tools and defensive programming. All this is not related to the core topics of traditional science.
- Beginners need a lot of time and some assistance, when they search for errors. Science teachers (who are not programmers) may not be able to help.
- Debugging – which is a necessary part of a complex project – is not very helpful for learning science content. Debugging and testing are genuine software engineering activities. The goal is to increase the technical quality of a software product. Science students might consider the need of coping with complexity as an annoying distraction.

Programming projects in pure science classes need to be simple. But they can still be relevant. The pedagogical potential includes three facets:

- Learning activities may get more interesting and more rewarding, since the students create useful and relevant products.
- Learning activities get more challenging, since the software design requires broad knowledge about a field.

- Learning can be more individual. In an iterative process, each student can individually decide how far she or he wants to develop the project.

References

Baddeley, A. (2003): Working Memory Looking Back and Looking Forward. *Nature Reviews Neuroscience*, Vol. 4, p. 829–839.

Beck, K. (1999) *Extreme Programming Explained*. Addison Wesley.

Beheshti, E., Weintrop, D., Swanson, H., Orton, K., Horn, M., Jona, K., ... & Wilensky, U. (2017, April). Computational Thinking in Practice: How STEM Professionals Use CT in Their Work. In American Education Research Association Annual Meeting 2017.

Dehn, M. J. (2008): *Working Memory and Academic Learning*. John Wiley & Sons, Hoboken, New Jersey.

Google (2018): *Computational Thinking for Educators*.

<https://computationalthinkingcourse.withgoogle.com/course> .

Kramer, J. (2007). Is Abstraction the key to computing? In *Communications of the ACM*, 50(4), 37.

Swaid, S. I. (2015). Bringing computational thinking to STEM education. *Procedia Manufacturing*, 3, 3657-3662.

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf .

Stark, R., Mandl, H., Gruber, H., & Renkl, A. (2002). Conditions and effects of example elaboration. *Learning and Instruction*, 12(1), p. 39-60.

Tausch, M. & von Wachtendonk, M. (2012) *Chemie 2000+. Qualifikationsphase*. Bamberg, Germany.

Media Parkour– Experiential Learning Activities for Media Education

Michael Weigend, *mw@creative-informatics.de*

Holzcamp-Gesamtschule Witten, Germany

Fenja Göcking, *fenja.goecking@gmx.de*

Alexander Knuth, *knuthalex@web.de*

Patrick Pais Pereira, *patrick-pais-pereira@hotmail.com*

Laura Schmidt, *laurakathleen2305@gmail.com*

Abstract

This contribution presents a sequence of quick and easy experiential learning activities for sixth graders, which were developed, performed and evaluated by a high school pedagogy class. The activities mainly take place outside in the schoolyard and challenge creativity, computational thinking and collaboration. The learning arrangements are inspired by ideas of experiential learning and cover topics from media education like appropriate reactions to cyber bullying, data transmission between mobile phones, recognizing commercial advertisements, falsification of news etc. The Media Parkour was conducted with almost fifty sixth-graders in the age of 11 to 12. They performed the activities in teams of four and then rated them in a questionnaire.

Keywords

media education; computational thinking; gamification; experiential learning

Introduction

In most German schools, media education is not a dedicated subject. But since media are omnipresent, media education is included in the curricula of several school subjects. Obviously, in language lessons students learn to distinguish between different text types, while in art classes they create and interpret visual media. In science digital models or simulations are used to visualize all kinds of processes. Computer science education takes place, if digital technology is not just used as a tool but underlying principles and ideas are made aware and discussed. This contribution presents a 90-minutes workshop called "Media Parkour" for sixth graders, which has been created by a high school pedagogy class (grade 12). It consists of eight experiential activities with the primary goal to develop the competence of using digital technology in a responsible way. Additionally, computational thinking is challenged. The design of the activities is based on the "NRW-Medienpass" and inspired by ideas of experiential learning.

"NRW-Medienpass" and Digital Competencies

Germany is a federal republic and each federal state has its own regulations on media education at schools. In North Rhine-Westphalia (NRW), the government has published a collection of media competencies – the "NRW Medienpass" (media passport) – the schools are encouraged to consider in their curricula. The primary goal is to help young people and children to use media in a safe, creative and responsible way. The "NRW Medienpass" consists of six groups of competencies for different age groups from Kindergarten to grade 13. These are "operating and applying digital technology", "researching with digital technology", "communication and co-operation", "creating and presenting", "analysing and reflecting" and since recently "problem solving and modelling". The "Media Parkour" has been designed for pupils in grade five and six and focuses on "communication and co-operation with digital technology". The skills include

- using digital tools for communication,

- knowing and following rules for digital communication and co-operation,
- reflecting ethical principles and cultural and social norms when communicating or cooperating via digital media,
- being aware of social, personal and economical risks of technology usage, knowing consequences of cybercrimes and knowing how to deal with them.

Learning Activities for Media Education

During the winter 2017/2018 two sixth grade classes of a comprehensive school in Witten, Germany participated in the "Media Parkour".

The 90-minutes workshop has been designed by high school students from a Pedagogy class and consists of eight learning activities or challenges, which were located at certain places community areas of the school building and on the schoolyard. Each activity was supervised by an adult student from the pedagogy class. The sixth graders went through the parkour in groups of four, solved tasks and received points from the students responsible for the activity. Each group was guided by a pedagogy student. At the end, the points were added up and a prize awaited the winning team.

The activity "What now?" is about correct comprehension of text messages and responsible communication. The team receives cards with WhatsApp messages with different intentions of the sender (insulting a classmate, requesting to spread a chain letters, important information given by the class teacher). The challenge is to recognize the sender's intention and suggest a procedure how to deal with the message.

In the arrangement "How to react best?" two members of the team must improvise a role play on cyber mobbing in a WhatsApp group chat using role cards. The other two are audience. They must recognize the roles and suggest how to react when watching such a dialogue.

The activity "Silent mail" is inspired by the well-known game and illustrates mechanisms of falsifying information that is spread on the Internet. Each member of the team draws a card with instructions. The first person additionally gets a photo depicting a playground scene. The children on the photo are labelled with their names. One can see a harmless scene: "Anna" and "Lisa" are playing on a swing. The first person's instructions say that she or he should describe the situation to the second person (whispering silently into his or her ear) and mention the names of the kids on the photo. The second person's role card says: "Anna once hurt you and you do not like her. When you tell stories about Anna, you usually make her look bad." Person 3 often mixes up names and Person 4 is asked to tell the story aloud, he or she had just heard but to exaggerate. At the end the team gets a list of possible reasons why spread information might get falsified and find the one which was not effective in this case.

In the arrangement "Who is it?" three members of the team mime a bullying situation as living statues. The fourth person is then asked to identify the different roles.

At the activity "The GPS" one member of the team (the "runner") gets a blanket put over the head. Under the blanket the "runner" can read text messages on a smartphone. The "navigator" writes text messages thus guiding the "runner" through a course.

"The Maze" is very similar. A blindfolded person must find a way through a maze while the "navigator" gives directions by tapping on the shoulders.

In "The Advertisement Quiz" the students must identify images from commercial advertisements.

The activity "Data Transmission" simulates a bit-by-bit data transfer from one mobile phone to another.

Three of these learning arrangements will be discussed in depth later.

Pedagogical Background of the Task Design

The *content* of the activities presented in the last section is based on the "NRW-Medienpass". The main goal is to improve responsible media usage. This section discusses some methodological aspects.

All learning arrangements are designed to challenge the children's imaginations and creativity. This is obvious in activities which contain role plays and "living statues". But even if the main task is pure logical reasoning, there is always space for creativity and individual solutions in details.

The activities have the character of experiential learning arrangements (Heckmeir & Michl, 2012). They take place in the outdoors and many of them include physical activity (e.g. "The GPS" and "Data transmission"). All challenges are performed in groups and require cooperation.

The French philosopher Roger Caillois (1958) defines game as a voluntary activity characterized by competition (*agôn*), unpredictability or chance nature (*alea*), adrenaline rushes (*illinx*) and masking (*mimicry*). According to this model, the "Media Parkour" is game-like. First, it is a competition. The teams collect points according to their performance. The activities involving physical activity (like "The GPS") may (occasionally) lead to this emotional excitement that Caillois calls *illinx*. Role plays involve pretending to be someone (or something) else which is a type of *mimicry*. Elements of chance, which make an activity more suspenseful and game-like (*alea*) appear in little details like drawing role-cards.

The book "Computer Science Unplugged" (Bell, Witten, Fellows 2015) has inspired many educators to develop learning activities on computer science without using a computer (e.g. Futschek & Moschitz 2010, Gallenbacher 2006, Weigend 2017). The "Media Parkour" follows the "unplugged" idea, but differs in some points slightly from the classic concept:

- All activities challenge creativity in some way. The participants are supposed to invent (subjectively) new algorithms, visual representations, verbal explanations and ideas. According to the constructionist idea, each activity should lead to some product that is individual and might be unique and surprising. On the other hand, "Computer Science Unplugged" contains a lot of (wonderful) tasks that are pure "brain teasers" and have just one correct solution, which must be found out.
- The "Media Parkour" is not completely "unplugged" since smartphones are involved.
- The activities of CS Unplugged are explicitly related to computer science techniques like data compression, binary numbers, fault tolerance and so on. The activities of the "Media Parkour" are primarily related to *media education*. But many of them are connected to computer science at a second glance. This will be discussed in the next section.

Computational concepts in media-related activities

Jeannette Wing (2006) popularised the term "computational thinking" as an approach to problem solving that is typically adopted by computer scientists. Selby & Woollard (2013) identify five facets of computational thinking: abstraction, algorithmic thinking, deconstruction, generalisation and evaluation. This section discusses three example activities of the "Media Parkour" with respect to their contribution to the development of computational thinking. We speculate about the educational potential; however, we cannot provide an empirical evidence of effectiveness.

Identifying advertisements – algorithmic thinking

The team picks one of three cards with items (soda, chocolate, car). For each of these items there are two photographs somewhere placed at the wall. One photo is from an internet advertisement and the other one is from an informative article just showing the item but not advertising for it. The task is: Go to the photo, which is from an advertisement for the picked item. There are several strategies to identify the correct photo:

- Check whether the item appears in the photo.
- Look for traces of post-processing. In one of the photos letters have been removed. By detecting areas of pixels with exactly the same colour, such manipulations can be identified.
- Check the uniqueness of the item. For example, if the photo depicts bottles of multiple soda brands, it is probably not from an advertisement for one of these brands.
- Check whether this photo is exciting and evokes positive emotions in the viewer.

While discussing how to distinguish ad photos from non-ad photos, the children develop algorithmic ideas. Additionally, the limitations of algorithms may become obvious. For example, humans decide

whether a photo is boring or exciting rather "intuitively" than algorithmically. In fact, modern software for recognising commercials (perceptual ad highlighter) uses artificial intelligence (Hwang 2017).

Data Transmission – Evaluation of Algorithms

At the beginning, the children listen to a story: Tom (sender) wants to send a WhatsApp message to his sister Tina (receiver). The characters of the message are (somehow) represented by zeros and ones (bits) and sent via multiple intermediate stations from the sender to the receiver. This process will be reconstructed in a role play on the schoolyard. The challenge is to transmit a sequence of bits from the sender to the receiver whose location is at least 100 meters away from the sender and around the corner of the school building so that the receiver cannot see the sender. The other two members of the team are intermediate transmitters.

The children are not allowed to talk during the transmission and must represent the bits through body postures. They get three minutes to develop a method for the transmission.

Then they form a chain of transmitting stations, the sender picks a piece of paper with a sequence of zeros and ones and starts the transmission. The receiver at the other end of the chain writes down the received bits. After one minute the transmission stops and the received sequence is compared with the original. The team gets points according to the number of correctly transmitted bits.

In this activity the students need to solve problems which are related to efficiency:

- Zeros and ones must be mimed in a way that is well recognizable from the distance.
- Sequences of identical bits must be handled. Transmitters must be able to recognize, when the transmission of a bit starts and ends. Some teams chose relatively complex movements using the whole body ("disco moves") to represent zeros and ones. This made it easy to recognize individual bits correctly; however, this method was time consuming.
- The speed of the data transmission must be adjusted to a pace, in which each repeater station has enough time to recognize a bit and to send it then to the next station. Usually, each student watched his or her successor and waited until he or she was ready again for the transmission of the next bit.

Of course there was not enough time to develop special data compression techniques to speed up data transfer. But the students were put in a situation where they had to judge algorithms in terms of their effectiveness and efficiency.

The GPS – abstraction and formal language

Each team chooses a "navigator" and a "runner". The runner puts a blanket over the head, which is then attached to his body so that she or he cannot see the surrounding area. But he or she can read text messages on a smartphone. The navigator (advised by the rest of the team) leads the "runner" through a course by writing WhatsApp text messages (commands). Any other form of communication is forbidden, and the runner is not allowed to ask questions.

During the exercise the team encounters the following problems related to controlling by verbal commands:

- The navigator's commands must be clear and unambiguous.
- Writing a command should not take too much time. The message must be short.
- The navigator observes the runner to see if he or she interpreted and executed the latest command correctly. If necessary the navigator can improve the commands.
- It is advantageous to have a limited set of commands that are fully understood.

During the workshop, not a single team had agreed on a set of commands beforehand. The WhatsApp messages were semantically elementary ("right", "left", "straight ahead", "stop") but often decorated – sometimes (especially among boys) with friendly insults ("Go left, you bastard!"). Communication problems were frequently observed. For example, the command "Go to the right!" does not clarify how far the runner should move to the right. Some runners were cautious and had little confidence in the navigator. They made just one single step and then waited for the next command. There were also nasty navigators who occasionally guided their runners against a wall on purpose.

The participants got the opportunity to “play with language” and recognize the benefits of a formal language with well-defined semantics. Short commands involve abstraction since all unnecessary details are omitted.

Another activity of this kind was developed at the Vienna University of Technology. Two players sit back to back and cannot see each other. Both players have the same set of building blocks. Person A builds a figure. Afterwards, person B must reconstruct the figure according to A’s oral instructions. Person B has only three ways to give feedback:

- "Ok" ("I’m ready for the next instruction")
- "Repeat the last instruction"
- "All over again!"

The activity "The GPS" follows the same idea but is much simpler, which makes it suitable for younger students. Additionally it includes physical exercise and can be performed on the schoolyard in the fresh air.

Evaluation

At the end of the workshop, the children filled a questionnaire. The data of 38 persons (20 boys and 18 girls, average age 11.4 years) could be used for evaluation. It turned out that most children loved the "Media Parkour". They were asked to give a grade from 1 to 6 (equivalent to A to F in American schools). The average grade was 1.9. The workshop has been designed as a contest. There is a controversial discussion on the question whether competition has a positive effect on learning (see for example Cantador 2015). In the questionnaire most of the participants claimed that that the competitive character of this workshop was important to them. On a scale from 1 ("super important") to 6 ("totally unimportant") the score reached an average of 2.4. There was only a small difference between girls (2.6) and boys (2.2).

Table 1 shows the results of the evaluation of the individual activities. The students could choose a maximum of three activities for each of the following categories:

- Activities that were the most fun.
- Activities that were the least fun.
- The easiest activities.
- The hardest activities.
- Activities that dealt with important topics.

Table 1. Evaluation of the eight activities

Activity	most fun	least fun	easy	hard	important
What now? (1)	16% (6)	29% (11)	24% (9)	11% (4)	42% (16)
How do I react correctly? (2)	13% (5)	24% (9)	34% (13)	13% (5)	50% (19)
Silent Mail (3)	26% (10)	37% (14)	16% (6)	42% (16)	24% (9)
Who is it? (4)	26% (10)	21% (8)	42% (16)	13% (5)	11% (4)
The GPS (5)	66% (25)	8% (3)	11% (4)	34% (13)	24% (9)
The Maze (6)	55% (21)	0% (0)	18% (7)	24% (9)	13% (5)
The Advertisement Quiz (7)	18% (7)	42% (16)	45% (17)	26% (10)	8% (3)
Data Transmission (8)	61% (23)	13% (5)	34% (13)	8% (3)	24% (9)

The children had most fun with those tasks that involve self-directed activity and body movement. At "The GPS" (66%) and "The Maze" (55%) one person had to go through a path while being "remote-controlled" by a classmate. In "Data Transmission" (61%) digital zeroes and ones were represented by body movements. Other activities were more like a quiz or exam offering the children relatively little opportunity for free action or interpretation. In the exercise "What now?" the children must find out the intentions of a sender of various text messages and assign them to the categories "chain letter", "cyber

bullying" or "informative message". Only 16% of the children rated this activity as the most fun and 29% rated it as one of the least enjoyable exercises. In "The Advertisement Quiz" the children had to distinguish normal photos from advertisements (most fun: 18%, least fun: 42%).

Most participants claimed that "The Advertisement Quiz" was especially easy (45%) while others found it to be difficult (26%). Other tasks that were found to be easy are "Who is it?" (Interpreting a "living statue" and recognizing roles, 42%), "Data Transmission" (34%) and "How do I react correctly?" (34%). On the other hand children found the exercise "Silent Mail" (falsification of news, 42%) particularly difficult.

All activities were on topics that were relevant for media education according to the "NRW Medienpass". But how do the children rate the relevance of the activities? A very important topic seems to be cyberbullying which was addressed in two activities, "How do I react correctly?" (50%) and "What now?" (42%). Only 8% considered the recognition of advertisement photos in "The Advertisement Quiz" as particularly important. The activities "Silent Mail" and "Data Transmission" which are about explaining how certain things work, were marked as relevant by 24% of the participants each.

Media and Computer Science – Educational Perspective

At the beginning of each activity the children got a little introduction, explicating the related technology concepts and media competencies, if they were not obvious: Recognizing commercial advertising is an important media skill. When sending messages, data is transmitted, and so on. For explaining the computer science aspects in more detail, specific technical expertise is required. But the tutors of the small groups were pedagogic students and not computer scientists. An improvement of the setting might be the use of audio-visual material (designed with the support of experts) presented on tablet computers. This could help the students to connect the activities to serious computer science knowledge.

References

- Caillois, R. (1958) *Les jeux et les hommes*. Paris 1958.
- Bell T. ,Witten, I. H., Fellow, M. (2015) *CS Unplugged, Computer Science Without a Computer*. URL: <https://classic.csunplugged.org/books/2015>.
- Boroditsky, L.; Ramscar, M.; Frank, M. C. (2001) The Roles of Body and Mind in Abstract Thought. In *Psychological science*, 13(2), p. 185 – 189.
- Cantador, I. (2015). An example of healthy competition in education. In *Proceedings of the 2nd International Workshop on Gamification in Education (gEducation 2015)*.
- Caillois, R. (1958) *Les jeux et les hommes*. Paris 1958.
- Futschek, G, Moschitz, J.: Developing algorithmic thinking by inventing and playing algorithms. In Proceedings: *Constructionism 2010*.
- Gallenbacher, J. (2006)*Abenteuer Informatik. IT zum Anfassen*. Spektrum Akademischer Verlag
- Heckmair, B., & Michl, W. (2012)*Erleben und lernen*. 7th edition
- Hwang, Y. (2017) The Best Adblocker – Using Artificial Intelligence to Perfectly Block Ads. In *IT for All*, 18. April 2017. URL: <https://www.iotforall.com/best-adblocker-artificial-intelligence-to-perfectly-block-ads/>
- Selby, C., &Woollard, J. (2013) Computational thinking: the developing definition. https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf, 2013.
- Weigend, M. (2017) Smartwalk: computer science on the schoolyard. In Proceedings: *IFIP World Conference on Computers in Education*. Springer, Cham.
- Wing, J. M. (2006) Computational Thinking. In *Communications of the ACM*, 49/3,p. 33–35.

A Creative Learning Sequence in an Introductory Programming MOOC

Elisabeth Wetzinger, *elisabeth.wetzinger@tuwien.ac.at*
Vienna University of Technology, Austria

Gerald Futschek, *gerald.futschek@tuwien.ac.at*
Vienna University of Technology, Austria

Bernhard Standl, *bernhard.standl@ifs.tuwien.ac.at*
Vienna University of Technology, Austria

Abstract

Since January 2017 we have designed and developed a Massive Open Online Course (MOOC) for learning computer programming, addressing high-school students without or with little experience in programming who intend to start studying at a technical University. The objective of this MOOC is to conform the heterogeneous levels of our freshmen students' pre-knowledge in computer programming. We decided to use *Processing*⁷² as programming language because Processing supports an easy learning start to programming by using graphics and animations. We included already at a very early stage of the MOOC a challenging learning sequence that involves creativity, peer feedback and communication with other participants: We asked students to create a graphical artwork through coding using previously in the course presented graphical procedures. Further, we ask them to create a fake copy of an artwork coded by a peer. Finally, artists and fakers should discuss their experiences. This paper summarizes and discusses our experiences with this learning sequence gained during the first run of the MOOC in summer 2017.

Keywords

creative learning sequence; MOOC; learning computer programming

Introduction

In the first-year of the bachelor programs of Computer Science and Business Informatics (CSBI), students have a heterogeneous knowledge and experience levels in coding due to different high-school pre-experiences. This fact challenges teaching and studying at the Faculty of Informatics and lead to a high drop-out rate⁷³. In order to address these issues, our faculty has set measures to improve the bachelor programme curricula as well as teaching practice. The faculty also has established pre-study bridging courses supporting students at the beginning of their studies in September before the actual lectures start by learning basic skills in programming, maths and other topics of Computer Science. The bridging courses are implemented as on-site workshops of which the programming course is limited to 100 participants following a first-come-first-serve rule. This means, that approximately 80% of students accepted for a CSBI bachelor programme are not able to attend the programming bridging course.

As an additional bridging course for prospective students of the Computer Science and Business Informatics (CSBI) and other STEM undergraduate programs, we developed and provided a massive open online course (MOOC) on introductory programming in German language called "Programmieren mit Processing" (engl.: Programming with Processing). We followed the guidelines published by *Ebner et al* (2014). Especially prospective students, who are not able to attend on-site workshops benefit from the online course as it can be accomplished at one's own pace and can be accessed from anywhere only requiring internet access. As it supports massive numbers of participants, literally every prospective or interested student is guaranteed to be able to accomplish the MOOC.

⁷² <http://processing.org> (last access: 06/2018)

⁷³ Source: <http://www.tuwien.ac.at> (Data from our institution)

This MOOC has started in April 2018 at the public MOOC platform iMoox.at⁷⁴ and aims at serving everybody who is interested in learning programming using Processing. Our goal is to provide a course that involves constructive learning sequences that provoke the students' creativity, interest, and passion.

In the next sections, we first present the idea of our MOOC and then the description of the learning sequence in detail. We will close our paper with an overview on students' outcomes and conclusions.

Background

Resnick (2014) claims to give P's a chance: Projects, peers, passion, play. He calls them "The 4 P's of Creative Learning: Projects, Peers, Passion, Play". He argues that active working on meaningful projects supports new ideas in finding solutions. Also, exchange with peers and building on work of others can boost motivation and creative learning processes. Therefore, he also remarks that projects should be rich enough to create passion in finding a good solution. Further, he underlines that a playful way with room for experimentation helps to explore different solutions. Based on that, we are convinced, that learn to code can also be more efficient when these 4 P's of Creative Learning are involved in the instructional design. In order to trigger these so called 4 P's we need not only a suitable programming system but also excellent learning activities. Nevertheless, typical Massive Open Online Courses (MOOCs) rely mostly on passive instructional video learning combined with text-based scripts and multiple choice self-assessment. Koedinger et al (2015) showed that interactive activities support learning more than instructional MOOCs.

For over a decade, computational thinking has been in the focus of educators and researchers in computer science, particularly when it comes to learn how to code. Even though computational thinking has become popular since Wing discussed and reintroduced the topic in 2006 (Wing 2006), computational thinking still has a long tradition in history over decades (Tedre et al 2016). Despite the lack of an overall definition of computational thinking, there is for the most part consensus in including problem-solving involving the thinking skills abstraction, decomposition, algorithmic thinking, evaluation and generalization (Dagiene et al 2017). Hence, learning programming should by its nature support computational thinking skills. At any rate it comprises algorithmic thinking, but while analysing programs and program output also evaluation skills are trained. Since a program is a kind of model of its outcome also abstraction skills may be learned.

Considering both, Resnick's 4P as a framework for building a useful learning environment that conformed to the constructivist approach to coding, and computational thinking as a definition for describing problem-solving in the field of computer science, we developed creative coding tasks as part of our MOOC. In particular, the learning sequence, as presented in this paper allows a playful experimentation while building on another's work and bringing in own ideas leading to a motivating and passionate learning environment. Techniques and skills required in the implementation of the solution are based on the decomposition of existing work, algorithmic reconstruction and the evaluation of the solution.

MOOC: Learning Programming with Processing

The MOOC comprises ten course lectures and is implemented on the virtual learning environment Moodle hosted by our college. The first lecture includes a course welcome and starts the course with a brief overview on fundamental questions about CS and programming, such as "What parts do computers consist of?", "How can we communicate with computers in order to make them solve our problems?", "What are algorithms and abstraction?" and "Why should I learn to program?"

In order to comply with the curricula of the on-site pre-study courses and the Introductory Programming lecture (which is compulsory for all CSBI bachelor students) the remaining course lectures follows a non-objects-first approach, which means, that the object-oriented programming paradigm is not introduced from the beginning explicitly. Course lectures 2 to 10 introduce the Processing Integrated

⁷⁴ <http://www.imoox.at> (last access: 06/2018)

Development Environment (IDE) and basic programming concepts, such as: variables, operators, data types, basic animations and user interactions, decisions using if- and if-else-statements and loops as well as functions, arrays and recursions.

Learning Sequence: Creating Visual Art with Code

The first practical programming exercise is positioned at the end of lecture 2. This lecture introduces the Processing Programming Integrated Development Environment (Processing IDE) which is used throughout the MOOC to write the Processing source code. Three videos and three handouts guide the students through downloading and installing the IDE as well as using it. In addition, they learn how to “draw” simple geometric objects and text by programming basic Processing commands. These include lines, triangles, rectangles or ellipses of different sizes, orientations, positions and the use of colours and transparencies. Finally, an overview of the Processing Reference is given which the course participants are encouraged to use to look up available Processing commands, functions and operators.

The overall exercise topic is to create graphical artworks through coding and to fake-copy a peer’s artwork. The assignment consists of three parts:

Part 1: Programming an artwork

In the first part of the exercise, the participants are asked to create a new sketch (i.e. Processing program) and program a graphical artwork of their own choice by using their previously acquired Programming skills and exploring the Processing Reference.

To keep the challenge of the second part of the exercise feasible especially for programming beginners, we have set the following constraints for each artwork: The sketch canvas size should not exceed 800 by 800 pixels. Only geometric objects of the types “2D Primitives” and “Typography”, referring to the Processing Reference, are allowed. The artwork should not consist of more than 20 of such objects. However, we did not limit the use of colours, contours and transparencies. As part of the exercise, we guide the participants through saving and executing their programs as well as we support them in creating a screenshot of their visual outcome. The submission consists of two parts:

1. Uploading the respective Processing source code file as well as the screenshot of the sketch window, which displays the artwork upon code execution.
2. Creating a thread in a forum provided for the exercise and uploading only the screenshot (but not the source code file) of the artwork, a fitting artwork title, as well as a short description of oneself (programming skills, motivation to take part in the course, optionally intended study programme, or similar).

Part 2: Faking a peer’s artwork through coding

After students have programmed and uploaded their artwork to the forum, every participant is asked to browse through the forum and choose one artwork programmed by a peer. Based on the provided screenshot he or she has to fake the artwork by programming it with Processing. The goal is to create a replica which is as close to the original as possible. The result, both the source code as well as a screenshot of the artwork copy, should be uploaded as a reply to the original forum entry. Along with it they are assigned to describe the main challenges they faced while copying the artwork and encouraged to respond to the personal introduction of the peer by stating similarities or differences concerning motivation for course participation, programming skill, study programme or similar.

Part 3: Discussion

Every participant is encouraged to check his or her own forum thread regularly for replies (i.e. artwork fakes programmed by peers). If a reply is received, the student should compare the fake to the original artwork and explain in the forum how the copy can be identified as such. Also, the original author is supposed to check the fake source code, compare it to the original code and discuss how the Processing programs differ and whether as well as how this is apparent in the resulting visual output or not.

Exercise Grading

Our main objective behind the exercise is to follow the four P's by Resnick, to foster a creative, playful and individual experimentation with coding and to overcome potential fears or doubts of programming easily. Therefore, we set only the first part of the exercise as compulsory. This means, for the overall course accomplishment we took into account if a student has uploaded an artwork as well as created a forum topic.

The second and third parts are voluntary yet strongly recommended, but not taken into account for grading or a successful course accomplishment. The reason for this decision are the students' heterogeneous pre-course programming skills which result in various levels of individual challenge and time needed to accomplish the assignments. Our priority was to challenge both beginners as well as advanced students at their individual levels. They should have fun and be able to solve the exercise to their own abilities, rather than leave anyone feeling overloaded or even unable to solve the exercise especially at this early stage of the course.

Didactic Background

With this exercise, we aim at encouraging the students to experiment with code and to take their first steps in programming in a playful, creative and interactive way while practicing the use of the Processing language and the use of the IDE. Without being much aware of the underlying programming concepts or processes the participants should explore and experiment with the use of functions, parameters and the computer graphics coordinate system as well as with the order of code execution. The graphical focus of the exercise and the immediate visual output of their programmed code enables visible feedback of the impact of their programmed code to the output sketch immediately. This supports them in reflecting their written code and the corresponding graphical results through learning by doing.

As the assignment is implemented as the first hands-on programming exercise in the whole course, programming skills are explicitly not needed as a prerequisite. Participants are provided only with basic tools they need to solve the exercise and are encouraged to explore and "*play*" creatively and individually using these tools as a starting point in order to create visual art.

Through this playful hands-on experience, we foster the students' curiosity and support them in experimenting with coding independently. With the programming reference, they are also equipped to try out further commands and extend their programming skills already at this early stage of the course. Using this approach potential fears and barriers concerning programming and technology can be overcome easily and stress as well as pressure to perform or deliver are avoided. Students are supported to study at their individual level, i.e. by creating simpler or more sophisticated and detailed artworks depending on their own programming experience.

Finally, this exercise aims at community-building among the course participants, i.e. prospective CSBI students already before they start their study programs. We consider this as important as our university is located in downtown Vienna and its distributed building structure does not support informal student socializing well. This means, that in addition to the challenges freshmen face with regard to administrative and organizational aspects as well as the contents of their studies, they also have to start networking with their peers, find friends and build study groups on their own at semester start. With the interactions encouraged between peers especially during this exercise, we support prospective students in getting to know each other prior their studies start so they might be able to master the challenging first weeks of their studies in teams rather on their own.

Results

During the summer holiday break in 2017 we pilot-tested the course with prospective students of the CSBI bachelor programmes. From a total of 315 active course participants, 126 students (i.e. 40%) submitted an artwork (exercise part 1). *Active* refers to having at least one compulsory course activity accomplished successfully. Figure 1 shows examples of artworks programmed and uploaded by MOOC students.

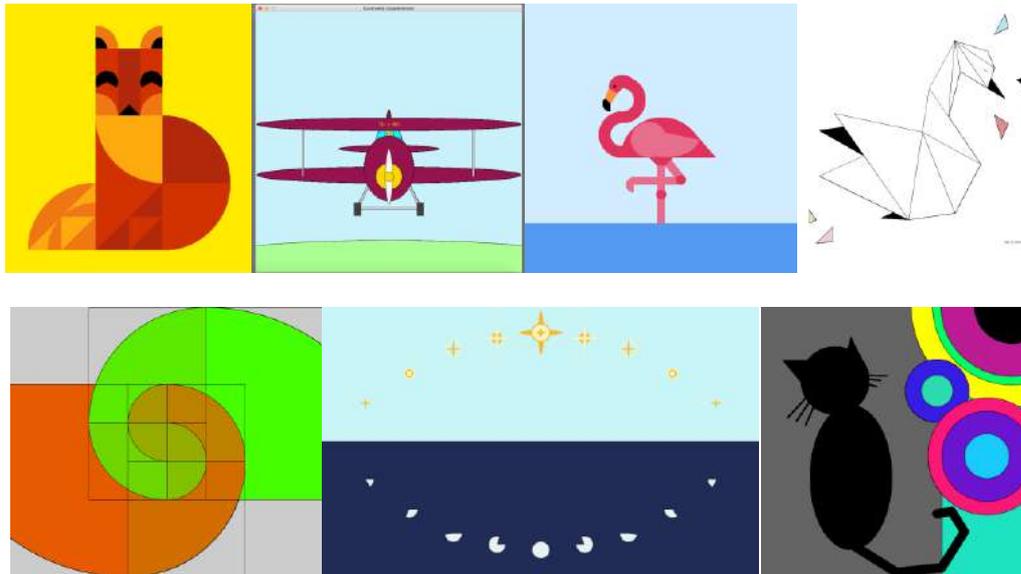


Figure 1. Examples of artworks programmed by course participants (ref. exercise part 1)

These examples visualize the huge variety and level of creativity of the resulting artworks programmed by the course participants using simple Processing commands only.

In total 38 artwork fakes were created by the students. Despite the fact that not every artwork was faked, the resulting copies are incredibly close to the original as Figure 2 shows exemplarily. Based on the results, it can be assumed that the course participants took high efforts in copying the original artworks as exact as possible. Hence, only subtle differences are perceivable, such as in different colour or transparency nuances or command order i.e. occlusion (e.g. artworks in Figure 2, rows 2 and 5) or small differences in position and scale of objects (e.g. artworks in Figure 2, rows 1, 3 and 4).

Participants' Feedback

At the end of each course lecture we asked the students for a brief feedback about what they liked and disliked about it and how they would grade the lecture. In addition, we conducted an anonymous online survey at the end of the course in order to get feedback about the overall course. With respect to the assignment the received feedback shows that the assignment was well-accepted among the students.

"The exercise was very much fun..."

"The exercises were cool. I personally find the interaction with other course participants through a discussion forum a really very good idea. By analysing the others' contributions one gets to know new ways of thinking and becomes more familiar with the geometric objects to be used in the exercise in general. The more participants in the course, the better the result!"

On the one hand, two students were concerned that their artworks should be shared in a forum, on the other hand there were also requests for more exercise of this style throughout the course, because it supports creativity and requires interaction with peers.

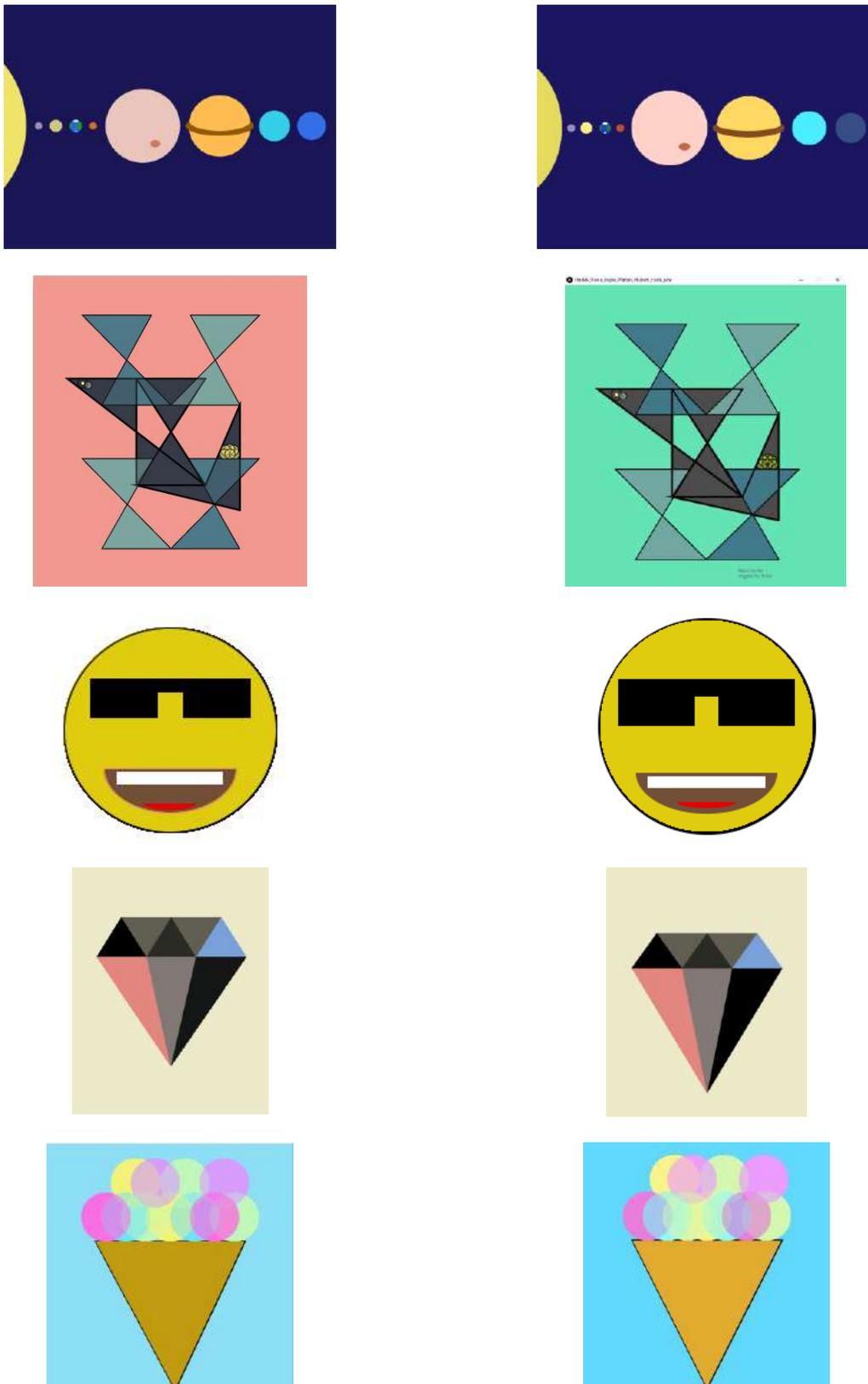


Figure 2. Examples of original artworks (left) and their fakes (right) programmed by course participants (ref. exercise parts 1 and 2)

In several comments, they appreciated that the artwork exercise allowed them to apply the learned content in a creative task and that they could independently create and program their own artwork as

well as having the opportunity to choose and be creative in copying artworks of peers. However, one student found it “*very challenging to create an exact copy of the original artwork*”. The “*simple introduction to programming allowing for visible feeling of success immediately*” was appreciated and another course participant evaluated the exercise as “*playful yet professional introduction to programming with Processing*”.

We compared the feedback on this learning sequence with feedback received on other parts of the course and conclude that this exercise was experienced as very exciting and encouraging.

Conclusion and Discussion

Although we intended to create a task that allows creativity, communication and peer feedback, we were overwhelmed by the passion and playful way of learning by the participants. It is a further proof that the 4 P's of Creativity are powerful. It also shows that creative tasks can enrich a mainly instructive MOOC. The only disadvantage was the relatively low completion rate of the fakes compared to the original artworks observed in our first run of the MOOC. In our opinion task part 2 provokes less creativity, has a higher difficulty and needs a high effort to complete, although it boosts learning evaluation CT skills. In the next runs of the MOOC we will improve this learning sequence according to our gained experiences and we will include further learning sessions that support creativity.

References

Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, 28(1), 23-44.

Ebner, M., Lackner, E., & Kopp, M. (2014, October). How to MOOC? - A pedagogical guideline for practitioners. In *The International Scientific Conference eLearning and Software for Education* (Vol. 4, p. 215). "Carol I" National Defence University.

Koedinger, K. R., Kim, J., Jia, J. Z., McLaughlin, E. A., & Bier, N. L. (2015, March). Learning is not a spectator sport: Doing is better than watching for learning from a MOOC. In *Proceedings of the second (2015) ACM conference on learning@ scale* (pp. 111-120). ACM.

Resnick, M. (2014, August). Give P's a chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the Third International Constructionism Conference*. Austrian Computer Society, Vienna (pp. 13-20).

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120-129). ACM.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Posters

Educational Robotics for STEM: From Workshops to Curricula and Framework

Carina Girvan, *girvanc@cardiff.ac.uk*
School of Social Sciences, Cardiff University,
Wales, UK

Wilfried Lepuschitz, *lepuschitz@pria.at*
Practical Robotics Institute Austria, Austria

Ivaylo Gueorguiev, *ivo@esicenter.bg*

Christina Todorova, *tina@esicenter.bg*
European Software Institute – Center Eastern
Europe, Sofia, Bulgaria.

Chronis Kynigos, *kynigos@ppp.uoa.gr*
Marianthi Grizioti, *margrizioti@gmail.com*
UoA ETL, National and Kapodistrian University of
Athens, Athens, Greece.

Angele Giuliano, *angele@acrosslimits.com*
Annalise Duca, *annalise@acrosslimits.com*
AcrossLimits, Malta

Julian M. Angel-Fernandez,
julian.angel.fernandez@tuwien.ac.at
Markus Vincze, *vincze@acin.tuwien.ac.at*
Technical University Wein, Vienna, Austria

Abstract

This poster presents the outcomes of the first two years of a three-year project exploring and refining the use of educational robotics to engage young people in STEM education. It shows the development of activity plans which leverage constructionist concepts such as powerful ideas, objects to think with and the construction and sharing of personally meaningful artefacts in order to explore, test and extend understanding within STEM domains. Workshops in which these activity plans have been tested and refined, have so far engaged over 3,000 children between the ages of 7 and 18 in six European countries. Additionally, conferences and competitions for young people have been held each year where they compete and collaborate with others. These workshops and competitions have been systematically evaluated through QUAL+quant mixed methods, in order to inform the development of future activities and refinement of existing ones. Through this process we identify key components of successful educational robotics activities for STEM education. These have, in-turn, informed the design of a framework for educational robotics. The activity plans have informed the bottom-up development of a generic curriculum for educational robotics in STEM and are available on a dedicated repository. This poster presents a snapshot of each of these project outcomes.

Keywords

Constructionism; educational robotics; science; technology; engineering; mathematics; learning; STEM

Introduction

Constructionism and the use of robots in education seemingly go hand-in-hand. Yet, any review of the literature on the potential of educational robotics to engage young people in science, technology, engineering and mathematics (STEM), shows that it is often only lip-service that is paid to the ideas of Seymour Papert. Simply stating an activity to be constructionist, does not make it so. Educational robotics, as an area of research, is also one that cuts across various fields with academics from education, psychology, mathematics, engineering, computer science and science conducting research in this space, with various pedagogical approaches, research methods and objectives, with activities and research often implemented outside of the constraints of the traditional school system. Teachers, keen to embrace robotics in the classroom, can struggle to know how to design activities. They face pressures such as demonstrating measurable learning outcomes for individual students, siloed subject teaching and packed curricula. Should concepts be taught in advance and applied to robotics activities, should activities be structured to provide or require engagement with specific concepts to be completed, or should we allow students to explore, test and extend their own understanding in their own time and in their own way, potentially never explicitly engaging with the concept which should be learned?

The project presented in this poster aims to foster the interest in STEM and powerful ideas that children have at a young age (Lammer, et al., 2017), but is often lost as they progress through formal education and are increasingly exposed to societal pressures. In particular we are interested in the potential of educational robotics to engage all young people with STEM, particularly girls (Nugent, et al., 2010). The project involves the development and redesign of robotics tools and learning spaces for young people, implementing these and other commercially available robotics tools in workshops and competitions with over 4,000 children ages 7-18, in six European countries. The workshops and competitions have been systematically evaluated through a large-scale, QUAL+quant mixed methods research design, involving in-depth case studies in each country. This research has informed the design of new and the development of existing educational robotics activities. It has contributed to the development of activity plans which guide the design of activities and exemplify a generic curriculum for robotics. As this work matures, we aim to develop a framework for the design of educational robotics activities, which is accessible to both teachers and researchers.

Constructionist concepts such as powerful ideas, objects to think with and the construction and sharing of personally meaningful artefacts in order to explore, test and extend understanding within STEM domains, are used to underpin the design of activity plans and in-turn workshops. They also provide a way to explicitly introduce 21st Century Skills, the creative arts and creative expression in such educational robotics activities.

Activity Plans and Curricula

A standardised activity planning template acts as a common learning design instrument through which project partners design and communicate their design of educational robotics workshops to each other. The activity plan template aims to raise awareness of the key characteristics of constructionist educational robotics activities (Yiannoutsou, et al., 2017) and has developed in response to partner feedback as well as issues and opportunities highlighted in the evaluation. This template has also informed the design of a bespoke online repository of workshops for teachers and others interested in the use of educational robotics.

Activity plans, act as a mediating artefact through which various groups involved in the project such as teachers, academics, commercial partners, engineers and computer scientists, are able to share their own beliefs and experiences. They provide a way to identify and question assumptions. Ultimately, they act as a tool through which a shared understanding of the key components of the design of educational activities can be developed and expressed.

Reviewing the activity plans and other freely accessible educational robotics resources, a bottom-up approach is currently being used to organise the implemented activity plans and develop a series of generic curriculum paths, organised around STEM domains and complemented by arts, business and 21st century skills.

Workshops and Competitions

By the end of the project, activity plans will have been implemented in workshops involving more than 4,000 students in Austria, Bulgaria, Greece, Ireland, Malta and the UK, between the ages of 7 and 18. Workshops typically take place in schools during normal hours. Some workshops are held as one/two full-day events, where the normal timetable is suspended, whilst others occur over several weeks, integrated into subject specific lessons and the regular school timetable. A wide variety of robotics tools and learning environments have been used or created and integrated into the workshops, including LEGO Mindstorms, LEGO WeDo, Finch, Hedgehog (Koza et al., 2016) and also SLurtles (Girvan et al., 2013) in purpose designed virtual worlds. Workshops may be delivered by a small team including the class teacher, or delivered solely by the teacher, without any additional support. There are advantages and disadvantages to these different approaches, both in terms of learner outcomes and research.

There are a range of national and international robotics competitions for young people in Europe. This project examines the European regional Botball tournament which involves multiple rounds and different forms of competition. While teams competing against each other are the most common form of robotics

competitions, the Botball competition includes an Alliance round in which two teams collaborate to score the most points together. Alongside the competition there is also a conference for which each team prepares and delivers a paper. These often focus on technical developments or suggested improvements to the kit identified by the students and tend to focus on the engineering and technology domains.

Research

One of the primary aims of the project was to increase young people's interest and engagement in STEM subjects and careers. The research team were interested to find out about what concepts students participating in workshops and competitions learned, whether gender stereotypes were challenged and whether those who were originally not interested in STEM became interested through the activities, and why. The research explores whether there is a gender dimension to the roles students take and their participation in activities, in addition to whether children develop soft-skills. Another aim of the project was to develop multiple ways for students to engage with robotics activities and for students to take ownership of the activities they were involved in.

To evaluate whether the aims of the project have been met and to begin to answer the research questions, data collection has been ongoing throughout this project. Students involved in the workshops and competitions have been invited to participate in the research, with parental consent. A single evaluation kit is used by all partners, which provides all data collection instruments and schedules, including: pre and post workshop/competition questionnaires, interview and observation schedules, procedures for artefact collection (robots constructed, code written, drawing produced, etc) and reflective activities for teachers and students.

In the first year of the project the aim was to pilot the evaluation kit and explore existing practices, to inform the development of the framework and activity plans, which in turn would influence the design of activities in the second year of the project. Each workshop or competition was treated as a single case study. Within each case study, data was analysed using the constant comparative approach, allowing the researcher to remain open to emerging themes throughout. Single case studies could then be drawn together to allow analysis across or within countries, age groups, gender or other factors.

The second year of the project focused on the recommendations which emerged from the first year, their implementation and the impact, in addition to exploring the original research aims of the project. While analysis of the data was more focused, it remained largely interpretivist, allowing researchers to remain open to emerging codes and themes. Currently the project is in its final year, in which the data analysis has become yet more focused on a specific set of issues raised in the second year, however interpretivist approaches remain central. In addition, the third year provides an opportunity to explore the impact of the developing framework by contrasting the findings with the previous two years.

Developing a Framework

The development of an inclusive framework for the design of educational robotics activities in STEM began with a review of the literature and existing frameworks. Synthesising this work and considering it in relation to the findings of the research evaluation in years one and two, demonstrated that whilst many aspects of these frameworks were relevant, there were aspects such as team formation and modes of collaboration which were either missing or lacking insufficient detail.

Informed by the evaluation, the project team are currently developing a framework for educational robotics activities which is underpinned by constructionist ideas.

Acknowledgements

The project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 665972. Project Educational Robotics for STEM: ER4STEM

References

- Girvan, C., Tangney, B. & Savage, T. (2013). SLurtles: A tool to support constructionist learning in Second Life. *Computers & Education* 61(1) 115-132.
- Lammer, L., Lepuschitz, W., Kynigos, C., Giuliano, A., & Girvan, C. (2017). ER4STEM Educational Robotics for Science, Technology, Engineering and Mathematics. In M. Merdan, W. Lepuschitz, G. Koppensteiner & R. Balogh (Eds.) *Robotics in Education Advances in Intelligent Systems and Computing*, vol. 457. Switzerland: Springer International Publishing, 95-101.
- Koza, C., Lepuschitz, W., Wolff, M., Frank, D., & Koppensteiner, G. (2016, November). Hedgehog light—a versatile, white box educational robotics controller. In *International Conference EduRobotics 2016* (pp. 229-232). Springer, Cham.
- Nugent, G., Barker, B., Grandgenett, N., & Adamchuk, V. I. (2010). Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes. *Journal of Research on Technology in Education*, 42(4), 391-408.
- Yiannoutsou, N., Nikitopoulou, S., Kynigos, C., Gueorguiev, I., & Fernandez, J. A. (2017). Activity plan template: a mediating tool for supporting learning design with robotics In M. Merdan, W. Lepuschitz, G. Koppensteiner & R. Balogh (Eds.) *Robotics in Education Advances in Intelligent Systems and Computing*, vol. 457. Switzerland: Springer International Publishing, 3-13. doi:10.1007/978-3-319-42975-5_1

Towards a Generic Curriculum for Educational Robotics in STEM: From Scientific Concepts to Technologies and Powerful Ideas

Ivaylo Gueorguiev, *ivo@esicenter.bg*
Christina Todorova, *tina@esicenter.bg*
European Software Institute – Center Eastern Europe, Sofia, Bulgaria

Nikoleta Yiannoutsou,
nyianoutsou@ppp.uoa.gr
Kristina Greka *xristgreka@gmail.com*
UoA ETL, National and Kapodistrian University of Athens, Athens, Greece

Pavel Varbanov, *pavel@esicenter.bg*
George Sharkov, *gesha@esicenter.bg*
European Software Institute – Center Eastern Europe, Sofia, Bulgaria

Carina Girvan, *girvanc@cardiff.ac.uk*
School of Social Sciences, Cardiff University, Wales, UK

Julian M. Angel-Fernandez,
julian.angel.fernandez@tuwien.ac.at
Vienna University of Technology, Vienna, Austria

Lisa Vittori, *vittori@pria.at*
Practical Robotics Institute Austria, Austria

Annalise Duca,
annalise@acrosslimits.com
AcrossLimits, Malta

Abstract

This paper and its corresponding poster present a “work in progress” concept for the visualization of 19 activity plans, into a generic curriculum map for teaching STEM concepts through constructionist robotics activities. There are six educational paths that represent potential use cases and these have been validated through 148 educational robotics workshops implemented with children between the ages of 7 and 18 in six European countries.

Keywords

educational robotics; curricula; robotics; workshop; science; technology; engineering; mathematics; learning; STEM

Introduction

Many children lose their natural curiosity for how things function and interrelate to each other along the way into their lives as young adults. The ER4STEM project aims to turn curious young children into young adults passionate about science and technology with a hands-on use case: robotics. In this poster we present our effort to develop a generic curriculum for robotics using a bottom-up approach: i.e. analysing several activity plans produced as part of the project. Our analysis follows different educational paths aiming to construct a curriculum that addresses the different facets of educational robotics: a) powerful ideas; b) scientific concepts and skills; c) technologies and engineering; d) mathematics. The result of this analysis is presented in the form of a ‘metro map’ showing how an educator could navigate through the educational robotics activity plans taking each of these educational paths.

We embarked on this effort understanding that a curriculum is a complex concept, which ends up being a matter of profound philosophical discussion, as it yet revolves around the notion of knowledge and notably, of what is of importance to be learned for the individual within the context of the society, in which they function (Bers, 2003). However, in our case, instead of following a top-down approach and looking at what should be learned in terms of educational robotics, we followed a bottom-up approach. Consequently, we analyse how we can identify teaching and learning elements, which are connected to the nature of educational robotics: i.e. constructionist technology, multidisciplinary domain, subject matter and a domain for contextualized learning of other subject matters.

ER4STEM project is funded by the European Commission through the Horizon 2020 Program (H2020, Grant agreement no: 665972).

A Generic Curriculum for Applying Educational Robotics in STEM-Related Education

Among the core goals of the project is to deliver a generic educational robotics curriculum. It aims to facilitate usability of the project's products by a broader community of relevant stakeholders such as, but not limited to, teachers, researchers and 3rd sector organisations.

Assuming the general definition of curriculum being a general plan for educational activities, Adams and Adams (2003) define a curriculum as everything that goes in the learners' life such as planned and not planned interaction of pupils with educational objectives, instructional content, materials and resources used and materials and resources not used, the sequence of courses, objective, standards and interpersonal relationships.

In order to accomplish the project objectives and to research the domain of using robotics in education, we organise educational robotics workshops. The activities developed follow a learning design instrument "Activity Plan Template", which aims at raising teachers' awareness of the key characteristics of educational robotics implemented in constructionist ways (Yiannoutsou et al 2017).

Taking into account the above mentioned curriculum definition and the experience gained so far, we decided to use a bottom-up approach for organising the 19 Activity Plans as adoptable generic curriculum paths organised around Science, Technology, Engineering and Mathematics domains and further complemented by arts, business and 21st century skills.

The activity plans were designed, implemented and validated through 148 educational robotics workshops with more than 3500 students implemented in Austria, Bulgaria, Greece, Malta, Ireland and the UK, over two years. However, due to time constraints, the entire curricula paths, consisting of an educational sequence of activity plans as a single entity were not implemented. Therefore, we consider the curricula on the level of educational paths of a generic nature by design and they will need to be further adapted to meet context specific desired learning outcomes, specific objectives and classroom needs.

The Six Educational Paths Visualised as a "Metro Map"

With this generic educational robotics curriculum, we want to assist teachers in adapting a learning path, consisting of a set of educational robotics workshops or single activities within the workshops activity plans, that are suitable for their context, students, specific objectives and prior knowledge.

The six educational paths built around the STEM domains, are based on the following use cases:

1. An educator aims to teach **technology and engineering with Lego Mindstorms robots**. This is a typical use case of using robots for education
2. An educator aims to teach concepts of **science, technology and engineering while demonstrating various technology platforms** such as Arduino, Beebots, Hedgehog, LEGO Mindstorms and LEGO WeDo **as well as a vast plethora of** programming languages C++, Python, LEGO Mindstorms Programming, Scratch or SNAP.
3. An educator aims to cover all STEM domains, **Science, Technology, Engineering and Mathematics, while also developing creativity and business skills among the students**.
4. An educator aims to teach concepts in **science, technology and engineering through "white box"** platforms such as: Arduino, Hedgehog, LEGO WeDo with a stronger focus on electronics and widely used industrial programming languages such as C++ and Python
5. An educator aims to teach **Mathematics by using educational robotics technology**. This path covers multiple mathematical concepts;
6. An educator aims to teach concepts in **Technology, Engineering and Mathematics**.

While providing those generic educational paths, we still consider it of utmost importance for teachers, pedagogues and researchers, while following the suggested curriculum paths, to tailor the activity plans to their context and thus design their own, separate curriculum.

The generic learning paths are a curriculum property, which will be further elaborated in the next months and will represent certain relationships between the activity plans contained in the curriculum.

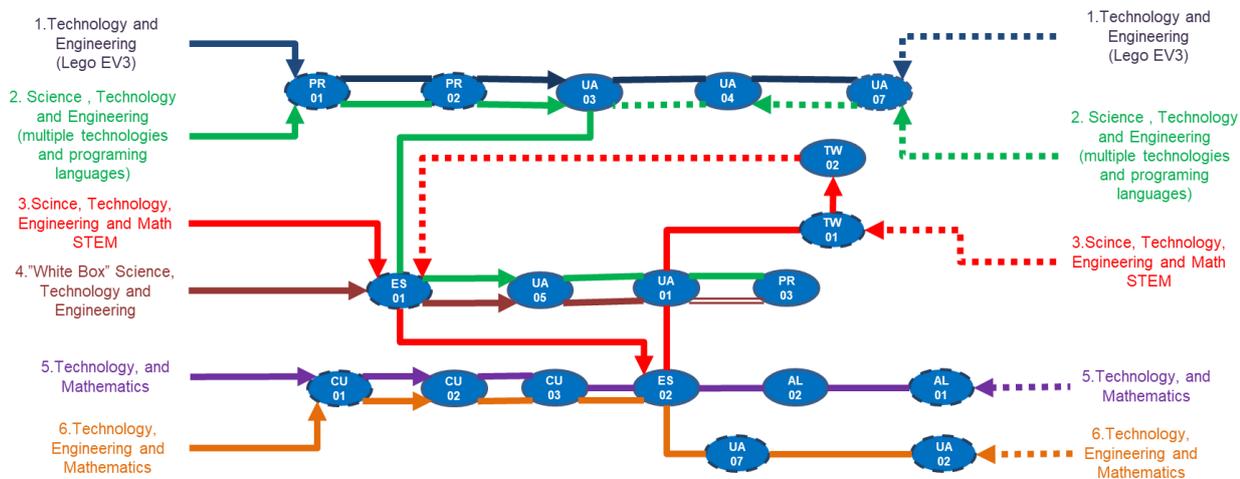


Figure 1. Six educational paths of generic curricula presented as a metro map in which each point is an activity plan for and educational robotics workshop

To illustrate the concept, we could take for instance red line 3, as presented in Figure 1. The entire educational path, represented by the red line, consists of 37 hours of educational robotics content to cover all STEM domains complemented by concepts from domains such as business and art. The first “metro stop” where we “start the journey” is ES01 named “Educational Robotics and Creativity Workshop”. Within this workshops, students with no prior experience, related to the implementation of a technical project in a collaborative environment explore the fields of technology and engineering while constructing and programming a robotic vehicle and applying mind-mapping to generate ideas on the application of robotics in various fields. The provoked students’ interest and curiosity in technology are further developed within the second workshop ES02: Visualising Mathematics with Mathbot” where the students, who are already familiar with the basics of programming robot in Scratch, are exploring mathematical concepts, such as measuring angles, types of triangles, properties of a circle and proportions. In the next “metro stop”, UA01 named “An introduction to robotics - exploring planet ektonis” students have a chance to learn, further develop and practice concepts of science, technology and engineering while also using mathematics and art. At the last “metro stops” TW01 named “Robot-video elementary” and TW02 named “Robot-video advanced” the students are embarking on a hands-on learning experience in technology and engineering, while developing business skills and expressing their creativity through video and robotics. The same “metro line” could be taken the other way around, just by following the dashed red lines. As one can notice, the “metro map” illustrated in Figure 1, three of the five “metro stops” of the red line are also part of other proposed generic educational paths, serving different desired learning outcomes.

Powerful ideas

Our work in this dimension is in progress and focuses on what the central ideas are around which the activities evolve and how these are expressed in the robotic constructions. Powerful ideas are a central theoretical construct of constructionism and involve the use of knowledge in new ways where learners can establish personal epistemological connections with various domains of knowledge (Paret, 2000). Furthermore, analysing powerful ideas Bers (2008) describes them as domains of knowledge where many diverse topics or subjects co-exist and become explicit. Viewing activity plans from this lens poses the following challenges: a) powerful ideas might not imply a “progress” from one activity plan to another

(in terms of student age, prerequisites etc) so we need to seek other connections that can be meaningful; b) not all activity plans are built around powerful ideas but they might have the potential to evolve in this direction. While traveling in the STEM metro line, students also develop collaborative skills, take roles within groups, solve practical problems in teams, communicate with other teams to exchange ideas and tips in the social and actions domains.

Conclusion

We are confident that the presented approach for the organization of a generic curriculum will provide value to educators that are considering to apply educational robotics activities in their classrooms or would like to generate ideas on how to diversify their already applied approach. Furthermore, the work presented will attempt at visualizing the context and content of the educational robotics activities designed and developed within the project.

References

- Adams, K. L., & Adams, D. E. (2003). *Urban Education a Reference Handbook*. Santa Barbara, California: ABC Clio.
- Bers, M. (2008). *Blocks, robots and computers: Learning about technology in early childhood*. New York: Teacher's College Press
- Papert, S. (2000). What's the big idea? Towards a pedagogy of idea power. *IBM Systems Journal*, 39(3-4), 720–729.
- Walker, D.F (2003) *Fundamentals of curriculum: passion and professionalism*, Mahwah, N.J: L. Erlbaum Associates.
- Yiannoutsou N., Nikitopoulou S., Kynigos C., Gueorguiev I., Fernandez J.A. (2017) Activity Plan Template: A Mediating Tool for Supporting Learning Design with Robotics. In: Merdan M., Lepuschitz W., Koppensteiner G., Balogh R. (eds) *Robotics in Education. Advances in Intelligent Systems and Computing*, vol 457. Springer

Heuristic Potential of Open Institutional Models in Researchers Education

Liudmyla Kryvoruchka, kryvoruchkald@ukma.edu.ua
National University of Kyiv-Mohyla Academy, Ukraine

Abstract

The national and cross-border training networks for PhD education established in accordance with EU Principles for Innovative Doctoral Training (2011) and Open Science agenda have already contributed to significant transformation of traditional researcher education settings across Europe. This poster presents the preliminary results of Erasmus+ DocHub project (2016-2018) being implemented in Ukraine by partnership of 5 Ukrainian and 4 EU HEIs together with 6 Ukrainian research institutions. The obvious impact of such collaborative efforts at organisational and systematic level often hides substantial heuristic potential of diverse instructional and mentoring strategies as well as learners' experiences. The jointly established learning environment of DocHub is structured by the idea of openness and aimed at engaging PhD students and early researchers in active and continuous exchange and thus enhance dialogic and innovative mind-set and support experimentation.

Keywords

Openness, inter-institutional cooperation, dialogic and innovative mind-set

Background

The need for educating doctoral students as competent innovative researchers has been actively discussed during the last decade. Rapidly increasing complexity of research work in the open global research community encourages HEI to search for the best practices and institutional structures to ensure their PhD graduates are ready for future challenges.

Doctoral education is considered a key contribution to all 3 main policy goals of EU research and innovation introduced in 2015: Open Innovation, Open Science and Open to the World. Open Science creates the vision of research as collaborative and transparent, therefore, recommendations from the Bologna Process ("Salzburg Recommendations" 2005, "Salzburg II Recommendations" 2010; "EU Principles for Innovative Doctoral Training" 2011) have to be re-interpreted by HEIs to implement an open collaborative model of doctoral programs.

In the context of Ukrainian educational system the need for the collaborative model is crucial to accelerate the Bologna 3rd cycle reform. Traditional one-to-one mentorship system remains a basis for the academic practices of Ukraine, so implementation of structured PhD training according to the Law "On Higher Education" (enacted in 2014) is still challenging for both HEIs and research institutes of the National Academy of Sciences. Though in 2016 the structured PhD training model became obligatory, only few institutions have developed PhD training programs of sufficient quality and relevant teaching methodologies compliant with the EU recommendations. National reform of the 3rd cycle in Ukraine requires cultural change through exposure to an alternative approach to researchers' training.

The EU funded Erasmus+ project "Structuring cooperation in doctoral research, transferrable skills training, and academic writing instruction in Ukraine's regions/DocHub (2016-2019), engaged 5 Ukrainian universities and 6 research institutes of the National Academy of Sciences of Ukraine (with the expertise input from 4 EU HEIs) into establishing national centers of excellence in PhD training for aggregation of critical mass, boosting research capacity and provisioning of high-quality transferrable skills and language training programs.

In this paper, we present and discuss the tools and models of collaborative doctoral training to show how open culture being established by DocHub project activities can foster both institutional development and the personal development of doctoral candidates and educators.

Impact of the DocHub community

In the diverse landscape of European doctoral education we see many examples of successful open models of doctoral training implemented on institutional and/or national level, such as: Innovative Training Networks (ITN), European Industrial Doctorates (EID) funded by Marie Skłodowska-Curie actions in Horizon2020, Doctoral Centers of Excellence, Doctoral Training Networks (UK), Finnish Doctoral Training Network (Finland), Regional Doctoral Schools (France). The key benefits of collaborative practice are establishing a high-level research environment with complementary competences, sharing of knowledge and infrastructure, the exchange and sharing training modules/courses, building supervision and mentoring groups for individual research projects, creating possibilities to work in interdisciplinary area, enhancing future employability of doctoral candidates, and an increase in mobility.

As we can see the networks are based either on regional models (i.e. Finnish Doctoral Training Network, Finland) or research subject model (i.e. Doctoral Training Networks, UK). The DocHub project will use both models in Ukraine by establishing:

- (1) 5 HEI-based regional centers of excellence (DocHubs in Kyiv, Lviv, Dnipro, Mykolaiv and Kharkiv) with developed infrastructure and systems for training advanced academic writing in English, including components of the blended learning format which has become crucial for the internationalization of research and innovation at the national level; providing validated curricula of transferrable skills according to EU quality assurance standards;
- (2) 5 subject-based inter-institutional research groups (Educational Policy, Chemistry/Biochemistry, IT, Political Sciences, and Finances, more than 110 in total) to provide collaborative professional training, supervising and mentorship by the cooperation of university researchers and scholars from research institutes.

To accomplish the above goals the project group is developing (with the support of the Ministry of Education and Science of Ukraine) the regulatory framework for inter-institutional cooperation and further development of the 3rd cycle of reforms based on the EU best practices. This involves drafting and adopting policies and national regulations (including changes to legislation) related to licensing and accreditation of PhD programs, funding of doctoral education, procedures of qualification. Project group initiated a regular open seminar with the participation of doctoral education stakeholders for 2018-2019 academic year.

A regional-based networking model in Ukrainian context proved to be an effective tool of enhancing institutional capacity of the regional universities: the modular courses (Advanced Academic Communications, Transferrable Skills Curricula) were designed and developed as a result of joint effort and active discussion of peer-group with the expert input from EU partners; but after internal and external evaluation the curricula was piloted by each lecturer/trainer individually with episodic co-teaching sessions. Using more advanced course materials increased the level of teaching skills and knowledge. As a result of DocHub activities all institutions introduced new courses and research topics in their doctoral training programs

In a subject-based model DocHub partners actively used the complimentary expertise and developed training programs based on the individual research interests and achievements of engaged researchers from research institutes and universities (36 courses). The aggregated groups of PhD students from different HEIs (up to 900 in total) participated in piloting curricula introduced by academicians from different regions and institutions.

Both models of networking can be a solution for most of Ukrainian HEIs that typically receive just a few state-budgeted PhD positions in each subject area and have difficulties with provisioning effective training and establishing critical mass. One of the objectives of the DocHub is to develop a legal framework and a model for “national credit mobility” to boost inter-HEIs cooperation and overcome the rapture between institutions of the National Academy of Sciences of Ukraine and universities inherited from the USSR.

Adoption of the DocHub Code of Practice and the formal establishing of collaborative units at the institutional level was re-scheduled from the 1st to the 2nd year of the project to give time for defining a common vision and a shared understanding of responsibilities based on realistic research-based connections. On the basis of agreed general framework (principles of inclusiveness, high academic standards, internationalization, research integrity) each DocHub should now set up a working model for decision making and management (via Academic Councils and administrative group) and tools for effective communications (seminars, conferences, working groups, groups for supervision and mentoring).

Mandatory national regulations on doctoral training (enacted in 2016) require provision of structural educational programs for at least 2 years of study totaling from 30 to 60 ECTS. In accordance with “Salzburg Recommendations” and “EU Principles for Innovative Doctoral Training” the significant part (at least 20 ECTS) of obligatory credit workload should be aimed at developing transferrable skills (research ethics, informational literacy, research management and language skills). However the issue of instructional design differentiated from 1st and 2nd cycles in content and purposes and suitable instructional methods for effective language and transferrable skills training is the most challenging area of doctoral education in Ukraine due to the lack of skills required by new framework of global research practice (like Open Science model) and issues with research integrity. Therefore project group initiated training on trends in 3rd cycle best practices and established a regular seminar on 3rd cycle teaching for 2018-2019.

DocHub project team is actively promoting collaborative scheme of doctoral training at regional and national level. Upon project completion regional DocHubs should be considered as centers of expertise for provision not only PhD students training, but for continuing professional development of researchers at all stages of career. DocHubs research communities will contribute to regional development serving as points of cooperation with interested public and business entities.

Collaboration and networking in doctoral education is a transformative concept related to the emerging Open Education practices and Open Learning culture and thus will move the discussion about the model of researchers’ professional development forward.

Heuristic potential of open institutional models

DocHub allows doctoral students access to a large pool of concepts, techniques, methods, skillset as well as personal experience and micro-networks of each senior scholar. The obvious efficiency of such collaborative efforts at the organisational and systematic level of doctoral education hides substantial heuristic potential of diverse instructional and mentoring strategies.

Any partnership strives to seamless connections, but institutional collaboration not only increases the rates of knowledge exchange, but also conflicts of approaches and values that need to be resolved through individual reflective efforts.

DocHub networks generate a unique learning environment driven by epistemological tension between distinct academic cultures (often under-articulated) and scholar approaches, though aligning in the basic objective to produce original research.

According to the EU Principles of Innovative Doctoral Training (2011) “research excellence” is expressed by a researcher that was trained as a “creative, critical and autonomous intellectual risk taker, pushing the boundaries of frontier research”. In this context DocHub sets the “attractive institutional environment” for researchers that would be ready to take risk of holding the uncertainty of open dialogue.

To ensure continuation in such open research collaboration an individual is to be committed to accept the complementary role of any individual position or expertise and this is possible when a participant develops dialogic mind-set. At the institutional level partnership should establish the scheme for constant communications to minimize misunderstanding.

When co-teaching and group discussion of doctoral projects in an institutionally open supervisory group, as well as national and international mobility, become a routine practice amongst PhD students, we can infer that such settings encourage open learning culture. In collaborative schemes, both doctoral

students and senior researchers are directly exposed to diversified communities and practices. This will aid them in the transition to new technology-driven and societally engaged research.

Conclusions

It is apparent that Ukrainian universities and research institutions are highly committed to develop innovative training networks to increase their research and learning capacities to equip doctoral students and early researchers with competences needed in a knowledge-based economy.

The Open Institutional Model is now considered a very effective tool for engaging researchers in a cooperative model of research work in globalized context.

The joint learning environment of DocHub is structured around the idea of openness, the support of experimentation, and focused on engaging researchers in an active and continuous exchange, thus promoting the development of a dialogic and innovative mind-set.

Acknowledgments

The author gratefully acknowledge the support of the Erasmus+ KA2 – Cooperation for innovation and the exchange of good practices – Capacity Building in the field of Higher Education. In particular through the funding of the three year project “Structuring cooperation in doctoral research, transferrable skills training, and academic writing instruction in Ukraine’s regions / DocHub” with project code 574064-EPP-2016-LT-EPPKA2-SNHE-SP.

References

A Renewed EU Agenda for Higher Education.COM (2017)

Doctoral Education – Taking Salzburg Forward. Implementation and new challenges. (2016) EUA-CDA

Castaño Muñoz, J., Punie, Y., Inamorato dos Santos, A., Mitic, M. & Morais, R. (2016): How are Higher Education Institutions Dealing with Openness? A Survey of Practices, Beliefs and Strategies in Five European Countries. Institute for Prospective Technological Studies. JRC Science for Policy Report,

Huitt, W G, & Monetti, D M. (2017). Openness and the Transformation of Education and Schooling. Open: The Philosophy and Practices that are Revolutionizing Education and Science. Ubiquity Press, 43–65.

Principles for Innovative Doctoral Training (2011).

Statement from the EUA-CDE Global Strategic Forum on Doctoral Education (2013).

Construction of a Project Monitoring Application Iteratively and Incrementally

Pekka Mäkiaho, *pekka.makiaho@uta.fi*

Timo Poranen, *timo.t.poranen@uta.fi*

Katriina Vartiainen, *katriina.vartiainen@uta.fi*

University of Tampere, Finland

Abstract

Developing complex software applications is a challenging task. It is hard to recognise all needed features in advance, and therefore software development is often done iteratively and incrementally. Every iteration usually contains activities like planning, refining requirements, implementation and testing. In this paper, we describe how university student teams developed Metrics Monitoring Tool (MMT) application during the years 2014-2018. The construction process of the MMT has contained five larger development iterations so far. All versions have been tested comprehensively with dozens of real users.

Keywords

Iterative and incremental development; metrics monitoring; application

Background

“Learning by making” is a simplification of ideas behind constructionism learning theory (Papert and Harel, 1991). Constructionism underlines student-centered learning where students use information they already know to acquire more knowledge. (Alesandrini and Larson, 2002)

In computer sciences, students learn by studying different concepts and then implementing the concepts as small exercises (simple programs, algorithms, user interface designs etc.) or larger course works. The course works often implement many concepts at the same time. Larger software application are very complex, and they can contain hundreds or thousands of minor concepts that the developer team integrates to work as a whole.

To manage complexity, iterative, incremental and agile software development models (Sommerville, 2010) have become mainstream in software construction. Main idea behind these models is to build software piece by piece in iterative manner. This ensures that there is no need to plan everything beforehand, but instead it is enough to plan next development iteration carefully and to have a wider vision of the project goal. Knowledge of the development team increases, and vision can be adjusted regularly.

Design science research methodology focuses on the development of artifacts, like software. Hevner and others (2004) have given seven principles for design science research: i) Design as an artifact, ii) Problem relevance, iii) Design evaluation, iv) Research contributions, v) Research rigor, vi) Design as a search process, and vii) Communication of research.

In this paper, we describe incremental construction of Metrics Monitoring Tool (MMT) application. The research follows design science methodology principles.

Incremental development of the MMT

Metrics Monitoring Tool (MMT) is a tool designed to support project managers, project members and upper management in reporting and observing projects' proceedings. Version 2.9 was introduced in an article “MMT: A tool for Observing Metrics in software Projects” (Mäkiaho et al., 2017). Main screen of the MMT is shown in Figure 1.



Figure 1. Main screen of the MMT

The Figure 2 shows how MMT was developed incrementally and iteratively. The current version of MMT is 4.0. The tool was developed in Tampere University’s Software Project Management and Project Work courses during three academic years. The paradigm used was Design Science Research (Hevner et al., 2004). According to the Design Science principles an artefact (increment) was built to solve a problem and then evaluated how it was succeeded.

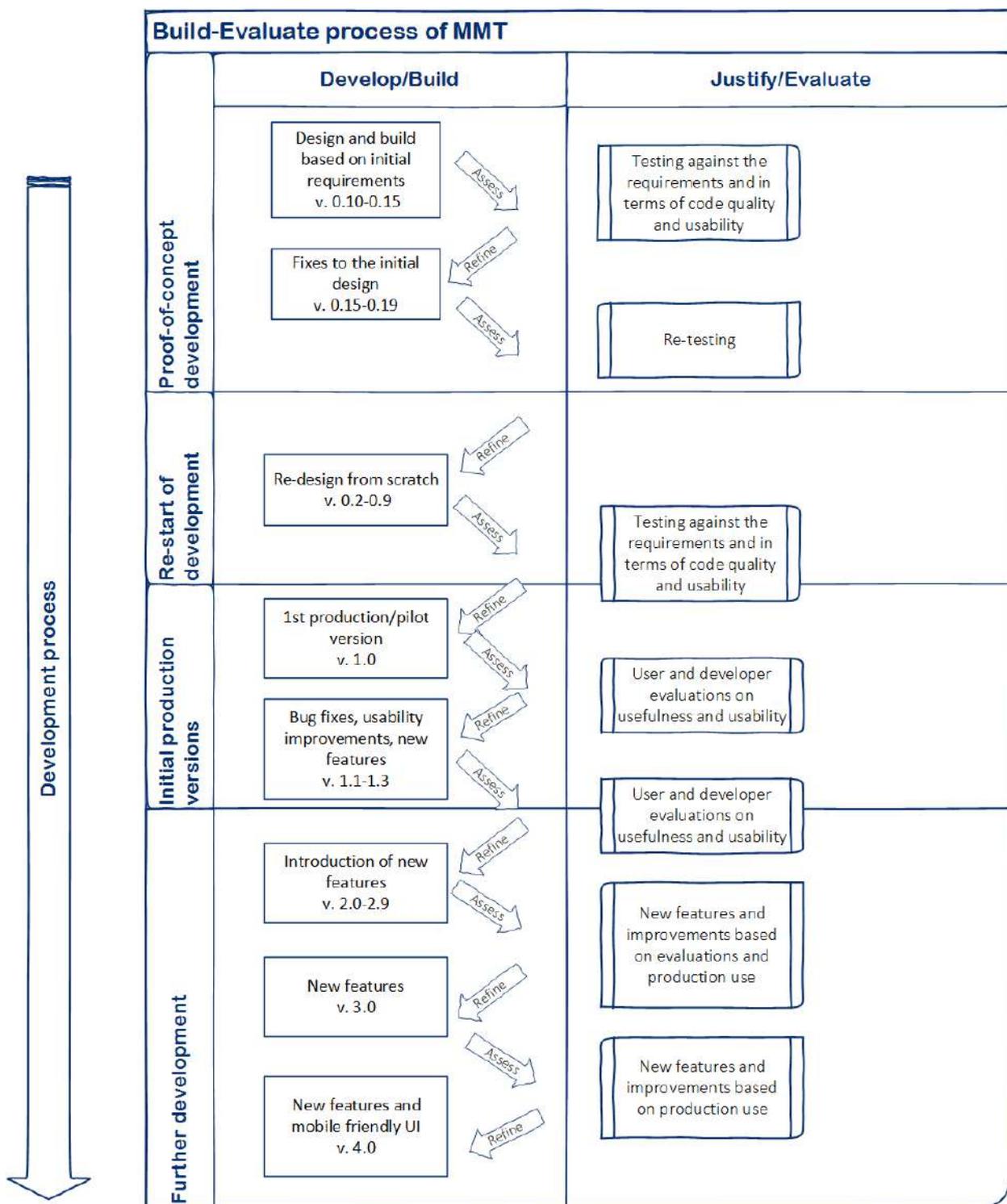


Figure 2. Development process of MMT

Version 0 - proof of concept

The first versions (0.1-0.16) were developed during the academic year 2014-2015. The client evaluated the version by reading the test report and decided that the quality of the software was too low for taking this version to pilot use. The versions 0.17-0.19 were developed during the summer 2015 by one individual member. However, it was also evaluated that the mature of the software was not good enough even for the piloting.

Version 1 - piloting

A new project team was formed at fall 2015. The first task of the team was to evaluate the previous version and the decision was made to begin the developing from the scratch. The version 1.0 was ready by the end of the fall semester and it was piloted on the project work courses during the spring semester 2016. The new versions (1.2-1.3) were also developed parallelly so that the version 1.3 was deployed by the end of the semester.

Version 2 - production use

During the summer 2016 it was evaluated the experience of the piloting and a single student from the team continued the developing by correcting bugs, adding some new features and increasing the usability. The main features of this version were *Logging Hours*, *Viewing Reports*, *Viewing Charts* and *Composing Reports* (Mäkiaho, et al., 2017). This version was on the production use on the Project Work courses at fall semester 2016.

Version 3 - different roles and rights

During the fall semester 2016, the developer of the version 2 continued the developing as a project manager of a new team. The tool was on the production use, it was evaluated and the team developed and deployed new increments. The version 3.0 was deployed by the end of the semester having new features based on different *user rights* and *roles* like project manager, member, supervisor and client.

Version 4 - mobile use and interfaces

During the production use of the version 3 at spring 2017, the tool was still continuously evaluated and feedback was gathered. A single student developed new increments and the version 4.0 was deployed by the end of the semester. The version 4.0 is currently in the production use and it handles new metric: risks. There are also interfaces to common software project tools like Trello and Slack. Moreover, the software was made mobile friendly by adding responsive UI. Example visualisations of number of commits, requirements, test cases, and the working hours by type of the version 4.0 are shown in the Figure 3.

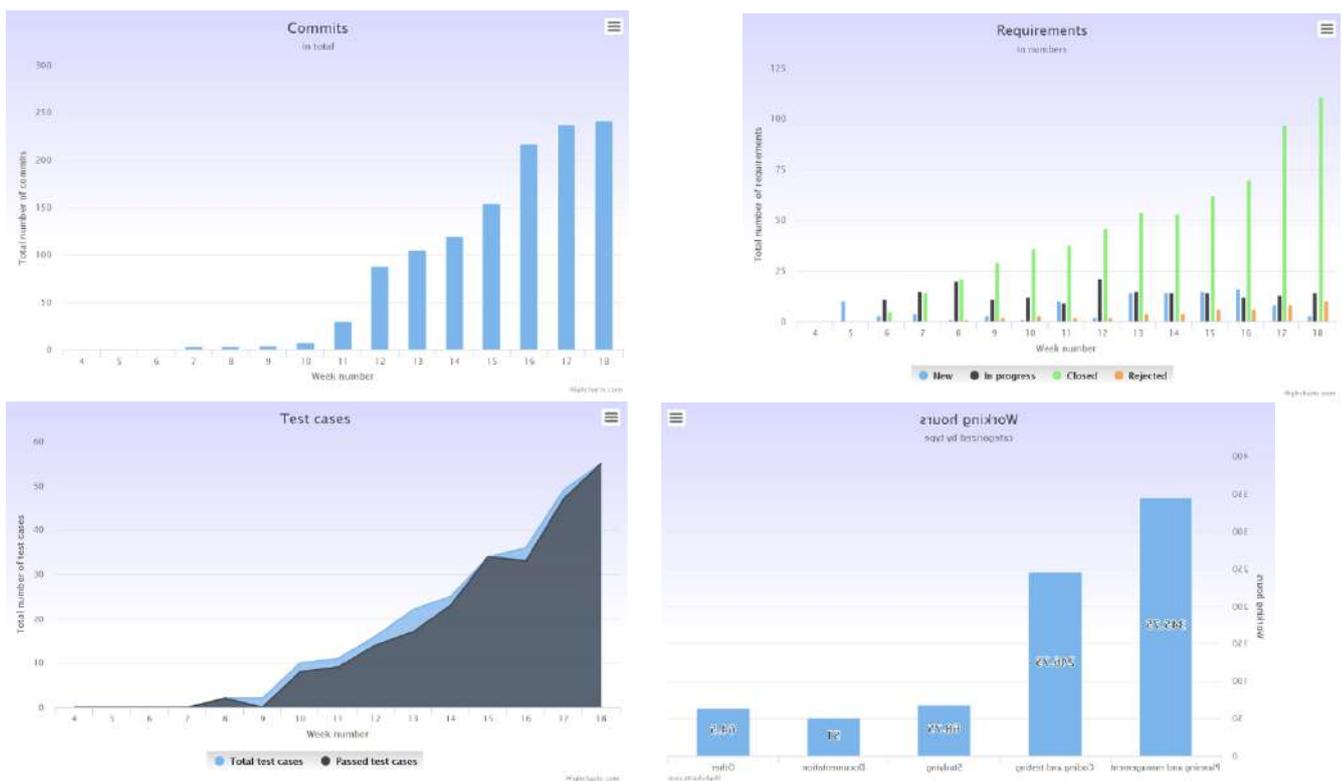


Figure 3. Example visualisations of number of commits, requirements, test cases and working hours

Conclusions

Construction of the MMT has been a long process that started in 2014. There have been roughly 20 people involved in the development of the MMT. Roughly 160 students from 30 projects have so far used the application. There have been also projects clients and teachers using the application. Experiences on constructing and using the application have been positive among students, teachers and project clients.

MMT is a software artifact that solves project reporting and monitoring problem. Design has been evaluated in dozens of software development projects. Future plans for further developing the MMT include enhancing user experience and adding a possibility to archive projects.

References

- Alesandrini, K. & Larson, L. (2002) Teachers bridge to constructivism. *The Clearing House*, 119-121.
- Hevner, A., March, S., Park, J., & Ram, S. (2004) Design Science in Information Systems Research. *Management Information Systems Quarterly*, 28(1), 75–105.
- Mäkiahho, P., Vartiainen, K., and Poranen, T. (2017) MMT - A Tool for Observing Metrics in Software Projects, *International Journal of Human Capital and Information Technology Professionals* 8(4), 2017.
- Papert, S. and Harel, I. (1991) *Constructionism. Situating constructionism*. Ablex Publishing Corporation, 193–206.
- Sommerville, I. (2010) *Software Engineering*, 9th edition, Pearson.

Exploration of Algorithm Abstraction Process with Cubetto and Middle Grade Elementary Kids

Yoshiaki Matsuzawa, *matsuzawa@si.aoyama.ac.jp*

Misako Noguchi, *a8115165@aoyama.jp*

Issei Nakano, *a8115156@aoyama.jp*

Aoyama Gakuin University, Japan

Abstract

In this research, the question of how lower grade elementary school students can solve a programming task which requires algorithm abstraction, was examined using the programming kit "Cubetto" designed for kids. In order to scaffold kids thinking abstraction, the following tools were introduced: (1) the "human-size robot simulator" where kids can simulate their algorithm using their embodied knowledge, and (2) the "Whiteboarding" method where kids can think of an abstraction by finding patterns in a visualized concrete algorithm. In-depth qualitative analysis conducted for the two task-solving episodes illustrates the contrast of the procedures: the one where a young kid could maintain high concentration over thirty minutes with the prepared scaffolds, whereas another where the kids lost interest in their haphazard trial-and-error process.

Keywords

computational thinking; algorithm abstraction; cubetto; lower grade elementary kids

Introduction

At Montessori school, distinct educational practices have been carried out over hundred years with the aim of fostering a love of learning. One of the insights of Montessori is the discovery of what children learn a lot through play (Montessori, 1936). The Montessori toys designed for kids are attractive enough for students to maintain a high concentration, and while kids "play" (called "work" in the school) with the toys, they can learn embedded scientific concepts deeply beyond the developmental stage.

"Cubetto" is a programming kit which was designed with being inspired by the Montessori toys. The wooden, lovely face robot can be controlled by physical programming blocks. The kit, itself, highly abstracts the concept of programming in children's playing activities. In this study, we developed a learning environment with Cubetto, and practiced it with elementary school kids. The main theme is to clarify how children learn the concept of abstraction, which is the core of Computational Thinking (Wing, 2006) through their experiences in this learning environment.

Method

Cubetto

"Cubetto" is a programming kit in which a user controls a wooden robot by the instructions defined by programming blocks on the board. A user can use four types of blocks. The green block works as "move forward," the yellow block works as "turn left," and the red block works as "turn right." Thus, the robot can move around on a map which is made of 6 x 6 tiles, on each tile has a figure which characterizes the tile place (mountain, tree, castle, etc)(Figure 1). A blue block works as a function call: the block calls the operation which is defined using other blocks at the bottom part of the programming board.

The Task

We designed a curriculum to develop skills of playing Cubetto for early elementary kids. The curriculum is composed of some tasks that are categorized into five levels. Students are expected to start from the first level which is comprised of simple, easy tasks and then go up to the upper level comprised of more complex, difficult tasks.

The task which used in the analysis, described in the result section, is categorized as level 5 (most difficult). The task is described as “Start from compass, and pass through tree, mountain, and finally get ship.” The trajectory of a robot expected as an answer of this task is shown in Figure 1.



Figure 1. Robot(Cubetto), Board, and Map, and The Trajectory of Robot as an answer of the task.

The task requires algorithm abstraction due to the limitation of the number of blocks. The number of cells which a robot has to pass through is nine, whereas the number of green blocks (move forward) is four. In order to solve the task, students have to find a pattern to repeat, and implement it using function definition and calling (by blue blocks). The abstraction is one of the keys in Computational Thinking (Wing, 2006); accordingly, we thought this would be an appropriate task for use as a benchmark to explore the learning process.

Tools to Scaffold Thinking

We developed tools to scaffold children’s thinking to solve the designed tasks. The developed tools are shown in Figure 2.

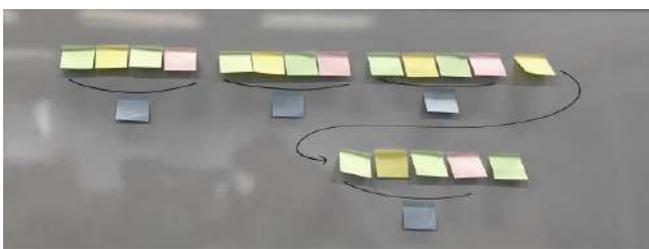


Figure 2. Scaffolding Tools for Thinking (left: human-size robot simulator, right: whiteboarding)

The “Human-sized robot simulator” (hereinafter called “Self Simulator”) is shown in the left of Figure 2. It includes the same map as the moving robot, but it is scaled up to the children's size. The aim of the tool is to think of an algorithm by simulating it by they themselves becoming a robot, so that the children can use their embodied knowledge in their thinking. “Whiteboarding,” a sub tool used with “Self Simulator,” is shown in the right of Figure 2. It includes coloured sticky notes aimed to support thinking program abstraction by temporarily avoiding constraints of the number of blocks. Learners can at once illustrate concrete algorithm without the constraints, which enables them to easily discover the patterns of a block sequence.

Empirical Study Field and Analysis Method

The educational environment was examined at the workshop for kids. Approximately 30 elementary school kids participated over two days. University students participated as a teaching assistant for each elementary student. Learners' learning process in practice was recorded by a camcorder, and two episodes were selected for in-depth qualitative analysis.

Results

The timeline of the selected two episodes in this study are illustrated in Figure 3. In the figure, a star indicates a notable event described in a script, white circles indicate trial points using "Self Simulator," and black circles indicate execution points with the Cubetto program.

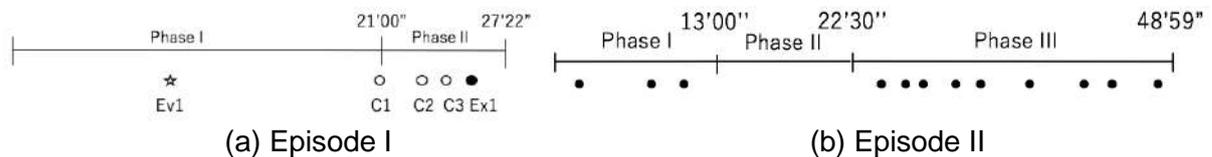


Figure 3. The Timelines of the two episodes analysed in this study

Episode I

The learner of Episode I is a boy (hereinafter called Kid A) who is estimated to be a mid-grade elementary school student. Before tackling this task, Kid A solved 9 tasks in about 30 minutes. As shown in Figure 3, the process is roughly divided into two phases. Phase I is the 21-minute phase of thinking without running Cubetto. Phase II is a 7-minute phase in which Cubetto is examined using "Self Simulator," and finally succeeded to implement a workable program with Cubetto.

Three patterns of actions by Kid A were observed in Phase I. They include "Thinking without moving", "Thinking while moving your finger in the air", and "Thinking while actually getting up and confirming the direction with your body". No blocks had been put on the board until Ev1 (09 '17"). After Ev1, Kid A started putting blocks on the board, but it had not been run.

Kid A was concentrating on thinking by himself. For example, although an instructor advised, "Why do not you think about using simulator?" (06'40"), or his mother advised, "It may be difficult to think with your head, it might be better to try it by yourself "(11'20"), this advice was ignored and he kept thinking in his own way.

Phase II has started from C1 (21'00"), C1 is the first trial with "Self Simulator." Unfortunately, there was a bug in the program and it did not work correctly. C2 (23'06") is a scene where the second trial by correcting the bug found in C1. Although it did not work correctly, Kid A seemed to get some clue, as he said "better, almost." Although his father tried to give advice, he refused it by saying "I'm thinking myself, do not say anything".

C3 (24'13") is a scene where he corrected the bug found in C2. During the trial, Kid A suddenly hit his hands and said, "I should use a (particular) block here." He seemed to come up with something[sL1]. After putting a yellow block to the board, Kid A raised his voice saying "Yeah, I made it!" Even if he had not run yet at this point, he seemed to have gained strong confidence with his success.

Ex1 (25'12 ") was the first time to run Cubetto, and surprisingly it worked correctly. Kid A was watching the execution with a pose of praying. When Cubetto approached the destination, a smile came. Finally, Cubetto arrived, he screamed "Ya-ha!" with a big gut.

Episode II

The learners of Episode II were two girls (hereinafter called Kids B, and C) and three boys. They were estimated to be mid-grade elementary school students. The durations were 48 minutes and 59 seconds. As shown in Figure 3(b), the process was divided into three phases.

Phase I is the first “attack” to the task by all five kids. During three runs of Cubetto during this phase, students became aware that the number of blocks was short for the solution by a concrete algorithm. They were also aware that they must use blue blocks. However, they failed to solve it within 13 minutes.

As they felt a difficulty in the task during Phase I, they went back to solve lower level tasks. Phase II was about 10 minutes where the kids tried the lower level tasks. They succeeded to solve several tasks, but they became tired as kid C commented during the phase.

C: “I got tired playing with (Cubetto). But Cubetto is cute.”

Phase III was about 25 minutes where kids tried the initial task again. However, three boys dropped out, and the only two girls (B and C) participated in this trial. The girls finally succeeded to solve after eight times execution [sL2]of Cubetto, without the “Self Simulator.” They had already understood they should use blue block; however, the patterns they tried seemed to be at random. The phenomena they were getting boring could be observed as the following conversation implies:

B: “I wish Cubetto could run faster”

C: “I wish we had a button that ran faster. I hate this kind of slow system.”

After this conversation, C seemed to neglect participating in the thinking process. She tried to check the next Cubetto booth (there were two Cubetto booths), and tried to copy the next program. B continued to try alone and finally found a solution. Although she had a happy face, she looked tired and she did not look like she found something deeply.

Discussion

In both two Episodes, the children could reach a workable solution. Algorithm abstraction is known as requiring a high level cognitive operation for kids [at this age[sL3]]. One notable thing is that Kid A in Episode I maintained a high concentration over 30 minutes on the cognitive operation and achieved the solution. The episode indicates to us that constraints are the mother of invention (Stokes, 2005).

However, contrasting the two episodes clearly shows us the importance of supporting their tinkering process. In Episode II, the kids lost interest in their haphazard trial-and-error process, and a single kid barely achieved the solution on the sacrifice of the four drop-outs. Kid A ran the program with a confidence in his solution, whereas kid B found a solution almost by accident through her trial-and-error process. The difference of deepness in their level of understanding is obvious. We believe the experience of kid A merely fostered “a greater sense of empowerment” (Papert, 1987).

References

- Montessori, M. (1936): “The Secret of Childhood”, (Mass Market version, Ballantine Books, 1982).
- Wing, J.M. (2006): Computational Thinking, Communications of the ACM, Vol.49, No.3, pp.33–35.
- Stokes, P. (2005): “Creativity from Constraints: The Psychology of Breakthrough 1st Edition”, Springer.
- Papert, S. (1987): “A Critique of Technocentrism in Thinking About the School of the Future” talk presented at Children in an Information Age: Opportunities for Creativity, Innovation, and New Activities.

Influence of Students' Self-perceived Use of Metacognitive Strategies and Sensory Preferences on Academic Achievement in Science and Technology

Enric Ortega Torres, *eortega@florida-uni.es*
Florida Universitària, Valencia, Spain

Vincent Sanjosé López,
Joan-Josep Solaz Portolés
Universitat de Valencia, Spain

Abstract

The present work gives empirical evidence of the influence of Secondary students' self-perceived use of learning strategies (including self-efficacy beliefs) and sensory preferences on academic achievement in Science & Technology subjects. MLSQ scales and VARK scores were used as predictors of the average academic marks. A significant percentage of the variance was explained, with a specific contribution from strategies with a metacognitive component.

Our data suggested that about 17% of the variance in academic achievement could be explained by the students' perceptions on their use of learning strategies and individual sensory preferences. When only the strategies with metacognitive basis were considered, the explained variance reached 16%. In particular, main contributions were obtained from Self-Efficacy and Metacognitive Regulation strategies, and also from the Kinaesthetic score from VARK questionnaire.

Keywords

science and technology education; MSLQ learning strategies, VARK sensory preferences, secondary students, academic achievement

Introduction

In Science and Technology academic subjects (S&T onwards), the personal construction of meaning is a fundamental learning process, so meaning construction should be related to academic success in S&T. As Ausubel exposed in *The Psychology of Meaningful Verbal Learning* (1963), that the personal construction of meaning has some prerequisites of different nature: 1) Well-organized, relevant prior knowledge student's structures (cognitive prerequisite); 2) Conceptually clear learning materials, with a potential logical meaning (logistic-instructional prerequisite); 3) A learner's emotional commitment to make the necessary effort to integrate the new information with his/her prior knowledge (emotional prerequisite). Constructivism also emphasized the student's self-implication in learning because he/she is considered as "active meaning builder". Therefore, when the instructional materials provided to students are not well-adjusted to their cognitive capabilities and emotional dispositions, constructional learning could be impeded. Thus, knowing students' preferences on particular learning materials, or on combinations of different types of learning materials, it could be useful to adapt these materials to students.

Among the varied features in learning materials influencing students' preferences, the format exciting particular sensory channels to capture and process information seems to be of main interest. In fact, Mayer (2005) developed the multimedia learning theory on the basis that human brain uses different cognitive resources to process incoming information with different physical support (images, sounds, text, textures, etc.). By means of associative processes, the brain relates these inputs and then builds a mental representation. Therefore, appropriate combinations of formats for the information provided could increase students' comprehension. Students' preferences for particular formats could be

associated with the easiness of processing and comprehension. In addition to the cognitive, affective and logistic-instructional prerequisites, metacognition has proved to be another influencing factor in academic success. Consistent evidence has been obtained on the academic benefits of a proper development of metacognitive skills (Wang, Haertel and Walberg, 1993). Developing metacognition implies making students' aware of their own strengths and weaknesses, and also the possibility of transferring responsibility from teachers to students themselves to make decisions on learning processes. This is also the claim of the Constructivist approach to learning and teaching (Dewey, 2007): students have to engage in planning, monitoring and evaluating the learning goals, processes and outcomes (Brown and Palincsar, 1982) in order to be active learners and meaning-constructors. Thus, developing metacognitive skills should imply greater academic success, especially in S&T: the consciousness of the internal construction of meaning allows the early detection of discrepancies and incoherencies, and fosters the student's active use of resources at hand to overcome perceived learning obstacles.

Methodology

A total of 365 male and female high school students in grades 7th to 11th participated, belonging to 8 high schools of different ownership located in the surroundings of a big Spanish city.

The instrument called Motivated Strategies for Learning Questionnaire (MSLQ, Pintrich, Smith, Garcia, and McKeachie, 1993) was used to obtain the students' self-perceived use of metacognitive strategies in S&T learning.

The VARK questionnaire (Fleming and Mills, 1992) was used to assign one of the 15 different types of SP to each participant: V: Visual; A: Auditory; R: Read/ Write; K: Kinaesthetic, or combinations of these "pure" preferences (i.e. VK, AR, ARK, etc.).

The academic achievement of each participant was obtained directly from the science and technology teachers of each secondary school. We used a scale of 4 levels, whose relationship with the usual 0-10 scale in Spain was chosen as follows: D: up to 5,0 points; C: 5,5-6,5; B: 7,0-8,5; A: 9,0-10,0.

Results

Influence of students self-perceived use of strategies on academic achievement in S&T

The scales included on MSLQ and the five scores considered in the VARK-questionnaire appeared to be independent, as any of their cross correlations were significant ($r < 0,093$; $p > 0,089$ in any case). Hence, MSLQ and VARK might have specific, unique contributions to the students' achievement in S&T. Together, all scales of the MSLQ explained almost 13% of the students' general achievement in S&T ($F(15,341) = 3,240$; $p < 0,001$; $R = 0,35$). However, only a part of the scales involved in this instrument had significant correlations with the achievement as shown in Table 1. These scales with significant correlations together accounted for 11% of the variance of general academic achievement in S&T ($F(10,346) = 4,081$; $p < 0,001$; $R = 0,33$).

Table 1. MSLQ scales with significant correlations with achievement in S&T

MSLQ scales	r-Pearson (p)
MOTIVATION	
Extrinsic Goal Orientation: EO	0,12 (0,013)
Task Value: TV	0,17 (0,000)
Self-Efficacy: SE	0,26 (<0,001)
COGNITIVE & METACOGNITIVE	
Rehearsal: R	0,20 (<0,001)
Elaboration: E	0,15 (0,005)
Organization: O	0,15 (0,005)
Metacognitive Regulation: ME	0,19 (<0,001)
MANAGEMENT	
Time and study environment: TE	0,19 (<0,001)
Effort regulation: ER	0,18 (0,001)
Support of others (or Help seeking): SO	0,14 (0,009)

As for the VARK concerns, four of its five scores had significant correlations with the students' achievement in S&T (Table 2): the V-score reached only a marginal signification.

Table 2. Correlations between VARK scores and students' achievement in S&T

VARK scores	r-Pearson (p)
V	0,08 (0,082)
A	0,11 (0,014)
R	0,11 (0,015)
K	0,16 (<0,001)
Total Responses	0,20 (<0,001)

When the VARK scores were added to the MSLQ scales, this extended set of predictors explained 17% of the variance of the academic achievement in S&T (Method: *Intro*; $F(13,313)=4,772$; $p>0,001$; $R=0,41$). Using the *back-step* method to keep only the significant contributions, the linear regression explained 15% of the variance of the dependent variable ($F(5,321)=11,494$; $p<0,001$; $R=0,39$), with a sub set of remaining predictors: Self-Efficacy ($\beta=0,25$); K-score ($\beta=0,17$); Support of Others (or Help-Seeking; $\beta=0,13$); Organization ($\beta=0,11$); R-score ($\beta=0,09$).

Importance of the metacognitive strategies on students' achievement

Some MSLQ scales have metacognitive essence, when this set of scales was taken as a predictor of academic achievement in S&T, the explained variance was also 11% (*Intro* method), A very close result to the one explained by all the MSLQ scales together. Using the *back-step* method to compute the linear regression, only GO, SE; CT, ME, SO remained as significant predictors explaining again 11% of the variance of the academic achievement in S&T ($F(5,351)=8,467$; $p<0,001$; $R=0,33$), being SE the most influent predictor ($\beta=0,33$), followed by ME ($\beta=0,14$) and SO (or Help-Seeking; $\beta=0,11$). A detailed inspection showed that the negative contributions of GO ($\beta=-0,16$) and CT ($\beta=-0,11$) were due to fine-tuning compensatory adjustment in the regression, but not to negative correlations with the achievement in S&T.

Conclusion and Discussion

The main purpose of the present study was to shed light on the importance of the students' use of learning strategies and sensory preferences to improve their academic achievement in S&T. Our data suggested that about 17% of the variance in academic achievement could be explained by the students' perceptions of their use of learning strategies and individual sensory preferences. When only the strategies with metacognitive basis were considered, the explained variance reached 16%. In particular, main contributions were obtained from Self-Efficacy ($\beta= 0,26$), and Metacognitive Regulation ($\beta= 0,16$) strategies, and also from the Kinesthetic score ($\beta= 0,16$) of VARK questionnaire. It seems that instructing students to properly use metacognitive learning strategies and to be aware of their individual preferences to process instructional materials is a promising way to improve S&T achievement. A better achievement, together with a greater responsibility in managing the own learning should increase students' self-efficacy beliefs.

References

- Ausubel, D. P. (1963). The psychology of meaningful verbal learning.
- Brown, A. L., & Palincsar, A. S. (1982). Inducing strategic learning from texts by means of informed, self-control training. Center for the Study of Reading Technical Report; no. 262.
- Dewey, J. (2007). Essays in experimental logic. SIU Press.
- Fleming, N. D., & Mills, C. (1992). Not another inventory, rather a catalyst for reflection. To improve the academy, 11(1), 137-155.
- Mayer, R. E. (Ed.). (2005). The Cambridge handbook of multimedia learning. Cambridge university press.
- Pintrich, P. R., Smith, D. A., Garcia, T. and McKeachie, W. J. (1993). Reliability and predictive validity of the Motivated Strategies for Learning Questionnaire (MSLQ). Educational and psychological measurement, 53(3), 801-813.
- Wang, M. C., Haertel, G. D., and Walberg, H. J. (1993). Toward a knowledge base for school learning. Review of educational research, 63(3), 249-294.

Modeling Across the Subjects

Barbara Sabitzer, barbara.sabitzer@jku.at
Johannes Kepler Universität Linz, Austria

Abstract

Jeannette Wing defines computational thinking as fundamental *skill* for everyone. We postulate that modeling is a fundamental *tool* for everyone for constructing knowledge and, certainly, for teaching and learning computational thinking. This is the basis for our new project “Modeling across the Subjects” with two main aims: (1) integrating computational thinking as transversal theme in primary, secondary and teacher education and (2) supporting learning and teaching in different subjects through modeling techniques from the field of computer science (e.g. UML diagrams). This contribution presents the project, its research focus, and some preliminary results from interviews with teachers and students.

Keywords

modeling; computational thinking; learning strategies; brain-based learning

Introduction

With the implementation of the new Austrian curriculum “Basic Digital Education” in autumn 2018 computational thinking becomes obligatory for all secondary schools (grades 5-8), be it as specific subject or integrated in different subjects. However, many schools do not have teachers with computer science background and will teach computational thinking as transversal theme. With our project “Modeling across the subjects” we want to help these schools and show them a practicable way for a useful and “easy” integration of computational thinking in different subjects.

Modeling or building models, which are abstract descriptions of real or planned systems (Hubwieser, Mühling, & Aiglstorfer, 2013), is well-known and important in the field of computer science and informatics education. According to (Diethelm, 2007), models even should be taught “strictly first” (before programming). In the Bavarian informatics curriculum for secondary schools modeling is a crucial topic. From the point of view of general education we strongly agree with Hubwieser, who was substantially involved in its implementation and articulates “that among all themes of informatics it is object oriented modelling that promises the most benefit for the students” (Hubwieser, 2006).

Certainly, not only computer science but also some other subjects like mathematics or geography have their own modeling techniques, or use e.g. the well-known flow charts like economy. In a wider sense, modeling is used and applied, often implicitly and unconsciously, also in all other subjects and daily life e.g. in the case of mental, verbal and physical modeling or also in form of mathematical formulas. For persons without computer science background modeling with diagrams can be compared to concept mapping, which is already used and investigated as effective teaching and learning strategy (Chang, Sung, & Chen, 2002; Horton et al., 1993; Novak, 1990). However, the use of computer science modeling, e.g. with UML (Unified Modeling Language), in non-related subjects is quite new. So far, there is not much relevant literature except few papers that propose some possibilities of application, e.g. the application of the UML for analysis of dramatic words (Tagliati & Caloro, 2008) or the use of flow charts (Al-Fedaghi, 2013) and other models (Kosara & Mackinlay, 2013) in storytelling.

In our project “Modeling across the subjects” we focus on modeling techniques from the field of computer science (e.g. UML-diagrams) and apply them as “brain-based” and constructivist learning strategy and/or tool in other subjects in order to construct and elaborate subject-specific knowledge (e.g. learning vocabulary in a foreign language by means of class or entity-relationship diagrams). In a wider sense concept maps may be compared to the diagrams we want to use in this project. However, modeling techniques and diagrams from the field of computer science have much more potential and possibilities because they can visualize not only structures and relationships but also processes, instructions, situations, events, etc. and train competences like abstraction, problem solving, algorithmic thinking, text comprehension, creativity etc., in short, they train many 21st century skills. In this way we

“kill two birds with one stone” and can teach computational thinking and problem solving implicitly in different subjects. At the same time, students use the diagrams as so called graphic, visual or advanced organizers, which support the brain in the learning and memory process. Additionally, such visual organizers are particularly effective with students who have learning disabilities (Kim, Vaughn, Wanzek, & Wei, 2004). And in our opinion they are ideal for constructing knowledge in different subjects, too.

Modeling across the subjects

The project “Modeling across the subjects” started in March 2018 and aims at

- 1) integrating computational thinking as transversal theme in primary and secondary education and
- 2) introducing appropriate modeling techniques from the field of computer science in order to support and improve learning and teaching in different subjects.

Based on different primary and secondary school curricula we will develop in the next three years

- a framework for “Modeling across the subjects”,
- guidelines and useful hints for teachers without computer science background and
- varied cross-curricular and/or multidisciplinary teaching materials suitable for all subjects.

During this three-year-project we hold modeling workshops for (prospective) teachers of primary and secondary schools, who want respectively have to teach computational thinking as transversal theme. In these workshops all participants are supposed to develop teaching materials based on different sample models for different purposes and to introduce them to their pupils as tools especially for structuring and elaborating subject-specific knowledge, describing activities, planning and summarizing texts, learning vocabulary, etc. Some of the introduced sample models and activities were developed in a former project of the author, where we could already gain positive experiences with flow charts, entity-relationship and activity diagrams (Sabitzer & Pasterk, 2015). In the current project we develop and introduce further UML diagrams like use case or communication diagrams and study the acceptance and usability as well as possible effects on learning outcomes. All materials will be collected, adapted and finally provided in an online collection.

We will investigate the following main research questions by using a mixed-methods design:

- (1) How and where can we introduce modeling in primary and secondary schools in order to teach computational thinking?
- (2) Which diagrams are useful and practicable for teachers and pupils without informatics background in order to construct subject-specific knowledge and to train subject-specific and general skills like abstraction, problem solving or text comprehension etc.?
- (3) To what extent shall modeling and computational thinking be taught to everyone and what shall be part of computer science education?

Our previous experiences show that modeling is considered as very useful and effective, but concrete sample units and activities as well as empirical studies are still missing. Hence, this project may close a gap with providing teaching materials and studying their use and possible effects on learning outcomes in different schools.

Conclusion

The project “Modeling across the subjects” introduces and investigates the use of computer science modeling techniques as brain-based and constructivist learning tools for different subjects. It further shows a practicable way of teaching computational thinking as transversal theme in primary and secondary education. Regarding our previous positive experiences and the preliminary results we suppose that the frequent and varied use of modeling in different subjects may not only train general learning skills like problem solving or abstraction but also improve subject specific knowledge and competences like vocabulary acquisition or text comprehension in foreign language lessons.

References

- Al-Fedaghi, S. (2013). Schematic representation for storytelling. In *IEEE Int. Conf. Computational Intelligence and Cybernetics*.
- Chang, K.-E., Sung, Y.-T., & Chen, I.-D. (2002). The effect of concept mapping to enhance text comprehension and summarization. *The Journal of Experimental Education*, 71(1), 5–23.
- Diethelm, I. (2007). Strictly models and objects first: Unterrichtskonzept und-methodik für objektorientierte Modellierung im Informatikunterricht. Pro Business.
- Horton, P. B., McConney, A. A., Gallo, M., Woods, A. L., Senn, G. J., & Hamelin, D. (1993). An investigation of the effectiveness of concept mapping as an instructional tool. *Science Education*, 77(1), 95–111.
- Hubwieser, P. (2006). Functions, objects and states: teaching informatics in secondary schools. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 104–116). Springer.
- Hubwieser, P., Mühling, A., & Aiglstorfer, G. (2013). *Fundamente der Informatik: Funktionale, imperative und objektorientierte Sicht, Algorithmen und Datenstrukturen*. Walter de Gruyter GmbH & Co KG.
- Kim, A.-H., Vaughn, S., Wanzek, J., & Wei, S. (2004). Graphic organizers and their effects on the reading comprehension of students with LD: A synthesis of research. *Journal of Learning Disabilities*, 37(2), 105–118.
- Kosara, R., & Mackinlay, J. (2013). Storytelling: The next step for visualization. *Computer*, 46(5), 44–50.
- Novak, J. D. (1990). Concept mapping: A useful tool for science education. *Journal of Research in Science Teaching*, 27(10), 937–949.
- Riley, D. D., & Hunt, K. A. (2014). *Computational Thinking for the Modern Problem Solver*. Taylor & Francis. Retrieved from <https://books.google.at/books?id=7AQNAwAAQBAJ>
- Sabitzer, B., & Pasterk, S. (2015). Modeling: A computer science concept for general education. In *Proceedings - Frontiers in Education Conference, FIE* (Vol. 2014). <https://doi.org/10.1109/FIE.2015.7344062>
- Schwill, A. (n.d.). No Title. Retrieved March 31, 2018, from <http://www.informatikdidaktik.de/didaktik/Forschung/VortragsfolienFundldeenMNU.pdf>
- Tagliati, L. V., & Caloro, C. (2008). UML and Object Oriented Drama. *Journal of Object Technology*, 7(1), 85–101.
- Wing, J. M. (2006). Computational Thinking. *COMMUNICATIONS OF THE ACM March*, 49(3). Retrieved from <http://www.cs.cmu.edu/~wing/publications/Wing06.pdf>

Constructing What? Knowledges of the Powerful, and Powerful Knowledges

Michael Tan, michael.tan@nie.edu.sg

National Institute of Education, Nanyang Technological University, Singapore

Abstract

If we consider constructionism as a pedagogical theory, it can be easy to neglect curriculum considerations—what is it we ought to communicate, and why. One approach to this problem is to be deliberate about the nature of knowledge, and the sociological implications of its differential distribution. The epistemologies of the knowledges of Science, Technology, Engineering, and Mathematics (STEM) may be distinguished by its tendencies towards generalisation (S/M), or its contextual application (T/E). Understanding this distinction may be a key to being clear what is being constructed, and how constructionism may be deployed for particular goals. In this theoretical paper, I describe the foundations for such a project.

Keywords

epistemology; nature of science; nature of engineering; curriculum

A curriculum for making?

While many educational jurisdictions exist with autonomy to develop and assess their own curricula, not all schools have that luxury. Standardised testing by state and regional boards of education can compel teachers to narrow outcomes on the ostensible grounds of social justice and the equity of outcomes for public schooling. It would not take much for teachers to sense the cultural logic of the assessment tail wagging the dog of schooling, and recognise that an optimal method of deploying makerspaces and making activity is as a means to the standard curriculum ends of 'concept acquisition' in Science and Mathematics specifically.

Research attention on making as a learning activity has been on the conceptual gains from making (Davis, Schneider, & Blikstein, 2017), or as means for increasing engagement in the STEM disciplines (Norris, 2014), or in contexts outside of the mainstream school classroom (Sheridan *et al.*, 2014). Notably, in sites such as public makerspaces and science museums, a question that has yet to have a convincing response remains the query by Bevan and associates (2015): "It looks like fun, but what are they learning?" While the pedagogical case for making as instructional activity remains convincing, it would appear that a similarly strong curriculum argument is not present. To be particular, looking beyond the 'workforce development' arguments for STEM (Blikstein & Worsley, 2016), what justifications do we in the making community have that students ought to design and make? What ought students make? Why? Should the Arts (and the Humanities?) be part of the STEM/design and making movement? Why? In what ways should these other disciplines contribute to the project of making, and what roles should each play?

Investigating binaries of knowledge

An important method to thinking about curriculum considers the nature of the knowledge and the sociological outcomes of a differential distribution of knowledge. A key question exists: what knowledge is of most worth, or, as modified by Michael Apple (1979) with a sociological angle:

whose knowledge is of most worth. Implicitly, it is this sociological approach that is being advocated when calls for the inclusion of STEM are accompanied by visions of economic futures. However, such sociological approaches suffer from the problem of arbitrariness—of what exactly constitutes phenomena of interest, what simplifying assumptions are to be made, and what are rules for acceptable explanations. If these high status knowledge claims was the result of social convention among those already in positions of high status, then there is a social justice concern in making sure that as many people as possible can have access to the social construction of scientific knowledge. This is especially

so if we believe in universalism of participation as a central value of the natural sciences (see, e.g., Collins & Evans, 2017).

Such a perspective drives efforts to increase participation of under-represented minorities, and which also criticizes 'Western Modern Science' as a White Men's Science. A useful shorthand to describe the issues at stake may be the conceptual distinction between the knowledges of the powerful, and powerful knowledges (Young, 2008). Here, Young points out the significance of the empirical grounding in a stable knowable reality for some knowledge forms. In other words, there are actually truth claims that are obligatory; reality is truly as we describe it to be; or even if it were a matter of social convention, an exceedingly high fraction of humans, with similar sense organs and cognitive apparatus sensitive to certain ranges of phenomena, would arrive at similar frameworks for understanding and description.

The point of the foregoing, then, is to introduce the concept of the social construction of knowledge, and the arbitrary/obligatory distinction, or more informally, the distinction between knowledges of the powerful, and powerful knowledges. Powerful knowledges are so called because, transculturally, such knowledge claims can confer upon its users access to high status deliberations about the world. Powerful knowledges can transcend the contexts of its generation, and find applicability across space and time, as when 'outdated' mathematical theorems are used in contemporary cryptographic algorithms. On the other hand, in invoking a concept such as the knowledges of the powerful, the reference is to situations where things are so, because "the Chief, the Party, or the Pope says so" (Moore & Muller, 1999, p. 193). While we may at first glance consider such epistemic warranting strategies to be largely obsolete in our enlightened times, it may not take us long to find claims that we accept purely on the grounds of scientific authority. This is not to cast doubt on scientific authority, but to minimally point out that students should learn accepted epistemic warranting strategies as part of training as scientists. At this point, the outlines of a plausible justification for educative value should be visible: it *may* be ideal for students to construct understanding of powerful knowledge for themselves (e.g., science), and not quite so much knowledges of the powerful (e.g. wine appreciation). Tentatively, the distinction lies in the ability of these knowledge forms to transcend their contexts of generation, or, in other words, possess at least an ambition to the nomothetic, generalising tendency that is typical of the natural sciences. Conversely, it can be easy to reject or at least find less valuable idiographic forms of knowledge that concern themselves with the particular implementation details of a particular project. However, it would appear that such a model may be incomplete, in that, given the inherent complexity of many systems in the 'real world', there is at least a practical difficulty in obtaining general principles that can make predictions outside of their contexts of generation. Sociocultural interactions, for instance, are not amenable to law-like generalisations of the form of the natural sciences. Much closer to our concerns, the disciplines of design and engineering present similar challenges; I elaborate on these below.

Natural and Artificial Sciences

While the foregoing may provide suitable justification for the natural sciences and mathematics (considered as the language for describing patterns and variations), the nature of the knowledge of technology and engineering provide challenges to this framework. As the subsection header suggests, technology and engineering provide us knowledge of the sciences of the artificial, in that its core concern is with the artificing of solutions to problems. For this insight, we are all, of course, in the debt of Herbert Simon (1968/1996). Simon helped to define the study of design as an intellectual pursuit, as a key concern of engineering:

The artificial world is centered precisely on this interface between the inner and outer environment; it is concerned with attaining goals by adapting the former to the latter. The proper study of those who are concerned with the artificial is the way in which that adaptation of means to environments is brought about—and central to that is the process of design itself. (p. 113)

There is much contained in this passage that is worthwhile of careful study. In considering the artefact as occupying the interface between the inner and outer environments, it is important not to be too literal; software programs can be engineered into existence, and have an internal code environment and user facing interfaces to interact with the code in goal directed ways. The significance for our purposes here is in the *goal directedness* of the efforts to adapt the inner and outer environments. When discussing

the goal directed nature of efforts of engineering, perhaps an underappreciated perspective is the nature of intention, and its relationship to our knowledge of the natural sciences. While we often believe that (design) solutions become clear when problems are correctly identified, we often fail to understand that a very fundamental gap exists between an accurate apprehension of the problem, and the development of a design goal, an intention for the artefact (see, e.g., Nelson & Stolterman, 2003). This gap is obvious if we were to observe that accurate apprehension is an act of description, and goal setting is one of deriving a prescriptive course of action. David Hume's device of the guillotine reminds us that, at least in ethical considerations, there is no necessary connection between the descriptive and normative. In other words, and pertinent to typical instructional sequences in STEM and makerspace projects, just because we can understand a problem in a particular manner, does not mean that the solution necessarily follows. For instance, just because we discover that a local waterway is being contaminated by industrial pollutants, does not mean that a device that can remove this pollutant is necessarily advised.

From this perspective then, the synthetic disciplines of engineering and technology are distinct from the analytic disciplines of science and mathematics in their foci—while we require accurate analyses of the problem in order to obtain a good description of the reality that we are confronted with, we also require a particular 'wisdom' to be able to develop appropriate goals for the problem at hand. Such wisdom is unfortunately not reducible to law-like statements as in the natural sciences. Perhaps the closest thing may be design principles, which are often obscure and opaque to novices with little exposure to problem scenarios. Even then, exceptions abound, and correct application of these principles requires wisdom that can only be gained through rich experiences, or, as is typical in schools of design, a familiarity with case studies that explore the boundaries of our knowledge. Similarly, the disciplines of the humanities and the philosophical inquiries associated with ethics point us in general directions to our best accumulated wisdom on what ought to be, often statements of taste of powerful groups.

Towards rapprochement

It should be clear that a productive engagement with both poles of this debate is needed, if only so that the compromises are known. We ought to communicate the nature of knowledge in the STEM disciplines accurately, so that we do not misrepresent these knowledges to students—we certainly should not want that they eventually enter weapons engineering firms expecting the fun and games of typical makerspace activities (Banks, 2018). Similarly, if we desire a more ethical engineering practice, we need students to see more clearly the contextualised, idiographic detail that requires thoughtful application of generalised principles, and not merely attempts to force-fit people and nature into amoral, abstract systems of thought. As educators, it is timely indeed to consider the problem of what, exactly, we are constructing. At the very least, students ought to become familiar with the distinctions between knowledges of the powerful, and powerful knowledges: where is it that knowledge claims are based on obligatory relationships with the way the world is, and where it is that arbitrary decisions have been made. Significantly, students should learn the nature of the interests to which such arbitrary decisions have been made, in order that they be able to make these decisions for themselves in future.

References

- Apple, M. W. (2004). *Ideology and curriculum* (3rd ed.). New York: RoutledgeFalmer. (Original work published 1979)
- Banks, D. A. (2018, January 24). Engineered for Dystopia. Retrieved February 8, 2018, from <https://thebaffler.com/latest/engineered-for-dystopia-banks>
- Bevan, B., Gutwill, J. P., Petrich, M., & Wilkinson, K. (2015). Learning Through STEM-Rich Tinkering: Findings From a Jointly Negotiated Research Project Taken Up in Practice. *Science Education*, 99(1), 98–120.
- Blikstein, P., & Worsley, M. (2016). Children are not hackers: Building a culture of powerful ideas, deep learning, and equity in the maker movement. In K. Peppler, E. Halverson, & Y. Kafai (Eds.), *Makeology: Makerspaces as Learning Environments* (pp. 64–80).

Collins, H., & Evans, R. (2017). *Why democracies need science*. Cambridge, UK: John Wiley & Sons.

Davis, R. L., Schneider, B., & Blikstein, P. (2017). Making the invisible visible: A new method for capturing student development in makerspaces. In B. K. K. Smith, M. Borge, E. Mercier, & K. Y.

Y. Lim (Eds.), *Making a Difference: Prioritizing Equity and Access in CSCL, 12th International Conference on Computer Supported Collaborative Learning (CSCL) 2017* (Vol. 1, pp. 175–182). Philadelphia, PA: International Society of the Learning Sciences.

Moore, R., & Muller, J. (1999). The discourse of “voice” and the problem of knowledge and identity in the sociology of education. *British Journal of Sociology of Education*, 20(2), 189–206.

Towards Girls' Self-perception in Technology and Craft: Challenges and Implications

Sawaros Thanapornsanguth, Nathan Holbert Monica Chan

{st2839, holbert, mmc2265}@tc.columbia.edu

Teachers College, Columbia University USA

Abstract

This poster explores findings on elementary girls' perception, confidence, and personal experiences in technology and craft. We look into how they define technology and their exposure to technology as consumers and producers. We also asked questions about their experiences with craft materials and activities. Despite coming from families that encourage and financially support extracurricular activities in STEM field, the majority of girls did not feel confident about their knowledge and skills in technology. They also had a tendency to position their own expertise as inferior to their male family members. When asked about crafts the girls showed high interest and engagement and each identified herself as someone who is good with crafts.

Keywords

girls in technology; technology education; craft; technology confidence

Introduction

A large number of articles show that stereotypes and low expectations of women in math, science, and technology play a major role in women's loss of interest in the field (Cohen et al, 2016; Spencer et al., 1999; Steele, 1999). These stereotype threats also contribute to the notion that they are technologically inferior to their male counterparts (Hyde et. al, 1990; Margolis & Fisher, 2003). These data may contribute to gender disparities in STEM professions. The most recent data from National Center for Women & Information Technology (2015) reveals that women make up only 15% of practicing engineers. When technology is described as being primarily for hackers, programmers, and engineers those who do not identify themselves as such are pushed out (Worsley & Blikstein, 2016). Martinez (2015) suggested educators should be sensitive of their classroom environment as girls can react negatively to surroundings that reflect stereotypical "hacker culture" in making, by rejecting their interest in technology and engineering. Literature has suggested creating an inclusive and supportive learning environment and drawing wider examples of what counts as STEM can also help nurture women's interests in these domains (Hill, Corbett, & Rose, 2010; Intel Corporation, 2014; Margolis & Fisher, 2003)

Digital fluency is a concept that has been discussed by many scholars (Kay, 1991; Jenkins, 2006; Resnick, 2012). The perception of being good at technology maps onto the concept of digital fluency, which has been formalized by the National Research Council (NRC), where it states that "people fluent with information technology are able to express themselves creatively" using their chosen technologies (Barron, 2004). Combining the NRC's definition about digital fluency to Resnick's (2012) stance—that true mastery of technology involves being both a confident producer and consumer of technology, we project that if the girls see themselves as a producer of technology, this change in perception might improve their confidence towards their technical abilities. Specifically targeting girls, scholars such as Buechley have explored using computational textiles to bridge the gap between traditional perceptions of technology and craft (Buechley et al., 2008). Crafting techniques such as sewing has a more feminine orientation traditionally, but combining craft and electronics such as e-textiles to cultivate a stronger producer of technology mindset has indicated positive engagement in females toward electronics and computing (Buechley & Eisenberg, 2008).

There is a paucity of research examining the barriers that impede young girls to identify themselves with high competency in technology. Likewise, little work has been done to explore these perceptions for girls from various backgrounds and experiences. This study is a part of the wider Bots for Tots project (Holbert, 2016). We are engaging young learners from diverse communities to build toys for younger kids in their school. In this paper, we explore how girls from privileged backgrounds who have access to high-tech tools and come from well-supported families feel less confident about their technology competency. In examining this population our research aims to answer the following questions: (1) How do the girls describe their experience with and knowledge of technology? (2) What tools, toys, or activities do the girls consider to be “technology?” (3) How do the girls describe their experience with and knowledge of crafts?

Methods

Population and Site

The data presented here is part of a larger Bots for Tots study on fourth grade students (aged 9-11) at a high-resourced all-girls private school in a suburban area in the North-Eastern United States. According to the school’s website, 71% of the students are white. Forty-one fourth-grade students participated in the study as a part of their Making and Engineering class, a bi-weekly maker education class which ran 45-minutes per session. Out of the 41 girls, 12 were randomly selected to be interviewed. In addition to the many extracurricular activities and programs available in the school, students in 4th grade and beyond have access to an Engineering and Design lab (EDL) which is filled with cutting edge equipment including a CNC machine, laser engraver, multiple 3D printers. This population is unique and interesting as they represent the group of girls who receive the best exposures and opportunities to technology. Many of the girls have parents who are engineers, have the opportunity to take extracurricular STEM courses such as robotics and programming, and have access to many high-tech toys and construction kits. Due to their upbringing, these girls are well suited for success in STEM fields. All names used in this paper are pseudonyms chosen by the girls interviewed.

Data Collection and Analysis

Twelve girls were randomly selected to participate in pre and post interview, before and after the Engineering and Design class. Using a semi-structured cognitive-clinical interview format (Ginsburg, 1997), the fourth-grade girls were asked questions about their 1) electronic toys or devices at home, 2) self-perception of technology, and 3) experience with making and crafts, both for themselves and for other people. We did not define technology or provide any specific examples as we sought to understand what the girls regarded as technology. Upon analyzing interview transcripts, we developed a profile for each girl and grouped them in categories and developed a code scheme (Bogdan & Biklen, 2007). This scheme focused on how the girls self-report their “technology and making confidence.” Examples included phrases such as “Umm not really,” and “My brother like to build [...] but I don’t build.” We also coded and categorized the girls based on their different craft activities and experience. Example codes include, “girls making crafts”, “girls talking about family members”, “girls making by or for themselves.” After grouping data in coding categories, we reviewed them for thematic connections (Seidman, 2013).

Results

Low confidence in technology and male family members as experts

When the twelve girls were asked if they considered themselves as someone who is good with technology in the pre interview, eight girls showed lack of confidence and said they were not. Answers included: “no,” “I don’t know,” “not that good,” “I’m not the expert,” and “not really.” One girl expressed uncertainty, answering “kind of”. Only 3 students indicated that they are good with technology. Out of the 12 girls, 6 girls mentioned male family member such as a dad, brother, or uncle, as someone who is good with technology.

For instance, Betty talked about her extensive experience spending her free time working with electronics and technology. She did not have a phone but she enjoyed assembling and playing with “Meccano” (a sophisticated robotics toy that requires 168 steps to assemble) and building bird houses with her dad who is an engineer. She liked technology and electronics, however, when we asked about her technology competency, she immediately responded “No.” She later shared that she did not feel confident as her dad would not let her do the work, “normally he’s like doing the whole thing.” She often looked at her dad building from behind. Similar to Betty, when Panni was asked about her technology competency, she answered “I don’t know” while shrugging her shoulders. Constantly denying her competence with technology or construction (“I’m not a big, big builder”) she referred to her brother as a person who fitted the description better: “I don’t really build stuff that much. My brother likes to build. If he sometimes gets something to play with, and it doesn’t come all together, he can build it. But I don’t build much.”

The girls also mentioned female family members but not necessarily referring to them as technology savvy. For instance, Aditi told us that if something goes wrong with her electronic devices, she would not go to her mom but will seek help from her dad or brother. Lightbulb said that she could not fix a computer but she often taught her my mom how to use different functions on her phone. Kate was the only girl who referred to a female family member as someone who “very good with technology.” She said that she would seek help and learn from her twin sister if she had difficulties with electronics. Additionally, we also learned about the girls’ experiences and usage of the technology. The most common use of technology was to play games (6 girls), watch videos (5 girls), learn an online lesson or do homework (4 girls), programming (2 girls), make videos (2 girls), build a robot (1 girl).

Nine months later, after participating in the Bots for Tots project, we asked them again in the post-interview about their perception of technology. Five of the students said they were good with technology but 2 out of the 5 expressed uncertainty when probed further about their experiences. For example, Kate switched between positive and negative comments about her technology competency, “I feel like working, trying to figure something out either on a phone or computer, I’m pretty good at, I’m not very good. But I think I’m still good.” Similarly, Erica showed a little hesitation and answered “Well, it kind of depends on what kind of technology, but when we were using the circuits during games I think it was great fun.” 7 out of 12 girls said that they were still not good with technology. However, 2 of the 7 girls thought that they were better because they had been building more projects but they still thought that they were not good with technology. For example, Betty said that unlike last year where she was just watching her dad building, he now let her do more hands-on building as she got older. “[My dad] is like teaching me more about [technology] so like I’m learning more from my dad.”

In post interview the girls’ perceptions of what counts as technology shifted somewhat to include programming (5 girls), robots (2 girls), switches (2 girls), fixing and putting things together (2 girls) and sewing machines (1 girl), in addition to including the usual devices such as computers, phones, iPads, and iPods (5 girls)

Confidence and experience in craft

We asked the girls about their confidence in craft and whether they enjoy making craft or not. In both the pre and post interviews, all of the 12 girls said that they enjoyed making crafts. Out of 12 girls, 11 indicated they often shared their craft projects with family members and friends—six of these doing so unprompted. Erica talked about her plan to make a handmade present for her cousin in exchange of the gift he gave her, “recently, my cousin made me a little bicycle for one of my dolls, and so I’m thinking I’m going to make him something.” Moreover, craft was perceived as a family activity for 6 of the girls. For instance, LillyJane talked about gathering random things at her room and make something out of them with her sister and baby sitter. She often made craft when her friends came over to her house. Lightbulb also said that her mom liked to do crafts and together they made small furniture and food out of clay for her dolls. In her free time, Lightbulb also enjoyed watching DIY videos on YouTube with her mom and sometimes by herself.

Additionally, the girls used craft to experiment and express. Kate talked about her experience experimenting with homemade slime putty or Play Doh. Describing her experiment, she said, “Play Doh—you need like flour, salt and water. And then for slime you need like detergent. You can actually

use shaving cream.” She also added that making things out of homemade Play Doh with her female friends was “exciting to me, making it and trying to like discover what could make it and what can’t make it. It’s fun to experiment.” Erica enjoyed making things from cardboard to illustrate the stories she wrote. She made a fairy house, trees, and castle as props to her story. Similarly, Betty made castles and characters that went with it, “I made little people with swords and spears and stuff.”

Conclusions and Implications

The participants of this study came from privileged backgrounds with an abundance of exposure to technology. Their parents and schools actively sought to provide them with powerful STEM opportunities and experiences. However, 8 out of 12 girls still did not consider themselves competent with technology. The girls’ self-report on their technology confidence during interviews aligns with the literature on stereotype threats suggesting that women can feel less confident and significantly underestimate their abilities in the field of math, science, and technology. Moreover, half of the girls interviewed mentioned male family members as technology experts and compared their own expertise to that of the male family members. For example, Panni denied her identity as a builder by suggesting this designation belonged to her brother. Erica also instantly mentioned her uncle as a technology expert when asked about her own experience with technology. This reflects the literature on women feeling technologically inferior to their male counterparts (Hill, Corbett, & Rose, 2010; Hyde et. al, 1990). When we investigate more closely what the girls considered to be technology, we found that programming and mobile devices were mentioned the most. Those activities and tools represent a narrow definition of technology. Furthermore, men are more represented in the professions related to these artifacts and practices, with women holding only 25% of all computing occupations (NCWIT, 2015). Broadening what counts as technology could increase diversity in these fields (Buechley, 2016) as a male-centric articulation of technology likely discourage girls from joining (Buechley, 2013; Margolis and Fisher, 2003).

In contrast to technology, all of the girls expressed confidence and interest in craft and enjoyed making craft as a social activity. While male family members were often associated with the girls’ experience with technology, female family members and friends were a big part of girls’ experiences with crafts. For example, Lightbulb’s mother taught her and her sister to make toy furniture and food out of clay. Margie spent every Tuesday drawing and painting with her female babysitter while LilyJane frequently made crafts when her female friends visited her house. Unlike the girls’ experiences with technology, their experiences with craft falls under the category of the producer rather than the consumer. Girls seem to have a deeper, more meaningful experience producing crafts, compared to playing games and watching videos which was frequently mentioned when asked about their technology usage. The girls also used craft to express themselves, as seen when Erica and Betty created scenes and characters of their stories from cardboards. Taking active roles in making crafts could positively influence girls’ confidence. Additionally, the social implication of making crafts and sharing them with others makes the experience meaningful and enjoyable.

In an effort to improve girls’ confidence in technology, we suggest that future pedagogical methods should first focus on widening what counts as technology. Limiting the definition of technology to male-centric tools and activities may discourage girls from seeing themselves as competent. We should also engage girls more as a producer of technology, as the true mastery of technology involves being both a confident producer and consumer (Resnick, 2012). Lastly, incorporating social implications to the girls’ technological experience makes learning more enjoyable and meaningful.

References

- Barron, B. (2004). Learning ecologies for technological fluency: Gender and experience differences. *Journal of Educational Computing Research*, 31(1), 1-36.
- Bogdan, R. & Biklen, S. (2007). Data analysis & interpretation. In *Qualitative Research for Education: Chapter 5* (5th ed.). Boston: Allyn and Bacon.
- Buechley, L. (2013). *Thinking about making*. Presented at the Fablearn 2013. Retrieved August 9, 2017 from <http://edstream.stanford.edu/Video/Play/883b61dd951d4d3f90abeec65eead2911d>

Buechley, L. (2016). *Inclusive Maker Education: STEM is Everywhere*. Presented at the Fablearn 2016. Retrieved August 9, 2017 from <https://edstream.stanford.edu/Video/Play/a33992cc9fb2496488c1afa9b6204a571d>

Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*, 7(2).

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 423-432). ACM.

Cohen, G. L., Garcia, J., Apfel, N., & Master, A. (2006). Reducing the racial achievement gap: A social-psychological intervention. *science*, 313(5791), 1307-1310.

Ginsburg, H.P. 1997. *Entering the Child's Mind: The Clinical Interview in Psychological Research and Practice*. Cambridge University Press.

Hill, C., Corbett, C., & St Rose, A. (2010). Why so few? Women in Science, Technology, Engineering, and Mathematics. American Association of University Women. Washington, DC

Holbert, N. (2016). Leveraging cultural values and "ways of knowing" to increase diversity in maker activities. *International journal of child-computer interaction*, 9, 33-39.

Hyde, J. S., Fennema, E., Ryan, M., Frost, L. A. and Hopp, C. (1990), Gender comparisons of mathematics attitudes and affect. *Psychology of Women Quarterly*, 14: 299–324.

Intel Corporation, MakeHers Report: Engaging Girls and Women in Technology through Making, Creating, and Inventing, (2014). Retrieved August 20, 2017 from <http://www.intel.com/content/www/us/en/technology-in-education/making-her-future-report.html>

Margolis, J., & Fisher, A. (2003). *Unlocking the Clubhouse: Women in Computing*. MIT Press.

Martinez, S. (2015). Making for All: How to Build an Inclusive Makerspace. Retrieved August 26, 2017 from EdSurge:<https://www.edsurge.com/news/2015-05-10-making-for-all-how-to-build-an-inclusive-makerspace>

National Research Council. (1999). *Being fluent with information technology*. National Academies Press.

NCWIT. (2015). Women in IT: The Facts Infographic. Retrieved August 26, 2016, from <https://www.ncwit.org/resources/women-it-facts-infographic-2015-update>

Resnick, M. (2012). Mother's Day, Warrior Cats, and Digital Fluency: Stories from the Scratch Online Community. In *Proceedings of the Constructionism 2012 Conference: Theory, Practice and Impact* (pp. 52-58).

Seidman, I. E. (2013). Analyzing, interpreting, and sharing interview material. In *Interviewing as qualitative research: A guide for researchers in education and the social sciences* (3rd ed.). New York: Teachers College Press.

Spencer, S. J., Steele, C. M., & Quinn, D. M. (1999). Stereotype threat and women's math performance. *Journal of Experimental Social Psychology*, 35(1), 4–28.

Steele, C. M. (1997). A threat in the air: How stereotypes shape intellectual identity and performance. *American psychologist*, 52(6), 613.

Worsley, M., & Blikstein, P. (2016). Children are not hackers: Building a culture of powerful ideas, deep learning, and equity in the Maker Movement. In *Makeology* (pp. 78-94). Routledge.

Collaborative Creative Music Activity with ICT: A Case Study for Children in Grade Five

Sayaka Tohyama, tohyama@inf.shizuoka.ac.jp

Yugo Takeuchi, takeuchi@inf.shizuoka.ac.jp

Shizuoka University, Japan

Abstract

How to support learning new knowledge without spoiling children's creativity seems a shared question in both constructionism community and STEAM education. We designed a workshop for 5th graders to scaffold learning musical knowledge using Jigsaw and "Vocaloid for Education" as "object-to-think-with". The result was that fourteen children out of twenty succeeded to make harmonies to given melodies using knowledge which they learned in the workshop even though they had no previous experience of Vocaloid, and five out of the fourteen created unique ones.

Keywords

keywords: Music; vocaloid for education; collaborative learning, STEAM

Introduction

Recently STEAM is used in the context of programming or ICT in education instead of STEM. As we know, children can make creative products including "art" essences using Scratch (Resnick, 2017). It seems that they learn about programming and art in their constructional process if we see it from the viewpoint of Papert's "Mathland" (Papert, 1980). Although the effect of constructional activity for educational context is expected, the process of the children's learning about programming and arts is still a hot topic in educational research.

In this study we focused on music as "art" in STEAM context. Music is widely accepted as an enjoyable and creative activity, and it has synergistic power with ICT. One of the previous researches suggested that children have musical sense and they could show their senses (Bamberger, 1991).

Our research question is to know what kind of scaffolds work for children to improve musical knowledge based on their creativity in a sense of Constructionism. To the question, our hypothesis is that collaborative learning with ICT may function for improvement of musical knowledge because collaborative work improves finding new viewpoints and deepening the participants' understandings (Miyake, 1986). To confirm it in a real situation, we used the jigsaw method (Aronson & Patnoe, 1997) and "Vocaloid for Education" which may work as "object-to-think-with".

Methods

We held a three-hours workshop for twenty fifth graders (12 boys and 8 girls) in an elementary school in December 2017. The children were chosen by lot from who hoped to participate this workshop. We posed the question throughout the workshop as "How to make beautiful harmony?" to enhance the children's constructional activity using Vocaloid for Education (Figure.1) in collaboration. The children's main activity was making harmony for a song. The target song was "It's a small world" because it is well known all over the world and in a music textbook for 4th graders in Japanese elementary schools. The venue of the workshop was the PC room in the school, and we specially installed Vocaloid for Education on the PCs. The first author conducted this workshop, and two adults who had experience in supporting children's learning using ICT during the workshop. The children worked in pairs when they created their own harmony, and in three-member groups when they collaborated in the jigsaw method.

The set of activities for the children in the workshop was as follows:

- Answering the questionnaire (preQ), and harmony-making test (pre-test)

- Correcting the disharmony of “Twinkle twinkle little star” using Scratch as introduction
- Answering the harmony-making test (mid-test)
- Planning how to create the harmony for “It’s a small world”
- Learning musical knowledge in the jigsaw method
- Creating harmonies using Vocaloid for Education
- Answering the questionnaire (postQ), and harmony-making test (post-test)

The children answered to each questionnaire and harmony-making test individually. In the questionnaire, we checked the children’s previous experience of music and programming, and attitude to musical creation. We evaluated the pre-, mid- and post-test for comparison of the harmony-making test, and checked the children’s pre-experience using the answered questionnaires. In the three times of harmony-making test, we showed the final four bars of “It’s a small world” and asked the children to write a harmony. We videotaped all the activities and collected the memos which were written by the children, and referred to them if we needed additional information for our analysis.

Scaffolds

Jigsaw Method

The jigsaw method is one collaborative learning method used to enhance the children’s ability for contributing and discussing a given question using handouts which contain knowledge or technique for helping children to generate their answer for the given question. Each child has responsibility to explain his/her assigned handouts to the other members, so they prepare it in collaboration with classmates who have the same handouts. After that, children who has different handouts share the contents of their handouts with each other. They generate their answer to the question through discussion for integration of their handouts.

The jigsaw method could be a scaffolding for constructional activity because the children can try to make relations between their prior knowledge or experience and new knowledge or experience.

In this research, we made three kinds of handouts to give the children musical knowledge. Each handout fits in A4 size and contained some figures to help the children’s understanding. The theme of the handouts is as follows:

- A: Codes and its structuring notes (C is made by C and 3rd notes and 5th notes, and so forth)
- B: How to choose harmonized notes (octave, avoiding 2nd notes, and so forth)
- C: How to make smooth and natural melodies (match with intonation of lyrics, and so forth)

Vocaloid for Education

Vocaloid for Education is a music creating application for Windows, developed by YAMAHA corporation. The target user is elementary to junior-high school children in Japan. Children can create mixed chorus in four voices by putting boxes onto a two-dimensional screen: tone length is horizontal and pitch is vertical. We can also enter lyrics to each melody. The artificial voice sings the set of melodies simultaneously when we push the “play” button, and the progress bar shows where the voice sings. These characteristics allow children to create music who have only little musical knowledge and ability to read and write musical scores.

Results

The harmonies made by all the pairs in the workshop differed from each other, and the harmonies made by individual children in post-test differed slightly from the pair results. From these results, it is suggested that they used given musical knowledge in their own ways, not stereotypically.

Questionnaire

The preQ revealed that nineteen children had no previous experience of Vocaloid for Education and programming. All the children liked musical activity and using ICT in preQ and postQ. Only seven children preferred expression using music over appreciation in preQ, hence twelve children preferred

expression in postQ. The number of children who felt good at reading musical scores was ten in preQ and raised by twelve in postQ.



Figure 1. Vocaloid for Education (Main melody is purple boxes, and harmony is light blue boxes)

Harmony-Making Test

We coded the nineteen children's answers about harmonies (missed collecting the question sheet from one child) of pre-, mid- and post-test into seven categories from the viewpoints of uniqueness, stereotype (rule-based), and harmonization. The categories are as follows:

- Major 3rd: Consecutive intervals which mainly consists of major 3rd of the main melody
- Major 3rd+Arrange: Adds a little unique harmony to Major 3rd
- Unique: A harmony which does not depend on major 3rd of the main melody
- Major 2nd: Dissonant intervals which mainly consists of major 2nd of the main melody
- Major 2nd+Arrange: Adds a little unique disharmony to Major 2nd
- Random: No rules. Randomly assigned notes, spreading notes, and so forth.
- N/A: No answer

Major 3rd, Major 3rd+Arrange and Unique means smoothly harmonized melodies. Major 2nd and Major 2nd+Arrange suggests incongruent melodies. Random is partly harmonized. In these categories, Unique is the most qualified answers because it appears when the children succeed orchestrating the musical knowledge and the creativity from their own viewpoints. Random is the lowest quality because it seems the children did trial and error without plans.

The results were shown in Figure 2. It suggested that the children tended to answer randomly, gave no answer, or made incongruent in pre-test. In mid-test, which is located after experience making harmonies using Scratch, the number of children who used major 3rd notes systematically increased from two to ten. Furthermore, the children who answered randomly disappeared. It suggested that they learned simple rules for harmonization and incongruency in harmony in their Scratch activity. The post-test which is located after the jigsaw method with Vocaloid shows that two children created unique harmonies for the first time. It appears the combination of the jigsaw method and Vocaloid worked for the children's creation of harmonies.

Conclusion

Our case study showed that children could discover and use simple musical knowledge such as 3rd notes from the main melodies' notes only experienced correcting disharmony activity. However,

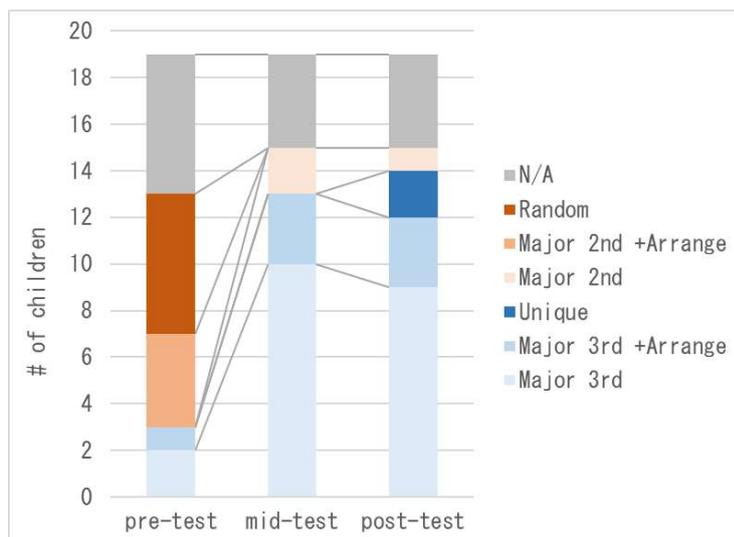


Figure 2. The number of the children's answers in harmony-making test.

to enhance the children's production of unique and qualified outcomes, the jigsaw method with ICT may function. In our study we used the jigsaw method and Vocaloid for Education simultaneously to scaffold the children from the viewpoint of constructionism because collaboration may emphasize the characteristics of "object-to-think-with" in Vocaloid for Education. For the future we will try to evaluate the effect of each one by protocol and physical-performance analysis using video data.

In programming education, the problem seems the same with this study because scaffolding in programming education is expected not to spoil children's creativity, but to enhance learning how to solve problems or disciplinary knowledge in constructional activity. We think finding principles for implementation such kind of activity is one of important themes in this research field. In this paper, we only suggested the effect of designed collaborative activity. For the future, we will try to find what is important principles in collaboration to cause meaningful constructional activities and share the results with STEAM educators.

Acknowledgments

We would like to thank JSPS Kakenhi (No. 17K17786) and foundation of promotion for engineering in Shizuoka University. We wish to thank Junko Tsutsui, Nobuhiro Shirai and Tomoyuki Kuno (Yoshinkita elementary school, at that time) who gave us great help to implement this workshop, and Yukako Enya (Smart Education System group, YAMAHA Corporation) who gave us helpful advice and materials for Vocaloid for Education. The handouts were inspired by the YAMAHA website.

References

- Aronson, E., and Patnoe, S. (1997). *The Jigsaw Classroom (2nd Edition.)*. Longman.
- Bamberger, J. (1995). *The Mind behind the Musical Ear*. Harvard University Press.
- Miyake, N. (1986). Constructive Interaction and the Iterative Process of Understanding. *Cognitive Science*, 10(2), 151-177.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.
- Resnick, M. (2017). *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. The MIT Press.

Applied Constructionism: Critical Reflection and Learning Through Play in Adult Learning

Nalin Tutiyaphuengprasert, nalin@stanford.alumni.edu

Darunsikkhalai School for Innovative Learning, King Mongkut's University of Technology, Thailand

Abstract

This poster discusses the combination of constructionism and critical reflection in a workshop design for adults in Thailand. Thailand applied constructionism in different fields including professional development for employees focusing on facilitating self-development and soft skills. In these situations, learners work together to create complex Rube Goldberg machines in less than 24 hours. Time pressure helped learners to reconnect with others and cope with their negative thoughts. While Papert (1999) stated "This acceptance of "negatives" is very characteristic of the Logo spirit", Mezirow's critical reflection (1990) can support safe expression of "negatives" and put them as an "opportunity" to understand and transform.

Keywords

applied constructionism; critical reflection, adult learning; professional development; agency; team building; metacognition; transformative learning; perspectives transformation

Constructionism Intertwined with Critical Reflection and U Theory

Since 1997, constructionism has been applied in Thailand both in formal education and non-formal education. For adult learning, constructionism has been applied in different professional development programs in businesses, industries and farmers' communities in rural area. "Think-Make-Reflect" is a motto that has been used as a core process in these situations as well as a learning outcome of constructionism in Thailand to foster continuous improvement and self-directed learning.



Figure 1. Rube Goldberg machine and Critical Reflection Workshop in Thailand

Our Rube Goldberg machine workshop was designed to tackle the solo mentality problem in workplaces, boost a more active learning environment and improve soft skills among employees. Companies who participated are from different businesses such as healthcare and service, satellite service providers, food and dairy manufacturing and conglomerate businesses. In this workshop, constructionism learning tools played a main role in allowing learners to express themselves through physical artifact creation. We applied critical reflection (Mezirow, 1990) to help learners when they are facing challenging tasks by facilitating learning from reflection, including facing negative thoughts and emotional challenges. U theory (Scharmer, 2009) was also used as a support for learners to reassess thinking habits and show a path to explore and experiment with new possibilities.



Figure 2. Secret Objects, design of contraptions and brainstorming Session

Explore Your Internal World with Critical Reflection and U Theory

Adults can learn from changing their perspectives from their past experiences. As Mezirow stated “Adulthood is the time for reassessing the assumptions of our formative years that have often resulted in distorted views of reality” (1990). Critical reflection helps adults be aware of their own assumptions about work, life, ability, values, rules, and so on, which impact how they perceive the world and make decisions in life. We applied critical reflection in the workshop to bring participants’ thinking habits or emotions to the surface. Short guided meditation was also used to help learners feel calm, be able to recognize their emotions, and think clearly.

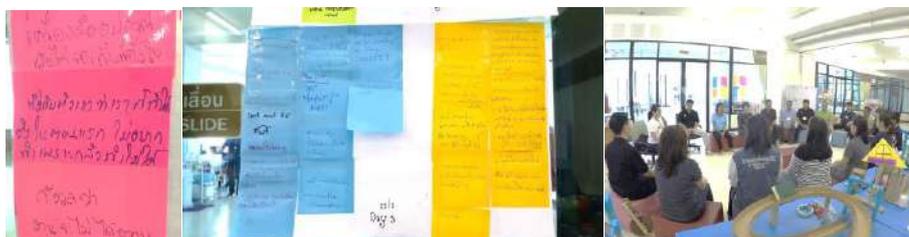


Figure 3. Daily reflection and examples of post-it notes

Translation of reflection in pink notes: “Tools didn’t fit with my expectation.”

“It’s amazing that I can do it which I didn’t want to do this at the beginning because I’m scared.”

“I’m concerned that the task wouldn’t be finished on time.”

In Thai culture, reflection can be useless if we do not take into account cultural sensitivity. It is normal for people to cover up their thinking or say things to please audiences, especially if they perceive hierarchy or seniority in the environment. I tried to build atmosphere of freedom of expression with some core principles: 1) participants choose what, when and how to express 2) critical or negative feedback is welcome 3) participants do not have to speak if they do not want to, and 4) silence is acceptable and a legitimate way to show respect to allow everyone to think, contemplate and decide to when to speak and when to listen.



Figure 4. Making contraptions (left and middle) and reflection time at the end of each day (middle right) and celebration of Rube Goldberg (right)

U theory gives us a point of view of how people respond and decide to react, especially when it is something uncomfortable or related to negative experiences in the past. This model introduced two paths to participants: knee jerk reaction (automatic reaction without thinking) or moving down through

the U shape model to explore our mental stages. Voice of judgement, cynicism and fear were explicitly introduced to learners with an encouragement to observe and refrain from these types of normal reactions. U theory explained the stages through which we can first hold back immediate reactions and detect negative thoughts in our minds, then be fully present, and finally allow the mind to recreate a different reaction without bias or judgements from the past experiences.

Conclusions

Construct and Reconstruct Your Inner World in a Playful Context

The combination of the Rube Goldberg machine challenge and critical reflection provided a playful environment where adult participants learned to recognize and transform their views and behaviour in a friendly way. As Ackermann (2014) stated, playfulness can give space to tolerate uncertainty and set the stage for shifting perspectives or repositioning oneself. Being playful also includes the notion of safe exploration. It provided the right touch of empowering learners to keep asking “what if” questions, giving learners opportunities to suspend their old assumptions and explore their internal thinking and emotional process. This enabled them to change their thinking if they wanted and at their own time.

Thinking and emotions are dynamic and rooted in deeper levels of belief. The critical reflection process helps explicitly capture feeling and thinking and helps learners to develop metacognition processes. Some feedback, which may initially be perceived as negative by adult learners, can be very useful for participants in observing their pattern of reactions, leading to deeper analysis of internal assumptions that they might not have been aware of before.

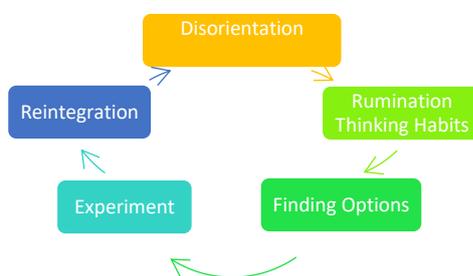


Figure 5. Process of Reconstructing Thinking Habits or Assumptions

Figure 5, shows the process of change when a learner decides to tackle their own negative thoughts. Detecting negativity helps liberate learners from self-deception and be more open to alternatives. We have often found that many learners challenged their fears and transformed their responses by the end of the workshop.

“I usually have phobia with microcontroller. I was shocked when I have to do this task. “Sh**!” I talked to myself when I first looked at the Gogo board package. I had a bad experience in my high school. I asked my friend to do it for me. I can pass those tests because my classmate did it for me. But now I can’t run away. But when I learned about input, process and output, I can jump out from my fear. I can get rid of that fear now and I feel really good about it. This is what I want to share with you guys. Fear is in us and it is going to be one day that we can jump out of it!” (Reflection transcription by participant in the workshop).

People can learn and grow from mistakes and negativity. Creating a safe environment allowed people to transform their misconceptions and heal from past distortions. Constructionism can serve as a safe and playful situation for adult learners to nurture confidence in facing challenges in life and learn from them.

“I made a lot of mistakes and when it was the 20th time, I thought that it was fun to fix problems. I’m ready to solve problems that arise and I learned so much from them and gained more confidence. In the past, I was so scared of making mistakes.

I avoided all kinds of mistakes but now I learned to be open to mistakes.
I promised that I will now dare to make mistakes and learn to fix them quickly.
I promised that I will do mistakes in everything but I will learn and fix them all.”

(Reflection Transcription by participant in the workshop)

References

- Ackermann, E. (2014). Amusement, Delight, Whimsy, and Wit, the Place of Humor in Human Creativity. In *Constructionism 2014 International Conference, Vienna, Austria, August*.
- Mezirow, J. (1990). How critical reflection triggers transformative learning. *Fostering critical reflection in adulthood*, 1, 20.
- Papert, S. (1999). Logo philosophy and implementation. *Logo Computer Systems Inc*.
- Scharmer, C. O. (2009). *Theory U: Learning from the future as it emerges*. Berrett-Koehler Publishers.
- Senge, P. M. (2006). *The fifth discipline: The art and practice of the learning organization*. Broadway Business.

Different Cultures – Different Approaches to Reasoning and Algorithms

Valentina Dagienė, *valentina.dagiene@mii.vu.lt*

Lina Vinikienė, *lina.vinikiene@mii.vu.lt*

Vilnius University, Lithuania

Abstract

The following five strands of mathematical actions are important in school curricula: 1) conceptual understanding; 2) procedural fluency; 3) strategic competence; 4) adaptive reasoning; 5) productive disposition. These strands have implications for mathematics teaching of both the practical and specialised perspectives and are more or less cultural sensitive. One of the challenges facing mathematics educators is to incorporate each of the mathematical strands in a multicultural classroom. We developed the module for teacher educators that provide theoretical background based on the role of reasoning and algorithmic thinking in mathematics. The module involves practical activities based on the understanding of ethnomathematics and Seymour Papert ideas. Through this module prospective teacher are able to investigate the understanding different approaches of reasoning and algorithms, explore examples of different approaches, experiment and reflect on the use of tasks, practice different algorithms, develop pedagogical approaches. The module is prepared under Erasmus+ project "Intercultural learning in mathematics and science education, IncluSMe". The project aims to increase the quality of higher education curricula for prospective teachers by linking maths and science education with intercultural learning (<http://inclusme-project.eu>). The poster describes the module content.

Keywords

Algorithmic thinking, ethnomathematics, intercultural learning, reasoning in mathematics

Background

Teaching mathematics require a deep understanding about previous student's knowledge and that they need to learn. The main challenge is the support them during the learning process. Due to that, teacher educator have to be prepared to show the same things using different approach or instruments. Astuti&Purwoko (2017) emphasize that learning instruments integrated with ethnomathematics involve the construction of mathematical knowledge by instilling positive cultural values. According to authors, model of learning based on culture-related mathematics is based on constructivism learning theory.

Ethnomathematics is associated with mathematics education, social and cultural background. D'Ambrosio stated that in order to understand how mathematics is created, it is necessary to understand the problems that precipitate it. It involves the cultural context that drives them (Rosa&Orey, 2011). Ethnomathematics is described as a branch of mathematics acknowledging the fundamental differences in mathematics content, mathematics understanding, and mathematics application that links culture and mathematics (D'Ambrosio, 1985). According to Rowlands, Carson (2002), Rosa, Orey (2011), teachers should know various teaching methods in order to teach formal, abstract concepts in different cultural background. The ethnomathematics perspective into the mathematics curriculum "help student to develop skills in critical thinking and analysis that can be applied to all areas of life and provide an effective environment for developing skills to solve real-world problems". (Rosa, Orey, 2011, p. 48). In addition, it helps students to achieve better academics results, know more about reality, culture, society. Rosa&Orey (2011) stated that ethnomathematics is a program that includes relevance and builds knowledge around the local interests, needs, and culture of students. According to Saymour Papert, as well as connecting with the formal knowledge of mathematics, it also connects with the "body knowledge".

As a way to accommodate different learning styles and demonstrate several way to solve the same problem, Fisher&Davis (2008) distinguished the importance of algorithms. Algorithms engage students

to learn mathematics in an active, fun environment. According to Kantner (2008), algorithms is steps to solve problems which are not universal across cultural and nations. He mentioned, that learning math affect different types of representation systems, concepts, native language, world views, informal mathematical experiences, conceptual differences in logic, reasoning, cognitive styles. However, teacher educator should present and know not only one algorithm, he/she should be interested in more alternative. In this way, in the cultural diversity classroom confusion and not understanding will be avoided (Perkins, Flores, 2002). Identification of algorithms and description why the algorithm works let students to rethink mathematical ideas. By solving mathematical problems student have to get elementary knowledge about mathematics methods and challenge the belief that this algorithms work (Philipp, 1996).

In this paper, we describe the module that is prepared under Erasmus+ project “Intercultural learning in mathematics and science education, IncluSMe”.

Module for mathematics prospective teachers

Module “Different cultures – different approaches to reasoning and algorithms in mathematics” is design to introduce pre-service teachers to intercultural learning in mathematics using different definitions. This module involve an introduction into reasoning and algorithms in mathematics, theoretical background based on the understanding etnomathematics and Seymour Papert ideas. During the module, we present the connection between theory and educational practice. The module consist for three main parts and one additional section. In addition, we introduce computational thinking as one of the way of mathematical reasoning. The main goal of the module is to engage student to try different problem solutions, share and explain their own ideas. Students discuss about text that present different reasoning methods, study specific examples, explore sources of culture-related contexts, solve problems using different approaches of reasoning and algorithms, compare solutions in pairs or groups, analyse text or identify examples and opportunities to use culture-related context, build knowledge using constructionist approach.

Introduction

This part of module involve discussion and introduction of theoretical background.

First activity. Firstly, students find video about cultural understanding, reasoning and algorithms in learning and teaching mathematics. During the lesson, they present the most interesting video and discuss about their findings in groups, and fill table. Filling the table, they have to describe the meaning of cultural understanding, ethnomathematics, reasoning, and algorithms. The idea of these tasks is not only to engage students share ideas about particular video, but also their experiences. Together students compare understanding and meaning, try to get one common problem solution. At the end of activity, teacher educator sum up discussion and provide the basics of theoretical background: provide the meaning of etnomathematics, reasoning, algorithmic thinking from the literature view.

The second activity introduce Seymour Papert’s book “Mindstorms: Children, Computers, and Powerful Ideas (1980). In this book Seymour Papert argues about the benefits of teaching computer literacy and illustrate many powerful mathematical ideas. Students have to analyze examples of text and discuss about reasons to use culture-related context, the most important characteristics of good approach of reasoning and provide approach for reasoning or algorithms in mathematics using personally. Analysing text student are able to compare constructionism and constructivism, concrete thinking and formal thinking, explain the meaning of assimilation, importance of LOGO, etc. Later, students are asked to analyse the following example (Figure 1a) and apply algorithms in practice.

Difficulties experienced by children are not usually due to deficiencies in their notion of number but in failing to appropriate the relevant algorithms. Learning algorithms can be seen as a process of making, using, and fixing programs. When one adds multidigit numbers one is in fact acting as a computer in carrying through a procedure something like the program in Figure 18.

1. Set out numbers following conventional format.
2. Focus attention on the rightmost column.
3. Add as for single digit numbers.
4. If result < 10 record results.
5. If result in rightmost column was equal to or greater than 10, then record rightmost digit and enter rest in next column to left.
6. Focus attention one column to left.
7. Go to line 3.

Figure 1B

[Seymour Papert's book "Mindstorms: Children, Computers, and Powerful Ideas, 1980, pp. 152]

Figure 1a. Text example

The idea of algorithms practice is that students compare the differences in culture diversity. At the end of lecture, teacher educator provide several example such as arithmetic operations in different countries, different multiplication algorithms, and different calculation methods. These examples should be in different type, such as text, video, practical exercise. In this way, students will be engaged to find differences between reasoning, understandings in different cultures.

Culture-related context (practical reasoning examples)

The second part of module involve practical examples of juggling, mathematical task, and programming. Firstly, student analyse video, than read the text from S. Papert "Mindstorms" (p. 105-112) and practice juggling by using pieces of light material.

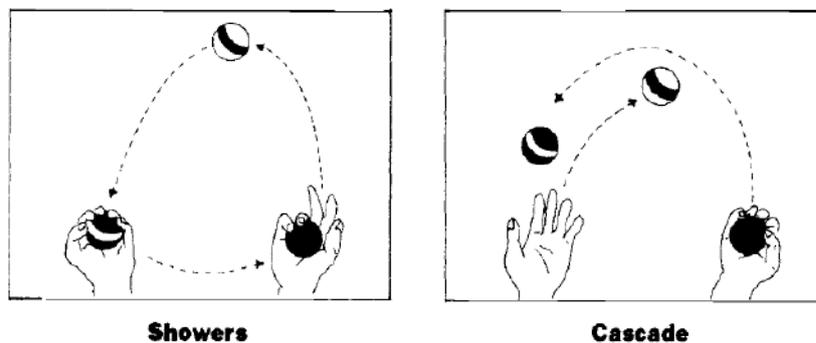


Figure 1b. S. Papert. MINDSTORMS. Juggling (p. 105-112)

The goal of this activity is to compare how students understand the same things in different ways and compare each student's experience. Not all student will be successful in juggling, due to that other students should help and explain how to reach the goal of task. This activity woke up discussion between students how to do the tasks step-by-step using various methods, such us juggling with one ball or more, juggling with a scarf. Presenting steps of juggling on the sheet of paper students describe the algorithm how to find the best solution.

The second activity is the analysis of the example "A string around the circumference of the earth" from S. Papert. MINDSTORMS... Students discuss about text and ways of reasoning.

The third activity could be implemented in a computer lab using programing language as *Scratch*. Using programming language students have to draw triangle, square, rectangle using *Turtle graphics*. After practice students compare solutions and read the text about *Turtle* from S. Papert book "MINDSTORMS: Children, Computers, and Powerful Ideas (pp.75-76).

Practical methods

The third part of module consist of attractive tasks examples and *Computer Science Unplugged* activity. Solving these tasks and participating in the unplugged activity students practice different algorithms and compare solutions.

Tasks examples focus on decimal and binary numbers (Figure 2), pattern recognition, rules, cycle, Dijkstra's algorithm, calculations, programming. Teacher educator have to pay attention how student try

to find the solution, how they understand the tasks, compare solutions and provide a short overview at the end. Students in groups compare solutions, discuss about differences and similarities, methods used to get the correct answer.

A BINARY SCALE. A Beaver scale shows weight both in decimal (left) and binary (right) numbers. A fish weighs 1100 kg in binary number system. Which weights you need to put on the scale plates that you can see the fish's weight in decimal numbers?

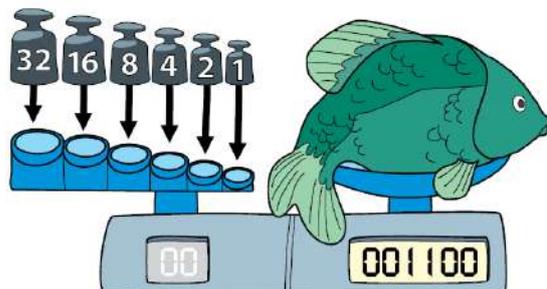


Figure 2. Task example

Later students practice *Computer Science Unplugged* activity “Orange game” (Figure 3). This is a cooperative problem solving game. The aim is for each person to end up holding the oranges labelled with their own letter (Bell, et al., 2015). This activity demonstrate constructivist approach.

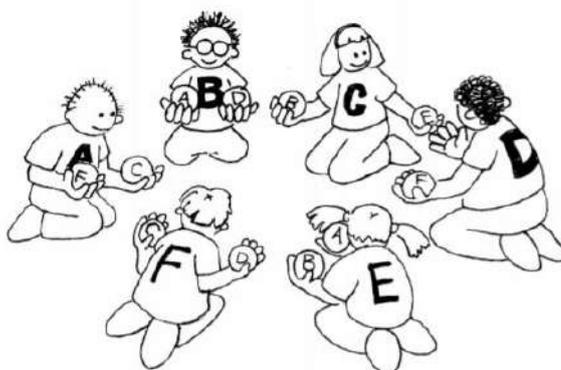


Figure 3. Orange game (Bell, et al., 2015)

Conclusion

The main idea of module “Different cultures – different approaches to reasoning and algorithms in mathematics” is to provide deeper understanding of diversity of approaches to reasoning and algorithms in mathematics. We focus on practical tasks using computers, technologies and various type of material. Practical, attractive activities and tasks engage students to reflect and share their own ideas, experience. All activities are covered with theoretical background on reasoning in mathematics and algorithms. Educator has opportunity to choose the particular activity depending on the student skills, practice, and related topic.

Acknowledgment

The presented module is based on the work within the project Intercultural learning in mathematics and science education (IncluSMe) of the Erasmus+ programme. The project grant no. 2016-1-DE01-KA203-002910. The authors gratefully acknowledge the project coordinator Prof. Dr. Katja Maaß, International Centre for STEM Education (ICSE) at the University of Education Freiburg, Germany.

References

- Astuti, E. P., Purwoko, R. Y. (2017). Integrating Ethnomathematics in Mathematical Learning Design for Elementary Schools in Proceeding "Research and education for developing scientific attitude in sciences and mathematics", 4th International Conference on Research, Implementation, and Education of Mathematics and Science, 2017 May 15-16.
- Bell, T., Witten, H., Fellows, M (2015). CS Unplugged. An enrichment and extension programme for primary-aged students.
- D'Ambrossio, U. Ethnomathematics and Its Place in the History and Pedagogy of Mathematics. For the Learning of Mathematics, Vol. 5, No. 1 (Feb., 1985), 44-48.
- Fisher, J. F., Davis, J. F (2008). Algorithms: through the ages and around the world.
- Kantner, J. (2008). The only absolute truth in mathematics is the myth of mathematics as universal. Perspectives: The New York Journal of Adult Learning 6(2). New York: Fordam University.
- Rosa, M., Orey, D. C. (2011). Ethnomathematics: the cultural aspects of mathematics. Revista Latinoamericana de Etnomatemática, 4(2). 32-54.
- Rowlands, S., Carson R. (2002). Where would formal, academic mathematics stand in a curriculum informed by ethnomathematics? A critical review of ethnomathematics in Educational Studies in Mathematics. Vol.50, No.1, 79-102.
- Perkins, I., Flores, A. (2002). Mathematical notation and procedures of recent immigrant students in Mathematics teaching in the middle school, 7, 346-351
- Philipp, R. A. (1996). Multicultural mathematics and alternative algorithms: Using knowledge from many cultures. Teaching Children Mathematics, 3(3), 128-135.

The Web – A Platform for Creation

Márton Visnovitz, visnovitz.marton@inf.elte.hu

Győző Horváth, horvath.gyozo@inf.elte.hu

Eötvös Loránd University, Faculty of Informatics, Dpt. of Media & Educational Informatics, Hungary

Abstract

The web is a platform that is becoming more and more accessible for many people. With the advancements in web technologies programming on the web is becoming easier to learn. The browser provides a programming API that can be used to express ideas quickly and easily. Using web technologies in learning programming has a lot of advantages, and this platform can be used to apply the constructionist methodology as well. With creative thinking, and basic programming knowledge students can use the web realise their ideas be it a simple computer game, a simulation or any other application. The same programming logic can be applied to many different scenarios; thus, students can explore the way to make their own ideas come to life, learning technology and programming in the process.

Every web application can be represented by three main components. These Three Pillars of a web-based application are: describing the underlying data structure, defining how that data can be displayed in a user interface and how that user interface behaves. Getting started with these steps does not require any technological knowledge about the platform itself, so students can acquire that during the process of implementation.

Keywords

Constructionism; web; JavaScript; game programming

Introduction

Nowadays the web is becoming a general, ubiquitous, rapidly developing and ever popular application runtime platform, that is used by billions of users every day. Its ubiquitous nature means that information and services are continuously available regardless of place, time and device. Web-based applications are becoming widespread and are present on almost every platform from the browser to smart phones, embedded systems and TVs.

Teaching of programming also uses the web for its purposes. There are many good examples of teaching various aspects of programming using videos, tutorials and online programming environments, e.g. Codecademy⁷⁵, CodeCombat⁷⁶. The programming language used in these environments can be anything from C# to Python, but HTML and JavaScript, the *lingua franca* of the web, noticeably occur often in these programming platforms. Altogether, these online learning environments consider the web as a runtime platform.

Web technologies are good for introducing programming to students in an exciting environment (Mahmoud et al., 2004). In this approach the web is not a platform, but the target of the teaching process. It provides all the necessary aspects for successful programming teaching: easy-to-learn and easy-to-use languages, modern tools and development environments, spectacular applications, familiar pieces of software like the browser (Horváth & Menyhárt, 2014). The whole teaching material (languages, technologies) can be introduced step by step, telling just the next necessary information (Horváth & Visnovitz, 2017).

Students can easily start programming with this platform as its main technologies are easy to get started with. Another benefit is that the definitive programming language of the web, JavaScript has many positive aspects from the educational point of view (Horváth & Menyhárt, 2014). A huge benefit of the

⁷⁵ <https://www.codecademy.com/>

⁷⁶ <https://codecombat.com/>

web platform that the markup language used for user interfaces (HTML) and the scripting language used for programming (JavaScript) both are extremely easy to get started with (Visnovitz, 2017). Both are just simple text files that don't require any boilerplate code to work, thus the source file only contains the code that represent the desired functionality, In this article we present how the web platform can be used as the target of a programming teaching in a constructionist way.

The Three Pillars of Web Applications

Creating applications for the web consists of three steps that can be executed one after another. The foundation of all these "pillars" can be CS Unplugged activities that don't require any previous knowledge. This way students can explore the process of implementing their ideas and learn about technologies that are required to build a web application.

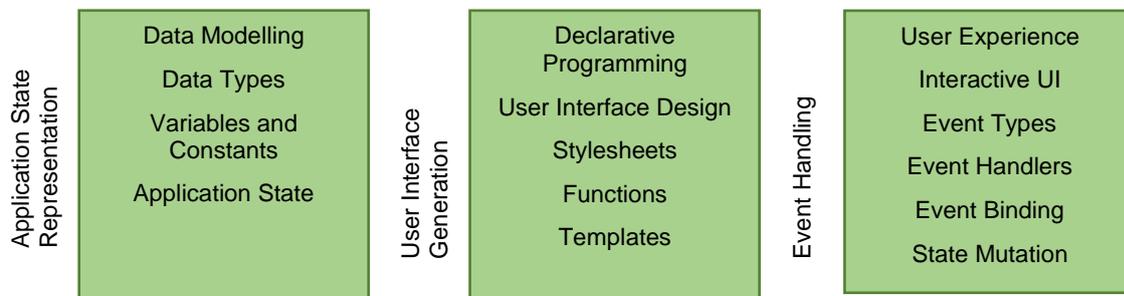


Figure 111. The Three Pillars of web applications

Application State Representation

To create any kind of computer program the first step is to describe the set of variables that represent the state of the application at any given time. This step can be entirely independent from the programming part and can be done even without computers. Students must answer simple questions like:

- What do I have to know about this program at any given time?
- What are the variables, what are the constants?
- How can I structure this data?

After answering those questions, they must translate the answers to some form of data model. This data model can be very basic with only names of variables and a description of what they represent, or they can be very complex with data type descriptions and custom data types. The complexity of this step can be adjusted to the previous knowledge of the students so that they can use their previously acquired knowledge but also can learn something new in the process.

During this step students can explore concepts of data modelling, data abstraction, basic and composite data types, the concept of constants and variables.

User Interface Generation

All users interact with a program using a user interface. Each UI can be a combination of static and dynamic parts. Dynamic sections of an interface are dependent on the applications current state while static parts do not change depending on the state. Students should identify the static and dynamic parts of the UI. Some governing questions for this step are:

- What do I want to show to the user?
- What does the user need to use my program?
- Which parts are static (state independent) and dynamic (state dependent)?
- What is the relationship between the state of the application and the user interface?

Students can design the user interface and user experience using simple tools or even pen and paper. This designing phase can help them to understand the various forms of user interactions Also, this step allows students to express their creativity when creating the looks and other visual elements of the UI.

For coding it is recommended to first create a static mock-up for the program that shows how it is going to look. This allows students to explore the markup language and styling without worrying about behaviour. The next step is to describe how the state of the application can be translated into this UI, what is the connection between the two. The last step is to create functions that implement this translation.

During this process students learn about declarative programming, user interface design, functions and templates. Alternatively, students can use 2D graphics techniques provided by the Canvas API of the browser to minimize the HTML knowledge requirement of the entire process. Instead of creating the UI with HTML elements students can use basic raster graphics, basic shapes and images to create the UI of a game (Horváth, Menyhárt & Zsakó, 2016).

Event Handling

The third thing to do with a web-based application is to make it interactive. Students can start this step by describing how the program behaves and what kind of interactions can users make. This step can be started by brainstorming and creating sketches and diagrams of the possible user actions. The three components of every interaction that students must identify are the following:

- What part of the user interface the user interacts with? (e.g. a button...),
- What is the nature of the interaction? (e.g. clicking, pressing a button...),
- What is the effect of the interaction? (e.g. the button turns red, a counter increases...).

Using these three components pupils can create event handlers that make the UI react to user input. Event handlers are basic programs with input/output handling and data processing, thus this is a very good exercise to learn or practice basic programming. The I/O in this case uses the JavaScript representation of the HTML page (DOM) to access UI elements and read or modify their content. A common event handler reads data from the UI and modifies (mutates) the application state. After modifying the state, the UI must be re-rendered in accordance with the changes. Sometimes it is overly complicated to store every single aspect of the UI in the state. In these cases, event handlers can also have side effects on the UI, such as modifying the visibility of an element or setting the focus. This means that the event handler directly taps into the DOM in an imperative way, modifying its contents or properties.

During this process students learn about the concept of events in computing, event types, event handlers, event properties, event binding and state mutation.

Practical considerations

For all the previously introduced Three Pillars there is a shared workflow that can be applied to each of them. This consists of brainstorming, planning & researching, and implementing. During brainstorming students collect their ideas and note them down using either paper and pen or the computer. This collection of ideas will be the basis of their execution plan and their research for corresponding technologies. The research phase can be assisted by the teacher. After students have a basic understanding of the technology that they shall use to realize their plan they can start coding the actual program. For coding the teacher can again function as a guide but can take a more direct and frontal educational approach if necessary.

It is important to note that using the Three Pillars concept does not have to mean that students must plan the entire program from the start. The business principles of Minimum Viable Product (MVP) and incremental development can be applied in this case as well. This means that at first students can create a program with a minimal feature set and later add new features by extending the existing three main components of their application. Helping questions for extending a program in such way are:

- What additional information do I need to know everything about the program with this new feature?
- What additions/modifications do I need to make to the user Interface?
- How do I translate this new state component user Interface?
- What additional interactions do I need to add this functionality?
- How do I have to change the current interactions to fit this new feature?

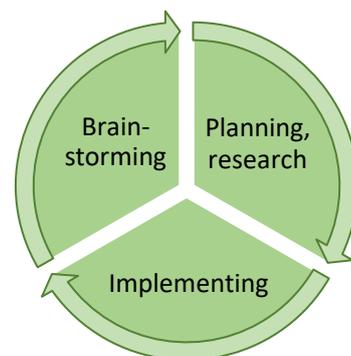


Figure 2. The cycle of adding new features to a program

Using this form of incremental development, the process of “brainstorming, planning/researching and implementing” becomes a cycle (Figure 2). From the start all the new features can be added using these steps.

Conclusion

The web is an easy to access platform that is available for all computer systems that have a web browser. Using client-side programming not even Internet connection is required to use JavaScript to create rich, web-based applications. Students can realize their ideas using a platform that they are most likely familiar with. Using the Three Pillars principle it is possible to start from CS unplugged activities and create a fully-fledged web application or game. The three main activities can all help students to learn about different aspects of programming while realizing their ideas.

By the nature of the browser and its main programming language, JavaScript, using web technologies makes it possible to create applications that can be deployed to both desktop computer, smartphones or other smart devices. This also can be very engaging for students.

Web technologies are widespread and are improving rapidly. HTML and CSS are well known technologies that are easy to get started with. In many countries HTML and CSS are even part of the high school curriculum. The JavaScript programming language is easy to learn and apply to many areas including server-side, client-side and mobile programming. Programming skills acquired with this language can be applied later in many different fields.

References

Horváth, Gy., Menyhárt, L. (2014). Teaching introductory programming with JavaScript in higher education. In Proceedings: *Proceedings of the 9th International Conference on Applied Informatics*, Eger, Hungary. p. 339-350.

Horváth, Gy., Menyhárt, L., Zsakó, L. (2016). Viewpoints of programming didactics at a web game implementation. In Proceedings: *Proceedings of the XXIX. DidMatTech 2016 Conference*, Budapest, Hungary. Eötvös Loránd University, Faculty of Informatics.

Horváth, Gy., Visnovitz, M. (2017). Egy bevezető webfejlesztési kurzus módszertani megfontolásai [Methodological Considerations of an Introductory Web Development Course]. In Proceedings: *Informatika a felsőoktatásban 2017 [Informatics in Higher Education 2017]*, Debrecen, Hungary, University of Debrecen, Faculty of Informatics. p. 265- 274.

Horváth, Gy., Visnovitz, M. (2018). A böngésző mint alkalmazásfejlesztési platform [The Browser as an Application-development Platform]. Retrieved March 30, 2018, from <http://webprogramozas.inf.elte.hu/tananyag/kliens/kliens.pdf>

Mahmoud, Q. H., Dobosiewicz, W., Swayne D. (2004). Redesigning Introductory Computer Programming with HTML, JavaScript, and Java. In Proceedings: *SIGCSE '04 Proceedings of the 35th SIGCSE technical symposium on Computer science education*, Norfolk, Virginia, USA, ACM SIGCSE Bulletin. p. 120-124.

Visnovitz, M. (2017). *Szöveges programozási nyelvek a közoktatásban [Textual Programming Languages in Public Education]*. MA thesis, Eötvös Loránd University, Budapest.

Programming Lessons for Kindergarten Children in Japan

Takeshi Watanabe, *watanabe@viscuit.com*

The University of Electro-Communications, LLC. Digital-pocket, Japan

Yuriko Nakayama, *nakayama@uec.ac.jp*

Kagawa-fujimigaoka Kindergarten, Japan

Yasunori Harada, *hakase@viscuit.com*

LLC. Digital-pocket, Japan

Yasushi Kuno, *y-kuno@uec.ac.jp*

The University of Electro-Communications, Japan

Abstract

Viscuit is a programming language developed by the one of the author. Distinguishing feature of Viscuit is that the programs are made of pictures only, and no characters are required to understand or make programs. This feature makes it possible to experience programming for pre-school children. Actual, LLC. Digital-pocket cooperates with Kagawa-fujimigaoka kindergarten in Japan for regular Viscuit classes since November 2015. In this poster, we analyzed children's programs and videos taken during the lessons held for them in 2017. In addition, we took questionnaire on the difference between the usual state and the state of the Viscuit lessons of the children to the teacher in charge of each class, and examined the features of the programming lesson at the kindergarten.

Keywords

keyword; programming for kindergarten; visual programming language; Viscuit

Preface

Background

In Japan, programming education is going to start from elementary school year 2020. As the Ministry of Education of Japan states, the aims for programming education in elementary school is not to acquire coding skills, but to acquire logical thinking abilities and problem solving skills or computational thinking skills (Wing, 2006). As for the programming experiences, children are expected to organize or combine primitive actions or symbols to construct the movement they want. As the result of a survey in 2016, the number of Japanese private programming classes are increasing, along with the children's age decreasing. Particularly, programming workshops or classes for preschool children are getting popular, perhaps parents are willing for their children to be get ready for elementary schools programming classes.

We observed that children who joined full-year lessons during 2016 had apparently changed in their programming abilities. And programs made by them became more complex and rich. In this poster, we show results from our analysis, along with actual programs made by children.

What is Viscuit

Viscuit is a programming language created by Yasunori Harada who is one of the authors, in Japan 2003 (Harada & Richard, 2003) (Harada, 2010), which is a rule-based visual programming language like KIDSIM (Smith, Cypher, & Spohrer, 1994). The most distinguish character of this language is user can make program without using any letters or numbers, but only with drawing. On Viscuit, you can make programs by putting your drawing on the rules that are similar to glasses (actually we call those rules glasses). Right lens of the glasses means "before" situation and left means "after" situation. If you put a picture upper on the right lens than picture that is in left, a picture on the stage get to move up direction.

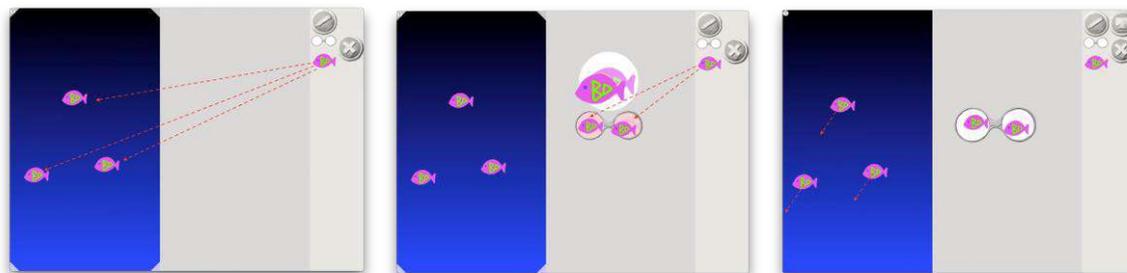


Figure 1. How to make program by Viscuit

Viscuit has a function of sharing program in a group. This function is called “Viscuit Land”. Programs created on each student’s tablet are gathered at another tablet through the network. For example, if the theme of a lesson is “sea”, participants will draw something that moves in the sea, make drawings move, send network and share programs in a group. In kindergarten, we have lessons using group work, not individual production.



Figure 2. Recital time by watching “Viscuit Land”

Kagawa-fujimigaoka Kindergarten

Programming lessons are held by kindergarten’ teacher. Nakayama, who is also one of the authors, is in charge of Viscuit lessons. Initially, the equipment has been brought by LLC. Digital-pocket, but then they bought iPad mini more than 30 units, also prepared Wi-Fi environment. So now, lesson has been carried out using all kindergarten equipment.

Purpose

In this research, we will clarify the growth of children that is shown in the continuing lesson of kindergarten. Also, clarify what preschool children are learning through programming lessons. Specifically, their programs that run in the computer are made as whether or not they wish. Also, we will clarify the features and important points of programming practice in kindergarten.

Research Method

Subject

The subjects are 56 children (5 and 6 years old) (28 people, 2 classes). In addition, a teacher in charge of each class is conducted a questionnaire survey on the state of the child at the end of each lesson. In the questionnaire, we asked the teacher how different between the child 's usual state and the state of the programming lesson, and who shows characteristic behaviour in the programming lesson.

Content of Lessons

There are some challenges for children to study advance programming every lesson. One lesson is 40 minutes. Lessons consist of three parts. Two practice tasks are carried out in the first half, and a free program production concerning those tasks follows. In the practice tasks, we prepare pictures for children to focus on their tasks. In the free production time we let children draw as they like unless it has relation to the task. Lastly, we have recital time that we appreciate what they make in groups on the screen like Figure 2.

On the other hand, it was found that even if the level of lesson did not raise every lesson, it could be enjoyed enough just by changing the picture of the practice. So, some important parts of programming are repeated again and again by changing the picture to fix the study without progress.

The layout of the classroom is like Figure 3. In this layout there are clearly separate "teaching space" and "production space". And by only placing the tablets in "production space" and do not allow children to bring tablets in "teaching space", we can make children listen to what the teacher says in the "teaching space".

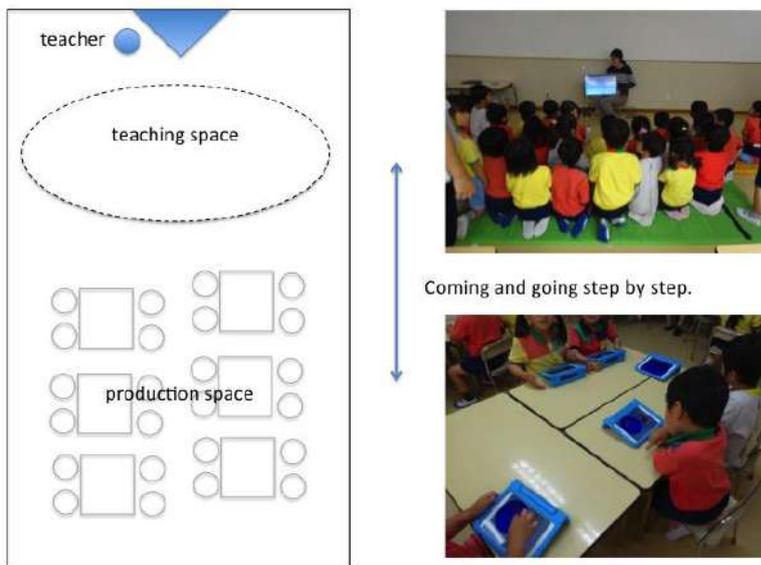


Figure 3. The layout of the classroom

When the teacher teaches something to children, they are gathered in "teaching space", and after they listen what they should do, then go "production space". This will allow children who cannot concentrate when the tablet is in front of them make concentrate on what the teacher says. This way of going and coming is only done at the time of the practice. When they work on free program, they remain "production space" and focus on what they want to make.

Analysis

Analysis of program

From programs saved at each time, we counted the number of works saved, the number of pictures drawn, the number of pictures on the stage, the number of glasses. And we told whether those glasses were valid or not, whether the direction of drawings and movement matched and whether the task and the movement of the pictures they made matched.

Analysis of video

I felt it difficult to understand what the children were learning with only quantitative data of programs. Therefore, we executed video research. We chose 4 characteristic children and shot and analyzed the situation of free tasks in video. As a result of observing the video, it turns out that each of the four people shows different changes and attitude.



Figure 4. Lesson in kindergarten. Children make programs with their fingers.

Inquiry of teachers

Regarding the questionnaire of each class teacher, we got a glimpse of the difference between the ordinary activity of the children and the programming lesson using Viscuit. In the free description of questionnaire the following comments came out. "Children's aggressiveness is high." "I think in this lesson we provide a different experience from the drawing on paper." " I was surprised that they had been able to maintain their concentration for 40 minutes." These are suggesting hints to clarify the features of programming lessons at kindergarten.

Conclusion

Now we have done analysis of the program of 9 children during May to July. As a result, 70% of children shows that a program that they made consistent with the tasks. In addition, we found that even if practicing tasks are completed in the lesson, children might start doing other challenges on the same stage and destroy previous program after they completed tasks. So, we could not understand whether they understand when they tried to practice tasks. Therefore, it necessary to take screenshots during lessons to check whether they manage to do tasks.

By observing each person about the video, we felt the possibility of being able to grasp the change of each person that cannot be found by program observation alone. Thus, we felt the possibility to grasp what kind of changes caused through programming lessons by combining log of program and observation of each person by video research.

References

- Harada, Y. (2010). Computer Programming Education Using the Visual Programming Language Viscuit. *NTT Technical Review* , 8 (11), 1-5.
- Harada, Y., & Richard, P. (2003). Fuzzy rewriting: soft program semantics for children. *Proc. of the 2003 IEEE Symposium on Human Centric Computing Languages and Enviroments* , 1 (1), 39-46.
- Smith, D. C., Cypher, A., & Spohrer, J. (1994). KidSim: programming agents without a programming language. *Communications of the ACM* , 37 (7), 54-67.
- Wing, J. M. (2006). Computational Thinking. *Communications of the ACM* , 3 ed.:, 33-35.

A Practical Report on a Programming Course with “Making” Using Micro:bit

Aoi Yoshida, aoi@si.aoyama.ac.jp

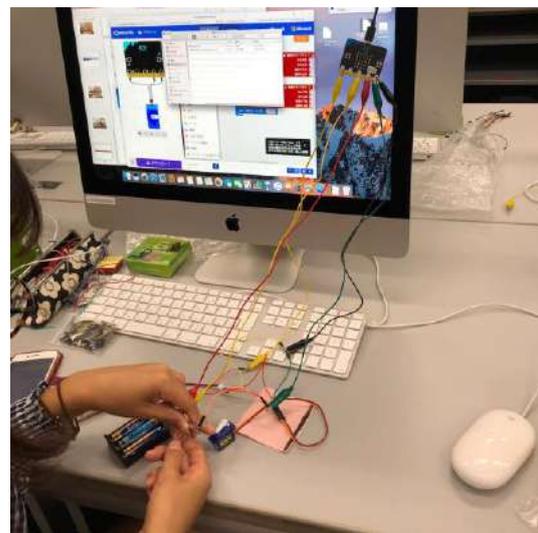
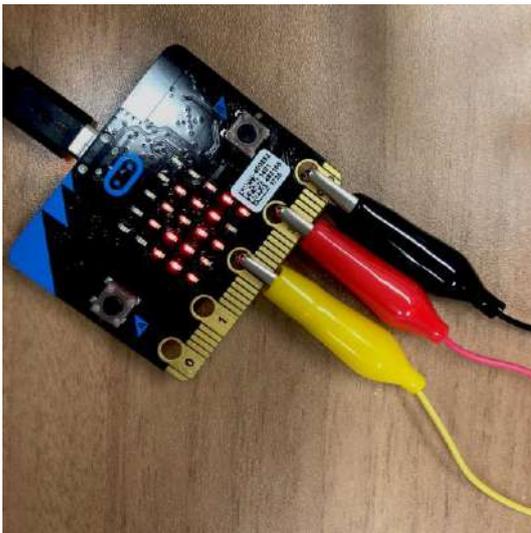
Kazunari Ito, kaz@si.aoyama.ac.jp

Kazuhiro Abee, abee@si.aoyama.ac.jp

Aoyama Gakuin University, Japan

Abstract

We design and practice hands-on course aimed at acquiring problem solving skills and thinking skills in programming through physical computing based on constructionism. We held 15 weeks classes, and there were 28 students who are third-year or fourth-year in this course. This course provided to make products with micro:bit using digital fabrication and physical computing. This poster describes the course content and students' final products.



Micro:bit (Left) Class scenery of this course (Right)

Keywords

physical computing; making digital fabrication; programming education

Introduction

We design and practice hands-on course aimed at acquiring problem solving skills and thinking skills in programming through physical computing based on constructionism. While previous studies have found that physical computing can be effectively incorporated into K-12 programming education, we believe that it can also contribute effectively to university education.

At Aoyama Gakuin University in Japan, we offered a course titled “General Practice in Information Science”. An objective of this course is to acquire knowledge and skills listed below:

- knowledge about sensors and actuators
- skill of controlling sensors and actuators by programming
- knowledge about technologies to constitute IoT
- skill of realizing own idea using information technology

For this purpose, we employed a method of learning by “making.” Students learn by making what they want to make. Lectures did not give students too much instruction.

Research question of this practice is to clarify what kind of knowledge and skill students acquire through this course.

Practice Outline

Participant

This course was offered in the second semester. There were 28 students who are third-year or fourth-year in this course. Most of them are studying programming (Java, Visual Basic, and Scratch). However, because of our multidisciplinary department, some students are not so good at programming. There were 2 instructors and 1 TA (teaching assistant, a graduate student).

Hardware and software

We used micro:bit, a an ARM-based embedded system designed by the BBC for use in computer education in the UK. Micro:bit equips accelerometer and magnetometer sensors, Bluetooth and USB connectivity, a display consisting of 25 LEDs, and two programmable buttons. There are some coding editor. We mainly used the JavaScript block editor on the web browser. (<https://makecode.microbit.org/>) On demand, students switched coding editors from block to text, and switched coding editors from JavaScript to Python. We rented “micro:bit box” to each student in order to use it freely outside the school hour. There are a micro:bit, a valuable resistor, a buzzer, a USB cable, and five cables with clips in the box.

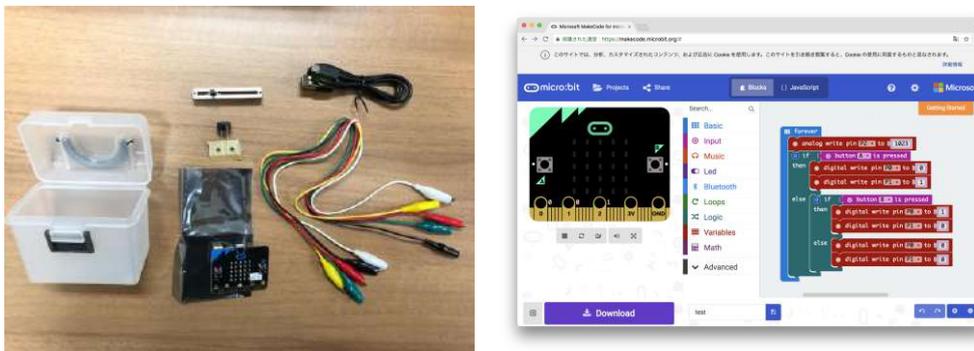


Figure 1. Micro:bit box that we rented (Left) Programming editor on the web (Right)

Course content

Table 1. The course content and activities

Phase	Class	Content	Activities
1	1	Basic of programming using micro:bit (except electrical work)	- Self-study on a text websites - Make a program freely
	2~4	Basic of programming using micro:bit (with electrical work)	- Hands-on practice
	5~6	Making (Individual work)	- Develop a product
	7	Presentation	- Show-and-tell
2	8	Digital fabrication	- Design 3D model for micro:bit case - Print out it on a 3D printer
3	9~13	Making (Group work)	- Think of creative ideas - Develop a product
	(12)	Intermediate presentation	- Show-and-tell - Peer review
	14	Interactive presentation	- Show-and-tell each other - Invest virtual money for favorite products in crowd-funding style
	15	Award and review	

Students' Final Products

All teams could develop and present their products, and most of them used CS technology other than what we taught in phases 1 and 2. Students' final products listed below:

- Shooting monsters game
- Music box
- Detector people entering the room
 - BLE communication, machine learning, various sensors, server-side programming
- Hourglass
 - BLE communication, 3D printing
- Savings box
 - servo motors, handcraft, structure
- Lier checker
 - electric resistance
- Remote cat feeding machine
 - Raspberry Pi, web camera, various sensors, server-side programming, laser processing
- Wake up device
 - a servo motor, handcraft
- Automatic switch press machine
 - BLE communication, various sensors

We introduced two examples of students' final products.

Shooting monsters game

This is a shooting game by Unity like an arcade game. When monsters appear, a player shoots them using a controller like a model gun. There is a micro:bit on the head of the gun. It is sent the value of inclination sensors from micro:bit to PC by serial communications. On the PC, a targeting is controlled using the received value.

This group made it consists of three students. Student1 made a controller by 3D printing(Figure 2). Student2 made a program on the PC side by Unity. Student3 made a program on the micro:bit side by Javascript (block editor).

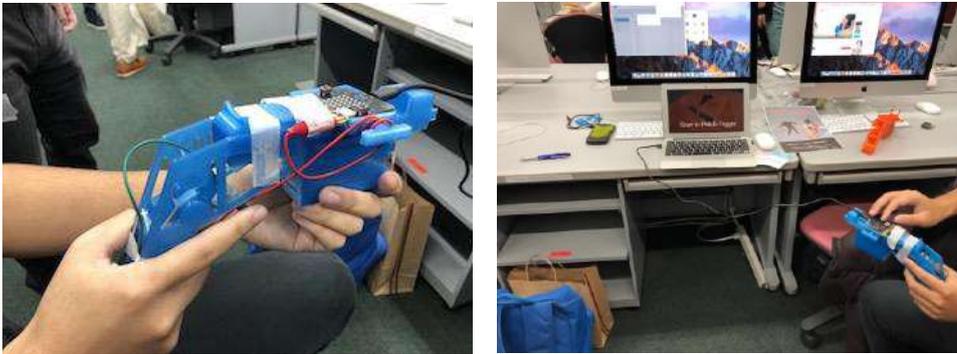


Figure 2. Shooting monsters game

Music box

This is a box that plays music by cooperating with multiple players. All players touch the objects of heart made by aluminum with one hand, and touch the aluminum buttons with other hand. And then, they can play one sound. It uses the electric resistance value from micro:bit.

This group made it consists of three students. They did all the work together.



Figure 3. Music box

The videos of this course and students' final products are available at <https://goo.gl/za13nY> .(only in Japanese. English version is to be prepared.)

Results

Students' final products show that each students acquired in a different knowledge and skill. We think that this is because the necessary knowledge is different when making what they want to make.

After all classes, we conducted questionnaires about this course. All the students except one were satisfied with this course, and also wanted to continue "making" and programming.

Moreover, we asked students to evaluate achievement for objectives of this class on five-grade.

- Q1. Can you understand the structure of various sensors?
- Q2. Can you acquire how to control with micro:bit?
- Q3. Can you understand technologies to constitute IoT?
- Q4. Can you realize your own idea?

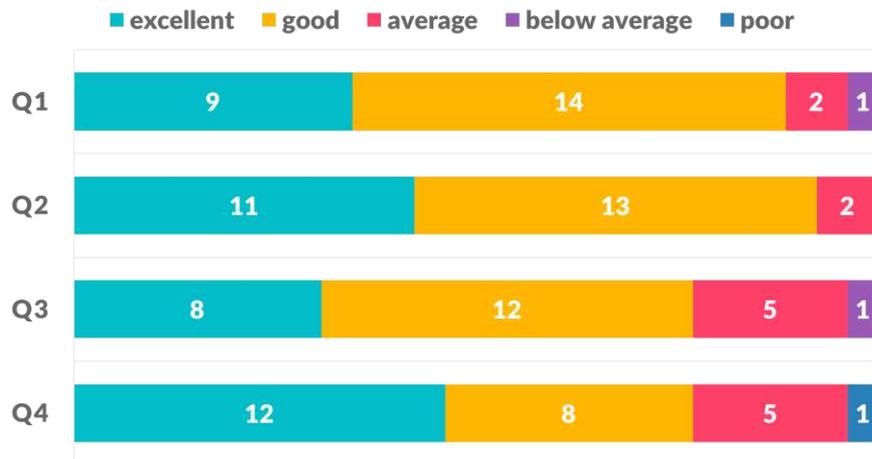


Figure 4. the result of students' self-assessment

Figure 4 shows that most of all students achieve an objective of this course by themselves though lectures did not give students too much instruction.

According to free description in a questionnaire, a lot of students answered that they thought deeper and participated in classes more aggressively than usual.

As future work, we plan to improve this course content more effectively. We also plan to expand target.

References

Papert, S.(1977) A Learning Environment for Children. Computers and Communication: Implications for Education. New York, Academic Press, p. 271-278.

Papert, S. (1980) Mindstorms, Children, Computers, and Powerful Ideas. Basic Books, Inc.

Resnick, M. (2017). Lifelong Kindergarten: Cultivating Creativity through Projects, Passions, Peers, and Play. MIT Press.

Sylvia, L.M., Stager, G. (2013) Invent to Learn: Making, Tinkering, and Engineering in the classroom. Constructing Modern Knowledge Press.

An Experimental Exploration of the Development of Design Thinking in University Maker Courses

Jinbao Zhang, Zhangjb@bnu.edu.cn

Beijing Normal University, Department of Education, China

Abstract

At present, Maker activity in colleges and universities rarely appears in the formal curriculum, and most of them exist in makerspace, training activity of entrepreneurship. In the research, we design a new undergraduate course named as *3D Printing and Maker Education*, including two parts: (1) the application instruction about the technologies of 3D printing, 3D software, Arduino and APP Inventor; (2) design thinking based innovative and creative design practice. In the process of the curriculum, the author does some observation and interview. The primary findings include the design thinking application level in creating scenarios and formation, the bottlenecks and obstacles encountered in various stages of design thinking.

Keywords

design thinking; maker; curriculum design; instructional design

Introduction

The Maker Movement and Maker Space, which have risen in recent years, have won recognition and attention from most people as a new educational phenomenon (Li et al., 2016). In universities, there is little research conducted in the form of a just-in-time and formal course. Whether this kind of education idea is suitable for the university curriculum implementation model, what problems will exist, how to design to play the value of such a teaching concept for students, are the starting point of this study.

Design is a purposeful behavior that requires creativity and innovation (Liu, 2015). The process of design thinking is a combination of divergence and concentration of thought (Lawson, 2005). Design thinking contains deep insight into the target group and efficient prototype testing, making the proposed solution more effective, far beyond the traditional hypothetical thinking. Besides, the emphasis on design thinking is to activate students' original thinking and expression methods, emphasize the cultivation of creativity and individuality. As it is closer to the essence of design education, it is "teaching for creativity" to become one of the main objectives of higher education teaching and has high educational values. Therefore, it is necessary to take "design thinking" in the design and creation process as the primary objective of the practical curriculum.

Based on the analysis of the idea of creating customers and the characteristics of undergraduate public elective courses, the researcher proposes an innovative curriculum that takes design thinking as the primary educational goal. With a relatively flexible teaching arrangement, through the implementation of the concept of "genius hour" (Liu, 2016), the undergraduate course at Beijing Normal University named as *3D Printing and Maker Education* is practiced to study whether students' design thinking can be well developed.

Research design

Given many students and the limited learning time, curriculum designers have thought that the core goal of this course is to "design thinking education under the guidance of creating for customers" rather than merely learning kinds of tool. Then researchers developed curriculum modules, teaching processes, and lesson schedules. The teachers at each stage are not the researchers themselves. It is beneficial to avoid the influence of the subjective approach of the researcher on the research and guarantee the objectivity of the study.

Research hypotheses

This study proposes the following hypotheses:

- 1) Design thinking as a guiding method is more conducive to designing innovative courses than previous teaching methods;
- 2) After the course, students can master 3D printing, open source hardware, open source software, and other standard maker tools and software, as well as a complete set of solutions to problems (design thinking);
- 3) Compared to traditional curriculum, this kind of teaching arrangement based on the theme of maker activity can be more favoured by students.

Research methods

To collect enough procedural data, this study adopts an educational observation method to record the entire process of class (teacher's instruction methods, teacher-student interaction process, and responses of most students). Students are interviewed after class. The research data collection tools are shown in the table 1.

Table 1. Data Collection methods

Data collection methods	Dimensions
Classroom Observation	Teacher Performance Student performance Interaction between teachers and students
Structured Interview	Degree of satisfaction of course content Satisfaction degree of teaching effect Teaching Suggestions Self-capacity growth

Data analysis and key findings

Findings in classroom observation

- 1) By observing the progress of each group's project, it was found that most of the teams had "disagreement of work" and "less cooperation". Team building may require teacher intervention.
- 2) In the group work, there are crucial problems in their learning: single player working hard without cooperating with others, team leader without relying on team member, more time to catch up and less to follow the plan, ignoring the actual printing, less logical planning of design thinking and so on. The team project may require more attention and guidance during the process of advancement.
- 3) But there is still a lot of good news: liberal arts students (Chinese, art, calligraphy, etc.) have a lot of creative ideas and some have gotten the supported by schools; students are enthusiastic and passionate, if they are supported and continue to be payed attention, students will come to the 3D Printing laboratory every day; young students have strong learning ability, they can master the knowledge quickly.

Structured interview conclusions

After the course is over, one-on-one interviews will be conducted with students in class. The following aspects need to be focused on:

- 1) Not all modules have gotten students' confirmation. students are deeply impressed by the following courses: First-course "Introduction to 3D Printing and Maker Education", the sixth-course

"Mixly Maker Education", and the seventh course "General "Application of 3D printing technology in technology courses" and the eighth course "APP Inventor".

2) The great reaction is that the curriculum is too fast. There are too many contents of the seven courses, and the time may be insufficient to be completely digested.

3) At the knowledge level, all students have a deeper understanding and unique understanding of 3D printing knowledge, open source hardware categories and principles, and app mobile terminal development;

4) Regarding core concepts, most of the students have a deeper understanding of the spirit of the Maker Movement which has promoted their practice in action. And in the final interview, we learned that apart from the group project, almost all students have things they want to make;

5) In the aspect of improvement of design thinking ability, almost all students can combine and complement one of their core concepts (user thinking, empathy) and the feature of maker (customer-oriented thinking makes them more targeted, continuous improvements) have a clear understanding;

6) Most students can recognize one of the core concepts of design thinking, but only a small part of these tools can be used proficiently (for example, in the development of group activities, the "plan assessment diagram"). Most of the tools can be used to eliminate small components and obtain a unified opinion. Many tools cannot be applied in actual activities.

Conclusion

Although it is trial and experiment, it also can get some inspiration for the future research and instruction:

1) The full range of design thinking tools is not required for every project. For example, the "user feedback" tool for Empathise. Since almost all student projects are modelled on real-world scenarios rather than actual projects in real society, the actual curriculum, and after-school activities are mostly substituting team members for playing customer roles in different situations to observe customer needs.

2) The application environment of some design thinking tools is relatively complex and not easy to master. A set of tool theory in precision manufacturing and quality control needs to be combined with the "brainstorming" method in order to be mastered. There are not many environments that need to be used and skilfully used in student projects.

3) Some of the design thinking tools have similar functions to each other. For example, the "priority chart" and "plan assessment chart" can compare and evaluate different programs.

For this course, we did not adopt a task-driven but fully open curriculum model. Students' performance was smooth after initial exposure to confusion, repeated selection of project plans, and selection of directions. The lecture-style teaching mode that is different from the ordinary course teaching mode is adopted. After the interview, most students will compare with the ordinary course how much they have learned the specific knowledge in the classroom. For example, if there are seven groups or less feedback did not learn or master three-dimensional design software, open source hardware, and App Inventor. This is not consistent with the goal of this course, which mainly aims at cultivating students' design thinking ability. How to improve it still needs further study.

References

Li X., Gao H., Zou J., & Wan K. (2016). Changes of STEAM Education to Maker Education in the "Internet+" Background - from project-based learning to the development of creative abilities. *Journal of Distance Education*, (5), 28-36.

Liu Jing-wei (2013). *Design Thinking*. Beijing: Chemical. Industry Press.

Liu, L. (2016). Inquiry and innovation in the classroom: using 20% time, genius hour, and PBL to drive student success. *Innovations in Education & Teaching International*, 10(1), 472-472.

Lawson, B. (2005). *How Designers Think: the design process demystified* (4th ed.). London: The Architectural Press.

Panels / Workshops / Demonstrations / Working groups

Panels

Constructionism across Cultures: Commonalities and Differences of Constructionist Implementations around the World

Paulo Blikstein, *paulob@stanford.edu*
Stanford University, USA

Abstract

The goal of this panel is to bring together scholars from around the world to discuss commonalities and differences amongst implementations of constructionist projects in different countries. Seymour Papert himself refused to precisely define what constructionism was, which opened up the possibility of local contextualization and definition by scholars and practitioners. The papers in this symposium analyze how constructionist theory was used, defined, and implemented within rich cultural contexts, discussing particular characteristics that might have emerged as a result of the combination of local educational practices and philosophies and constructionist theory.

Keywords

Constructionism, cultural context, implementation

Constructionism at Scale

Celia Hoyles, *c.hoyles@ucl.ac.uk*

Richard Noss, *r.noss@ucl.ac.uk*

University College London, UK

Yasmin B. Kafai, *kafai@upenn.edu*

University of Pennsylvania, USA

Kylie Peppler, *kpeppler@indiana.edu*

University of Indiana, USA

Deborah A. Fields, *Deborah.Fields@usu.edu*

Utah State University, USA

Nathan Holbert, *holbert@tc.columbia.edu*

Teachers College, Columbia University, USA

Abstract

Constructionist designers have used new technologies to engage learners in rich opportunities to build personally meaningful artifacts for decades. In recent years, new technologies have brought these experiences to large numbers of learners in distributed places. In this symposium we bring together expert designers and scholars that have successfully developed constructionist innovations using emerging technologies in a range of domains and have brought these innovations to scale. In a panel discussion format, participants highlight key challenges for scaling constructionist design and discuss how these tools and environments evolve as the community of learners increases by orders of magnitude.

Keywords

programming; construction; making; mathematics; scale

Inside the Trojan Horse – A Discussion Among the Next Generation of Constructionists

Sylvia Martinez

Constructing Modern Knowledge, USA

Abstract

“I think the technology serves as a Trojan horse all right, but in the real story of the Trojan horse, it wasn't the horse that was effective, it was the soldiers inside the horse. And the technology is only going to be effective in changing education if you put an army inside it which is determined to make that change once it gets through the barrier...

Just 100 years ago, John Dewey was saying things about educational change, not very different from what I believe in. He couldn't get very far. And the reason why he couldn't get very far is that he had only philosophical arguments. He didn't have an army. You must have an army, and it's an army primarily of children and the adults also are a political force in this.” (Papert 1999)

This discussion, comprised of practicing educators from around the world, will address personal challenges and triumphs bringing constructionism to life on a daily basis. Examples of classroom practice, student projects, professional development, and strategies for sustaining constructionism will be shared.

The panelists:

- Sylvia Martinez – President, Constructing Modern Knowledge (moderator)
- Amy Dugré – Director of Technology, Innovation, and Curriculum: Dusseldorf International School, Germany
- Angela Lombardo – MalpighiLaB & CoderDojo Bologna, Bologna, Italy
- Susana Tesconi - Lecturer and researcher at Department of Information Technology /UOC/ Universitat Oberta de Catalunya DARTS/Interdisciplinary research group in arts, technoscience and society, Barcelona, Spain
- Tracy Rudzitis – Faculty, Constructing Modern Knowledge, USA
- Brian C. Smith – Computing & Learning Technology Teacher: Hong Kong International School, Hong Kong
- Jaymes Dec – Fab Lab Integrator: Marymount School, New York City, USA

Papert, S. (1999). "Ghost in the Machine: Seymour Papert on How Computers Fundamentally Change the Way Kids Learn." [Interview of Seymour Papert by Dan Schwartz](#).

Keywords

constructionism, Logo, Seymour Papert, Scratch, physical computing, coding, teacher education, fabrication

Workshops

WS1: The Essence of Programming at School – Learning for Life

Jacqueline Staub, Jacqueline.staub@inf.ethz.ch

ETH Zurich, Switzerland

Pädagogische Hochschule Graubünden, Chur, Switzerland

Abstract

School is responsible for priming and preparing pupils such that they develop a deep understanding of technology. Computer science education serves a vital role in fostering algorithmic thinking and problem solving skills, as exemplified by programming. This form of learning is constructive, enriches creativity and teaches precision. We have been introducing primary school pupils and their teachers to programming in Logo for more than a decade and thousands of children across Switzerland have learned to program using our curriculum and purpose-built programming environment. In this workshop, we give insights into how our curriculum guides pupils to progress individually and how we make pupils building up competence by recovering from their programming errors autonomously. This workshop caters towards educators and people interested in how to introduce computer science to novices. Participants gain practical insights into our curriculum and discuss its didactic structure.

Keywords

algorithmic thinking; constructionism in novice programming; Logo; spiral curricula; modular design; debugging; syntactic errors

Materials Provided

In this workshop, participants will be given the opportunity to learn about our approach of teaching algorithmic thinking to novices by actively programming in Logo. Workshop attendees will solve selected exercises from our curriculum at a computer and later discuss the thought processes children go through. One major theme of discussion is the role of errors; we claim that children who learn to recover from errors autonomously gain higher confidence and a deeper understanding for core computer science concepts.

Participants will be enlightened with the core design ideas behind our curriculum involving the concepts of repetition, modular design and parametrization as a preparation for variables – all neatly woven in a spiral curriculum.

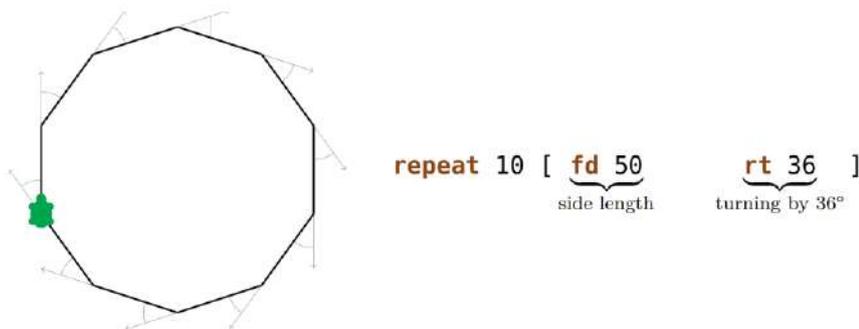


Figure 1. Example of an introductory exercise from the chapter on regular polygons and circles

Exercises will address:

- A basic instruction set used for drawing lines, rotation and moving a turtle
- A simplified looping construct which does not make use of variables
- A mechanism to extend the language by introducing new commands and nesting them to solve more complicated exercises
- Generalizing solutions to existing problems by parameterizing custom commands

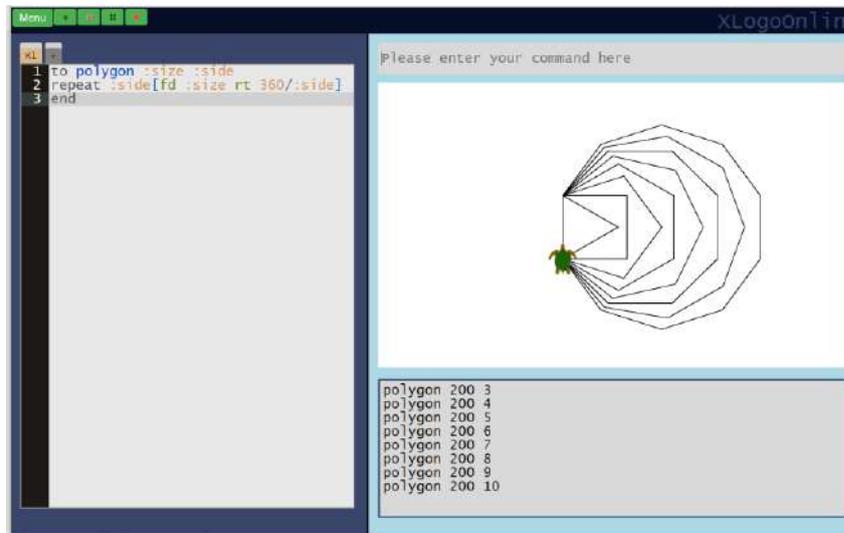


Figure 2. Drawing polygons with custom side lengths and number of sides

Resources:

Both programming environment and curriculum are freely available online in various languages. They can be downloaded from <http://www.abz.inf.ethz.ch/logo>

Agenda

1. Introduction: A decade of introducing a programming curriculum at Swiss schools
2. An Example: Algorithmic thinking as a problem solving strategy – a linguistic view
3. Our approach: Problem decomposition and an iterative refinement of solutions
 - i. Theory: Why use modular design in the first place? Implications on the cognitive development of novice programmers
 - ii. A practical example: From squares to circles
4. Hands-on: Our curriculum illustrated
5. Conclusion: Our approach introduces children to algorithmic thinking, a skill for a lifetime
6. Plenary Discussion: Participants ask questions on the matter of experiences, setup in schools, limiting factors, typical errors and debugging strategies

Biography

Jacqueline Staub is a second-year PhD student in didactics of computer science at ETH Zurich and a researcher at the STEM-Research-and-Services division at Pädagogische Hochschule Graubünden - a university of education based in Chur, Switzerland. She works on the topic of how to introduce pupils to algorithmic thinking and enabling them to progress autonomously. Seven years ago, she joined the group as a student and henceforth acted as a Logo teacher in many primary and lower secondary school programming courses. She completed her studies in Computer Science at ETH Zurich and, furthermore, graduated the teachers' education program. In her Masters' thesis she developed XLogoOnline, a programming environment for novices which is tailored around the curriculum we present and use in this workshop. Lately, her main focus lies within finding a model to identify the struggles pupils go through while programming and how to counteract these struggles by providing additional help through the programming environment.

References

- Heidi Gebauer, Juraj Hromkovic, Lucia Keller, Ivana Kosírová, Giovanni Serafini, and Björn Steffen. Programming in LOGO (Status as of March 18, 2018). http://abz.inf.ethz.ch/wp-content/uploads/unterrichtsmaterialien/primarschulen/logo_heft_en.pdf.
- Hromkovič, J., Kohn, T., Komm, D., & Serafini, G. (2016). Examples of algorithmic thinking in programming education. *Olympiads in Informatics*, 10(1-2), 111-124.
- Hromkovic, J., Kohn, T., Komm, D., & Serafini, G. (2017). Algorithmic thinking from the start. *Bulletin of EATCS*, 1(121).
- Hromkovic, J. (2014). *Einführung in die Programmierung mit LOGO*. Springer Vieweg.
- Serafini, G. (2011, October). Teaching programming at primary schools: visions, experiences, and long-term research prospects. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 143-154). Springer, Berlin, Heidelberg.
- Hromkovič, J., Serafini, G., & Staub, J. (2017, November). XLogoOnline: a single-page, browser-based programming environment for schools aiming at reducing cognitive load on pupils. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 219-231). Springer, Cham.

WS2: Developing Body Tracking Software with Scratch and Kinect

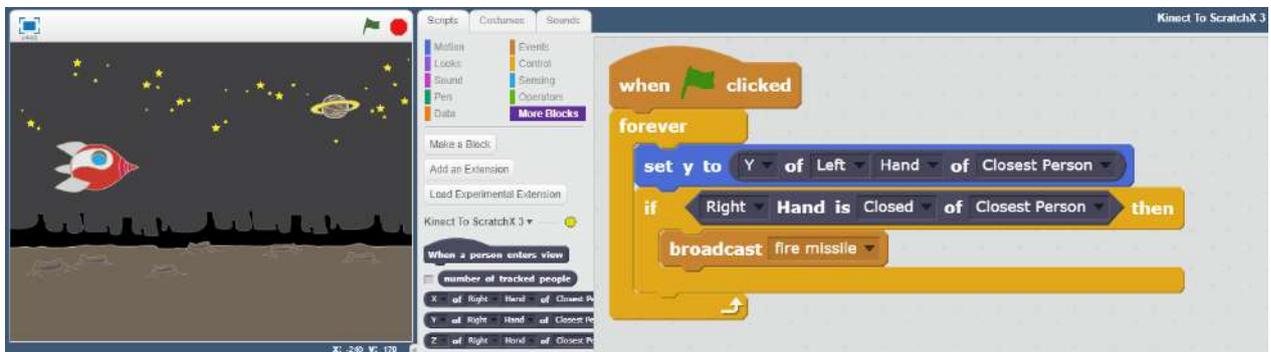
Stephen Howell, stephen.howell@ucdconnect.ie
SMARTlab, University College Dublin, Ireland

Lizbeth Goodman, lizbeth.goodman@ucd.ie
SMARTlab, University College Dublin, Ireland

Abstract

In this workshop participants will design, develop and test programs that can be controlled with the user's body. The code is developed with a visual programming language (Scratch) and an infrared body tracking camera (Kinect).

Starting with simple, familiar retro games, and a creative art and music performance application controlled by keyboard and mouse, participants will learn how to add tracking blocks so prescribed movements including individual joint positions, gestures, and stances can be continuously tracked and responded to.



Scratch sample code with a spaceship sprite tracking the user's left hand on the y axis, and tracking the user's right hand on the open/closed gesture

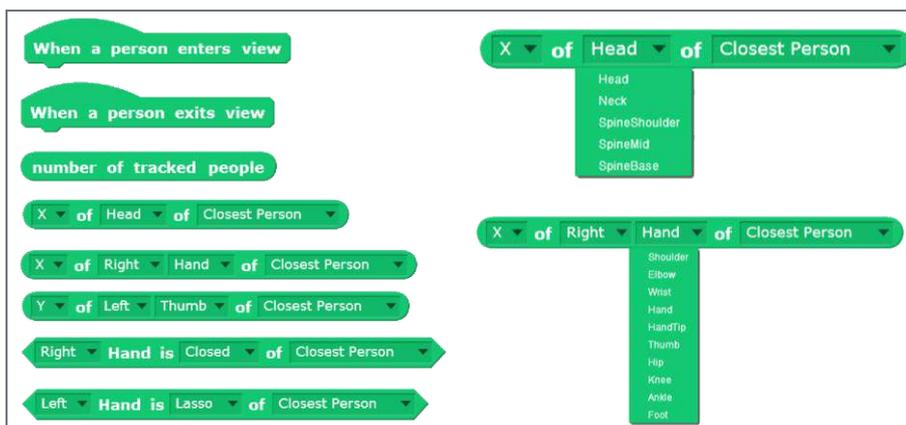
This workshop is appropriate for educators and researchers interested in teaching Computational Thinking by having students develop proprioceptive programs.

Keywords

visual programming languages; body tracking; proprioception; body syntonicity

Workshop

This workshop explores *Kinect2Scratch*, a hardware extension for Scratch that enables body tracking games and applications to be developed simply and rapidly (Howell, 2016). This extension was originally developed to enable children learning Computational Thinking code more complex Scratch projects using novel input controls. This includes moving their heads, limbs, and making gestures such as clapping, jumping, and pointing. This is made possible by adding additional event and reporter blocks to Scratch that receive data from an infrared sensor with body tracking capabilities. The body tracking data is normalised for the Scratch stage and streamed to the Scratch project continuously. Each new block allows the developer to detect, track, and respond to bodily movements of either the closest participant to the sensor or up to 6 different users at once.



The Kinect2Scratch extension blocks

Format

The workshop format will be interactive and include an initial demonstration of how the body tracking Scratch programs are developed, including a simple retro game and creative art and music performance program. No prior knowledge of Scratch is required but familiarity with a visual programming language would be useful. The samples will be ‘live coded’ to explain the reasoning for their inclusion and not presented as previously completed samples.

Following this, participants will learn how to respond to gestures including Boolean gestures (e.g. is a user’s hand open or closed) and complex gestures (e.g. is a user clapping, kicking, or jumping). Based on participant suggestions, we will then build a new Scratch program incorporating the gestures and movements the audience wants to track. Participants will then be able to test and refine the gestures using the workshop sensor.

Participants

A typical participant may be interested in teaching Computational Thinking principles or exploring syntonc learning. However, several researchers without software development backgrounds have published work using Kinect2Scratch to conduct research that required body tracking. For that reason, participants who are interested in writing body tracking software themselves but not necessarily teaching it to students may find this workshop useful. Participants do not need any special equipment, as the Scratch programs will be developed using a sensor and the workshop presenter’s laptop.

References

Howell, S. (2016). Kinect2Scratch v3 [Computer software]. Retrieved from <https://stephenhowell.github.io/kinect2scratch/>

WS3: Group-based Simulation and Modelling: Technology Supports for Social Constructionism

Corey Brady, corey.brady@vanderbilt.edu
Vanderbilt University, Nashville TN, USA

Walter Stroup, wstroup@umassd.edu
University of Massachusetts Dartmouth, North Dartmouth MA, USA

Tony Petrosino, ajpetrosino@austin.utexas.edu
University of Texas at Austin, Austin TX, USA

Uri Wilensky, uri@northwestern.edu
Northwestern University, Evanston IL, USA

Abstract

Through a US National Science Foundation-funded project called Group-based Cloud Computing (GbCC), we have created a web-based system for designing and implementing group-centred modelling activities on top of the NetLogo Web platform. For the past two years we have been iteratively building and testing this system in our own classrooms and in the University of Texas's UTeach and UTeach Computer Science programs. In this **workshop**, we introduce GbCC to the Constructionism community, bringing participants to a point where they can create and publish their own activities. Removing barriers to designing one's own group-based activities, we open up the discussion to engage the research and teaching challenges and opportunities for a deeply *social* form of constructionism.

Keywords

collaborative learning; collective learning; generative design; social constructionism; agent-based modelling; participatory simulations

Introduction

Perhaps the most salient feature of learning in schools is the group-based nature of classroom life; however, typical classroom instruction focuses primarily on individual learners, emphasizing approaches such as assignment sets of procedural exercises and questioning that follows an initiation-response-evaluation (IRE) sequence (Mehan, 1980). In contrast, with a *generative* approach, instructors design situations for groups of students to construct relations between stored knowledge, experience, and new information (Wittrock, 1991). Using the taxonomy of generative design provided by Stroup, Ares, and Hurford (2004), our project uses the “resonant” technological infrastructure of collaborative classroom networks to engage learners in generative learning around the core practice of modelling. We build upon the broad cross-disciplinary reach of agent-based modelling and the NetLogo language and platform (Wilensky, 1999), to support activities in the Social Sciences and STEM disciplines, including Computer Science.

Attempts to “scale up” individual learning approaches fail to capitalize in essential ways on the diversity of thinking that can be found in the group. In contrast, our GbCC project aims to use the technological infrastructure of collaborative classroom networks to engage learners in generative learning around the core practice of scientific modelling. In particular, we aim to support activities in which the in-process learning of one's peers can act as a learning environment to cultivate not conformity and convergence toward a single, “right answer” but rather diversity and a collective exploration of the space of possible ideas.

In the GbCC project, we connect innovative approaches to computational and participatory modelling to theoretical roots in Constructionist thought. In addition, the project is creating detailed analyses of activity designs and implementations fostering authentic STEM practices in group-centred learning environments, as well as nuanced studies of high-leverage teacher “moves” and strategies within such

environments. This workshop offers a hands-on introduction to the ideas of generative and group-centred design that have inspired GbCC, and our plan for the session is driven by the goal of inviting attendees into the group-centred discussion and enabling them use the technology to express their own ideas for activity designs.

Workshop objectives

The workshop will be of interest to participants who engage with learners at middle-school, high-school and university levels. A projector for the facilitator is required, and all participants need internet-connected computers. Thus, having a few extra computers available would be highly desirable. By the end of the workshop, our intention is for every participant to have created their own GbCC activity and either made it “live” on their personal (new or existing) Heroku account, or sent the activity to the workshop facilitators to publish it on the GbCC project’s Heroku account. We choose this objective in part to ensure that participants take away something concrete and useful, and in part to demonstrate that we have achieved the goal of making social constructionist learning as simple to support *technologically* as activities that do not offer group-centred collaborative supports. Of course, success here then allows us to turn attention to the challenges of adopting group-centred pedagogies and designing activities that make effective use of the group. But this is a challenge we welcome and where we have also made headway, as we will illustrate through brief experiences of several “genres” of group-centred activities that we have identified. (This is not an exhaustive list of genres the project has explored.)

Genre 1: Exploring Together. A common pattern in classroom agent-based modelling instruction is to explore the “parameter space” in an existing model by changing the values of key inputs to the simulation. Key goals of such activities are to investigate the range of behaviours the model can exhibit and to determine how sensitive these behaviours are to changes in the model’s parameters. However, with NetLogo and NetLogo Web alone, it is difficult to bring different students’ discoveries together in an aggregate display of the class’s exploration, to facilitate group discussion. With GbCC, *any* NetLogo model can be converted easily and deployed as a group exploration activity. Each student can publish her findings to a shared “Gallery,” which can permit others to see and also repeat (to examine and/or verify) the findings that she has published. “Going public” in this way with in-process findings has been shown to enhance classroom learning in agent-based modelling and to help identify emerging themes in group exploration (Brady et al, 2015).

Genre 2: Constructing Together. Whether as an introduction to the NetLogo language as an expressive medium, or as a means of identifying and exploring the impact of design choices in modelling a particular phenomenon, GbCC can be used to open a space for a classroom group to rapidly create and share “snippets” of NetLogo code that can define agent or model behaviors in a way that supports radical remixing and collective construction. For instance, in an activity we have called “IntroButtons,” learners are given an array of buttons that allow beginners to NetLogo code to explore Turtle Geometry collectively. By producing “something beautiful” on the screen, they begin to explore what turtles and patches can do. On publishing to the Gallery, learners share the graphic *results* of their explorations as well as *text-based code representations* of the command sequences that produce these results. Sharing these personally meaningful objects helps the classroom group to make the move to text-based coding, together. For more advanced groups, similar activities to IntroButtons can invite learners to create varying implementations of a particular agent behaviour and help the classroom to “think together” about modelling and code representations of phenomena they are studying.

Genre 3: Participatory Simulations. Since its earliest releases, the NetLogo platform has incorporated the HubNet architecture (Wilensky & Stroup 1999a) as a technological support for a genre of group-centred modelling activities called Participatory Simulations, or PartSims (Brady et al, 2017; Colella, 2000; Colella, Borovoy, & Resnick, 1998; Wilensky & Stroup, 1999b; Klopfer, Yoon & Perry, 2005). The GbCC system also provides support for PartSims, and it removes many of the implementation barriers that past research has encountered with these activities, by being fully web-based. Any browser-capable device (including cell phones, netbooks, and single-board computers, as well as tablets and laptops) can participate in GbCC activities over the standard HTTP network connection permitted by almost all school networks. Moreover, the GbCC’s constructs of activity “Rooms” and the role of “Room Creator”

(which can be filled by the teacher or even by a student small-group leader) makes setting up and running a PartSim as easy as following a link and giving the new Room a name. These technical innovations have come out of a decade of design research on making HubNet activities more feasible for real classroom implementations, and they have also radically broadened the design space for PartSims allowing us to engage, for instance, with in-and-out of classroom activities and activities involving data collection and the Internet of Things (IoT).

Genre 4: “Fish Tank” and “Program to Participate” Simulations. A common thread between agent-based modelling in general and PartSims in particular is that they offer opportunities for social structures and interaction patterns of the classroom to be mapped onto structures and agent-interactions in the phenomena being modelled (Stroup, Ares, Hurford, & Lesh, 2007; White, Brady, Huang, & Stevens, in press; Wilensky & Stroup, 1999b). This enables groups to identify a *social syntonicity* (Brady, Weintrop, Anton, & Wilensky, 2017) in their shared modelling work. One approach we have explored to use social syntonicity to drive collective modelling has been to identify key behaviours of agents (whether of one breed or several breeds), and to create a model similar to a PartSim, but where students’ participation consists of contributing the *code* that their agent ‘avatars’ will use to execute these behaviours. In this way, the traditional PartSim genre can move beyond an activity structure in which learners are immersed in a simulation, role-playing agents, to creating behaviours for those agents, which then run collectively. We do not see these “program to participate” activities as replacing traditional PartSims; rather they foster a different perspective on the emergent phenomena studied in the simulation and they offer opportunities for new ways to distribute the expression of model behaviours over the room. Moreover, as behaviours can be expressed in multiple agents and multiple agent breeds, the students’ connection to the simulation can be varied to exploit resonances with social and group structures.

Integrating other Representational Infrastructures. While NetLogo Web itself has enabled us to support a wide range of activity designs, we have also identified additional representational capabilities that are attractive for new GbCC activities. These include a Mapping and Geotagging Interface (using Leaflet); a Cartesian and Euclidean environment (using GeoGebra); a Data Modelling interface (using CODAP); and a Physics Engine, with force and collision simulation (using Box2d). Extensibility of NetLogo Web and the GbCC system allows us to integrate these other open-source, browser-based tools. If time permits, we will show how these environments are integrated with GbCC to permit smooth interoperation between NetLogo Web and the representations and interactions they support.

Conclusion

Cooperative and collaborative learning approaches have long been heralded as a powerful resource for educators to use in supporting student achievement (Johnson & Johnson, 2009), fostering dialogic interaction and shared meaning-making (Stahl, 2006), and encouraging equitable participation (Cohen, 1994). Constructionist approaches offer learners environments for engaging authentically with the *powerful ideas* of a discipline, creating personally meaningful artefacts that express their perspectives, and their understanding, and their interests (Papert, 1980; Papert & Harel, 1991). Generative design bridges the traditions of cooperative and constructionist learning, carefully formulating activity environments to achieve 100% participation and to enable the classroom group to accomplish explorations and produce insights that the individual members of the group could not accomplish alone. Group-based, generative activities can take many shapes, and the GbCC project and system aims to provide technological support for a wide range of these. In this workshop participants will learn to create their own GbCC activities, and in the process, will engage with some of the important theoretical and practical questions of group-based teaching and learning.

References

- Brady, C., Holbert, N. R., Novak, M., Soyulu, F., & Wilensky, U. (2015). Sandboxes for model-based inquiry. *Journal of Science Education and Technology*, 24(2-3): 265-286.
- Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilensky, U. (2017). All Roads Lead to Computing: Making, Participatory Simulations, and Social Computing as Pathways to Computer Science. *IEEE Transactions on Education*, 60(1), 59-66.

- Brady, C., Weintrop, D., Anton, G., & Wilensky, U. (2016). Constructionist Learning at the Group Level with Programmable Badges. *Proceedings of the Constructionism 2016 Conference*.
- Cohen, E. G. (1994). Restructuring the classroom: Conditions for productive small groups. *Review of educational research*, 64(1), 1-35.
- Colella, V. (2000). Participatory simulations: Building collaborative understanding through immersive dynamic modeling. *Journal of the Learning Sciences*, 9(4), 471-500.
- Colella, V., Borovoy, R., & Resnick, M. (1998, April, 1998). *Participatory simulations: Using computational objects to learn about dynamic systems*. Paper presented at the Computer Human Interface (CHI) 1998 Conference, Los Angeles, CA.
- Johnson, D. W., & Johnson, R. T. (2009). An educational psychology success story: Social interdependence theory and cooperative learning. *Educational researcher*, 38(5), 365-379.
- Mehan, H. (1980). The competent student. *Anthropology & Education Quarterly*, 11(3), 131-152.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Papert, S. & Harel, I. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 1-12). Norwood, NJ: Ablex Publishing Corp.
- Stahl, G. (2006). *Group cognition: Computer support for building collaborative knowledge* (pp. 451-473). Cambridge, MA: MIT press.
- Stroup, W., Ares, N., & Hurford, A. (2005). A dialectic analysis of generativity: Issues of network-supported design in mathematics and science. *Mathematical Thinking and Learning*, 7(3), 181-206.
- Stroup, W. M., Ares, N., Hurford, A., & Lesh, R. A. (2007). Diversity by design: The what, why and how of generativity in next-generation classroom networks. In R. A. Lesh & J. J. Kaput (Eds.), *Foundations of the future: Twenty-first century models and modeling*. New York, NY: Lawrence Erlbaum.
- White, T., Brady, C., Huang, J., & Stevens, M. (in review). A Distributed-by-Design Approach to Supporting Collaborative Learning with Dynamic Mathematics Software. *Educational Designer*.
- Wilensky, U. (1999). NetLogo (Version 6.0.3) [Software]. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University.
- Wilensky, U., & Stroup, W. (1999a). NetLogo HubNet [Software]. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University.
- Wilensky, U., & Stroup, W. (1999b). *Learning through participatory simulations: Network-based design for systems learning in classrooms*. Paper presented at the International Conference of the Learning Sciences (ICLS), Atlanta, GA.
- Wittrock, M. C. (1991). Generative teaching of comprehension. *The Elementary School Journal*, 92(2), (169-184).

WS4: AI Programming in Snap!

Ken Kahn, toontalk@gmail.com

Department of Education, University of Oxford, UK

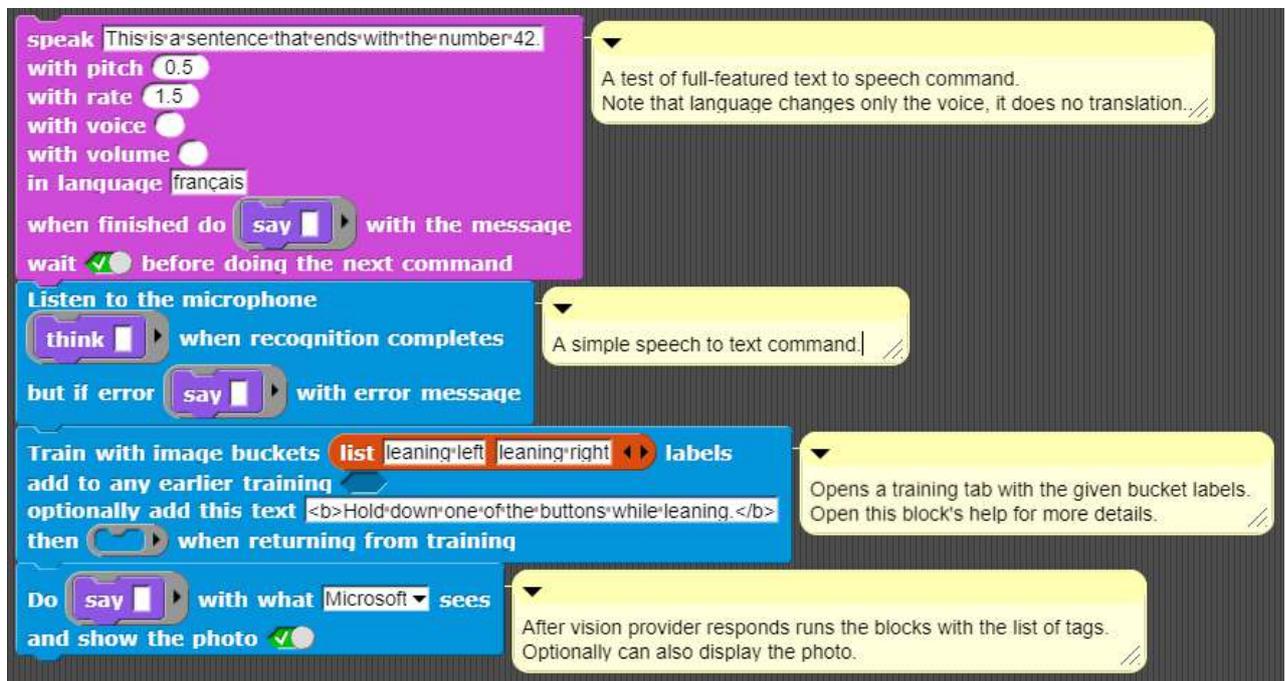
Abstract

We have added new blocks to Snap! for speech synthesis, speech recognition, image recognition, and machine learning. These run in a Chrome browser without any extensions or installation. An interactive guide has been developed. Twelve sample programs are available. During the workshop participants will explore these new blocks. They will have the opportunity to modify the sample programs. Familiarity with Snap! or Scratch will be very helpful but not absolutely necessary. While this software can run on tablets and smartphones we recommend you bring a laptop.

More details can be found in the full research paper with the same title in these proceedings.

Keywords

Visual programming; machine learning; block languages; Snap!, AI services; cloud services; speech synthesis; speech recognition; image recognition;



Speech Synthesis and Recognition

New Snap! blocks will be explored that build upon the browser's Web Speech API (Mozilla 2018). The most complex of these blocks enables control over the rate, pitch, volume, voice and language of the generated speech.

The Web Speech API also supports speech recognition. New Snap! blocks enable programs to respond to speech after it has been recognised. The most complex version enables one to receive partial results as speaking is still ongoing. It also supports the specification of the language being recognised. One can ask for several alternative interpretations of what was spoken and the confidence the system has that they are correct.

Image Recognition

There is no web API standard for image recognition. However, many companies provide web-based vision cloud services. The new Snap! blocks can send image recognition queries to Google, IBM, or Microsoft. While these are commercial services all the providers have free quotas for limited use. In addition to being able to obtain image labels and their confidence scores the most advanced Snap! blocks provide access to other information these services provide such as locations of faces, image properties, and more.

Machine Learning

Snap! blocks could be created to use machine learning web services. We, instead, use the tensorflow.js library (tensorflow.js 2018) to perform machine learning locally in the browser. This library is able to use the GPU (graphics processing unit) to run very fast. Currently the new Snap! blocks are limited to training to classify images. Programs using these blocks can interleave training (either from the camera or from sprite costumes) and classification of new images. A sample program using these blocks implements a Rock Paper Scissors game where players make moves by configuring their hand in front of the camera.

Conclusion

Creating AI programs in Snap! can be fun. One can learn about perception and machine learning. And one can be creative in making programs that do very impressive and interesting things.

References

tensorflow.js (2018) <https://togetherjs.com/>.

Mozilla (2018) Speech Synthesis API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API.

WS5: Developing Affordable STEM Maker Projects with BBC Micro:bits and Microsoft MakeCode

Stephen Howell,
Stephen.Howell@microsoft.com
Microsoft Ireland

Neeltje Berger,
Neeltje.Berger@microsoft.com
Microsoft Research

Peter Heldens,
Peter.Heldens@microsoft.com

Microsoft Netherlands Kevin Marshall,
kevmar@microsoft.com
Microsoft Ireland

Clare Riley, *Clare.Riley@microsoft.com*
Microsoft UK

Abstract

In this hands-on workshop, we will challenge the participants to build 3 projects including wearables and other physical artefacts using affordable microprocessors (BBC Micro:bits) and common arts and crafts materials.



A typical bill of materials including BBC Micro:bits, batteries, paper plates and cups, crocodile leads, Velcro and cardboard coffee cup holders

Each project was devised by a multi-disciplinary team of teachers and industry educational technologists in an initiative to develop free, high-quality resources with a depth of content that could be taught in a classroom but breadth of accessibility so that students working at home (without a mentor or educator to guide them) could complete the projects.

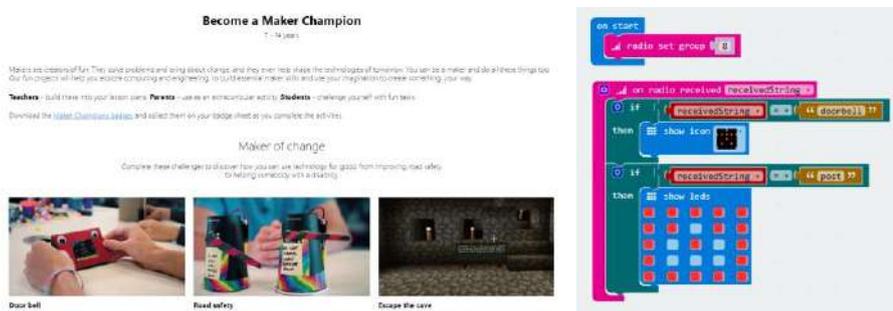
Keywords

constructionism; STEM; computational thinking; inclusive design; teacher training

Workshop

Sophisticated tools have become a common purchase for educators wishing to teach Computational Thinking, and popular microprocessor and robotics kits are expensive when purchasing for a class. This workshop presents 3 constructionist projects that each require just one or two microprocessors (BBC Micro:bit) and some cheap maker/electronic components such as LEDs and crocodile leads. A range

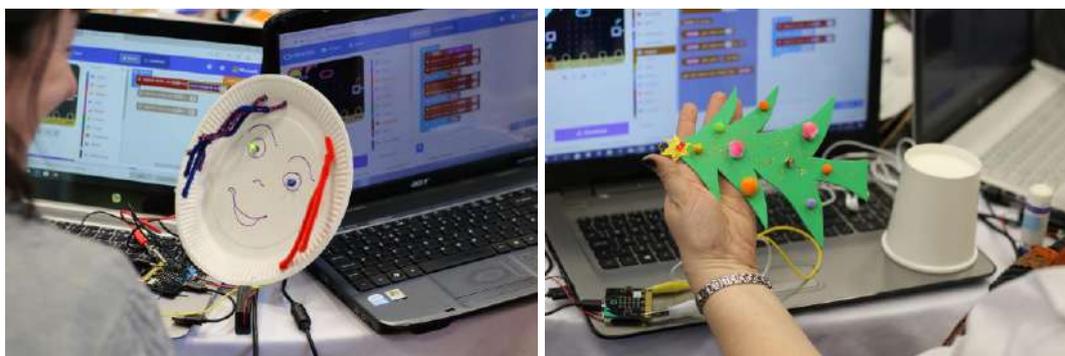
of arts and crafts, including markers, paper plates, cardboard cup holders, and Velcro make up the remainder. The development environment used to code the BBC: Micro:bits is Microsoft MakeCode (Ball, 2017). These projects were developed by teachers, researchers, and technologists from England, Scotland, Netherlands, and Ireland as part of a *Maker Champions* STEM initiative from Microsoft (Riley, 2017).



Become a Maker Champion Website and MakeCode snippet showing radio communication

Format & Participants

Participants will build a wearable to act as doorbell for a deaf user (and build the doorbell), make a mask or holiday decoration, and build a traffic light. This will be a hands-on workshop with minimal coding (MakeCode is a visual programming language) but some arts and crafts skills would be useful. The expected participants for this workshop are interested in learning about BBC Micro:bits and Microsoft MakeCode.



Teachers workshop with arts and crafts enhanced with LEDs to make masks and Christmas decorations powered and controlled by BBC Micro:bits and Microsoft MakeCode

References

Ball, T. (2017). Physical computing for everyone. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, 3. <http://dx.doi.org/10.1109/icse-seet.2017.31>

Riley, C. (2017). Digital Skills: Become a Maker Champion [Website]. Retrieved from <https://www.microsoft.com/en-gb/athome/digitalskills/discover/#maker>

WS6: NetsBlox: A Constructionist Environment for Creating Distributed Applications

Brian Broll, *brian.broll@vanderbilt.edu*
Vanderbilt University, Nashville TN, USA

Corey Brady, *corey.brady@vanderbilt.edu*
Ákos Lédeczi, *akos.ledeczi@vanderbilt.edu*
Vanderbilt University, Nashville TN, USA

Abstract

NetsBlox is a collaborative learning environment extending Snap! with carefully-selected abstractions that enable students to create distributed applications. It makes distributed programming accessible to young learners using simple yet powerful visual programming primitives, an intuitive user interface and a sophisticated cloud-based architecture. This **workshop** introduces the environment and gives participants hands-on experience with activities that demonstrate NetsBlox's utility for creating multi-player games and client-server applications that access public-domain scientific data sources. Our early work using NetsBlox with students suggests that integrating these connected features into learners' early experiences of programming widens their perspective on the role of computation in both inquiry and broader social life.

Keywords

Block-based programming; visual programming; distributed computing; network programming

Introduction

The majority of computer applications we interact with daily are distributed, that is, they involve more than one computer to function. The web, texting, Twitter, Facebook, online games, Pandora, Netflix, Amazon Echo, Siri, Google Maps and YouTube are just a few of the most popular examples. Self-driving cars, home automation and the Internet of Things (IoT) are other prime examples of distributed systems. Teaching distributed programming then constitutes both a necessity and a great opportunity. It is a necessity because distributed computing is becoming part of basic computer literacy. And it is also an opportunity because children already use the technology every day and their natural curiosity will provide excellent motivation for them to learn more about it.

NetsBlox (<https://netsblox.org>) is an open-source, web- and cloud-based visual programming environment that enables users to create distributed applications (Broll et al, 2018). NetsBlox extends the widely used Snap! environment building on its JavaScript codebase. NetsBlox supplies simple abstractions for synchronization and communication across computers to enable students to create truly engaging distributed programs.

Peer to peer communication in NetsBlox is supported by Messages. Messages are very similar to Events already present in Snap! and in Scratch. In NetsBlox, a Message is an Event that contains a structured data payload and can be sent to other NetsBlox programs across the Internet. Messages are typed and the message type is defined just like custom blocks are. The sent data shows up as appropriately named variables on the receiving end. Addressing is simple yet powerful. An example message sending and receiving block is shown in Figure 1.



Figure 1. Sending and receiving messages in NetsBlox

Remote Procedure Calls (RPC) allow for invoking code that will be executed at a remote location, and then (optionally) getting back the results of the computation. From the user’s point of view, RPCs are also similar to custom blocks. The main use for RPCs in NetsBlox is making some of the large number of publicly available interesting data sets and data-centric web services on the web available to NetsBlox programmers. Examples include maps, weather, air pollution, seismic data, astronomic image sets, a movie database, stock quotes and many others. Essentially, the NetsBlox server provides a mapping between RPCs and the corresponding public web API. Related RPCs, that is, RPCs interfacing with the same web API, are grouped into Services. For example, NetsBlox wraps Google Maps into a Service providing RPCs for getting maps, satellite images and coordinate transformations. Relying on this mapping service, it only takes a few blocks of NetsBlox code to provide a fully navigable map of the world as an interactive background.

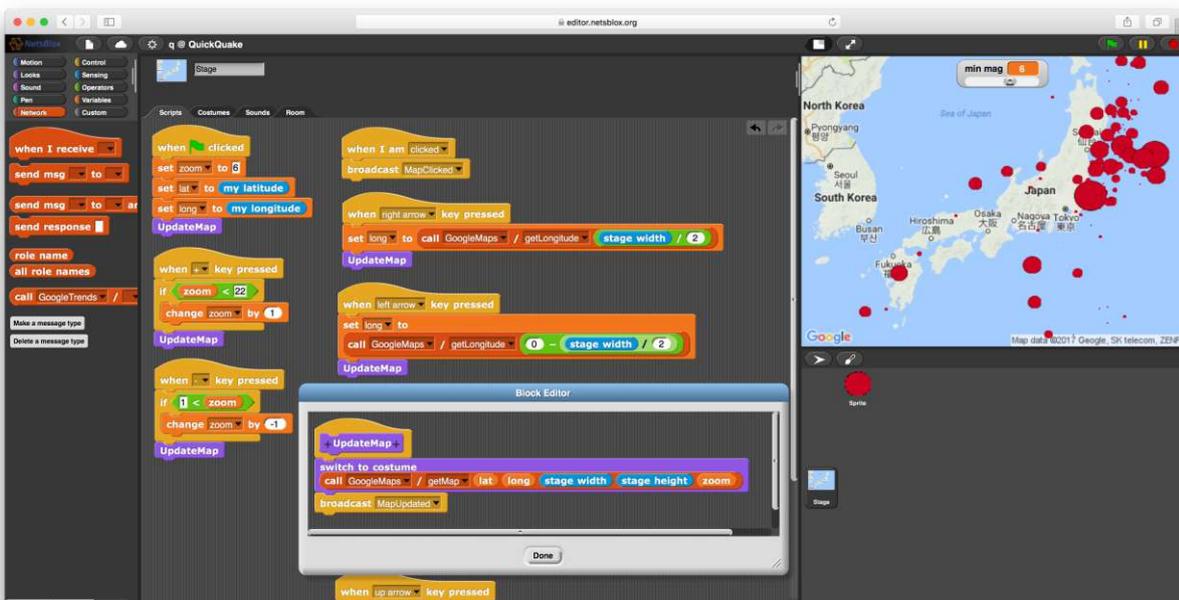


Figure 2. Earthquake project using data from Google Maps and USGS with the help of RPCs

In another example, the project shown in Figure 2 displays historical earthquake events for the anywhere on Earth. The code shown is that of the stage: the +/- keys zoom in and out, while the arrow keys pan the map. When the map is clicked, the sprite code (not shown, but it is of similar complexity to the stage scripts) requests USGS data and displays red dots whose radius is proportional to the magnitude of the given seismic event. In this case, the minimum magnitude is set to 6, so only severe quakes are shown. Notice that the project only assumes students know variables, if-statements and custom blocks from the traditional curriculum. Conceptually, the only challenging aspect is the coordinate transformation from x and y stage coordinates to latitude and longitude. But the map service provides RPCs for the actual computation.

NetsBlox also supports collaboration: students can work on a shared project from their own computers just like they would collaborating on a Google Docs document. This enables team projects and online mentoring. Collaboration support requires keeping a log of all editing events. This log also enables unlimited undo/redo as well as providing learners with the means to “replay” the entire history of the

project creation via a YouTube-like media player interface. This supports them in reflecting on the construction process and enables teachers and researchers to study the process as well.

Workshop objectives

The workshop will introduce the environment and give participants hands-on experience with a sequence of activities that demonstrate NetsBlox's utility in supporting middle and high school students in creating engaging distributed applications. A projector and Internet access, as well as several spare computers would be desirable for running the workshop. Participants may form two-person groups if desired and work on the problems together using the collaboration features of the tool. Based on the interests of the participants, a subset of the following applications will be included. Though building these applications efficiently illustrates important features of the platform in an engaging context, workshop participants will be encouraged to consider other constructionist activities that can be uniquely supported with NetsBlox.

Weather app: the program will display current conditions anywhere on Earth. The background will be a fully interactive map just like the one in Figure 2. When the user clicks on the map, the sprite costume will change to the appropriate weather icon and the current temperature and the name of the closest city will be displayed as well.

Movie app: the program will ask the user for the title of a movie and it will display photos of the three leading cast members. This project, just like the weather app, will introduce how to use RPCs in NetsBlox.

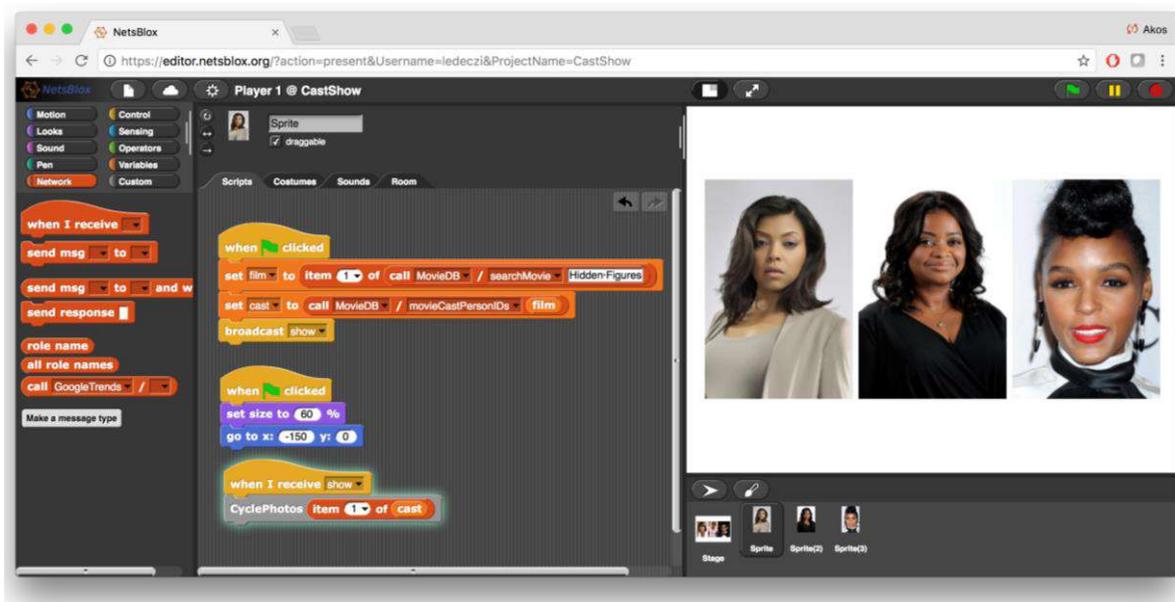


Figure 3. The Movie app

Chatroom: the presenter will create a chatroom server app, while the attendees will create a client that sends text messages to the server which it broadcasts to other members of the chatroom. This program will illustrate message passing and addressing in NetsBlox.

Shared Whiteboard: participants will implement a simple two-user drawing program. As one user draws by dragging the mouse on the stage, it also sends a list of the mouse coordinates to the other client that display the trace and vice versa. To make it more challenging, erasing the board will require a consensus of the two users.

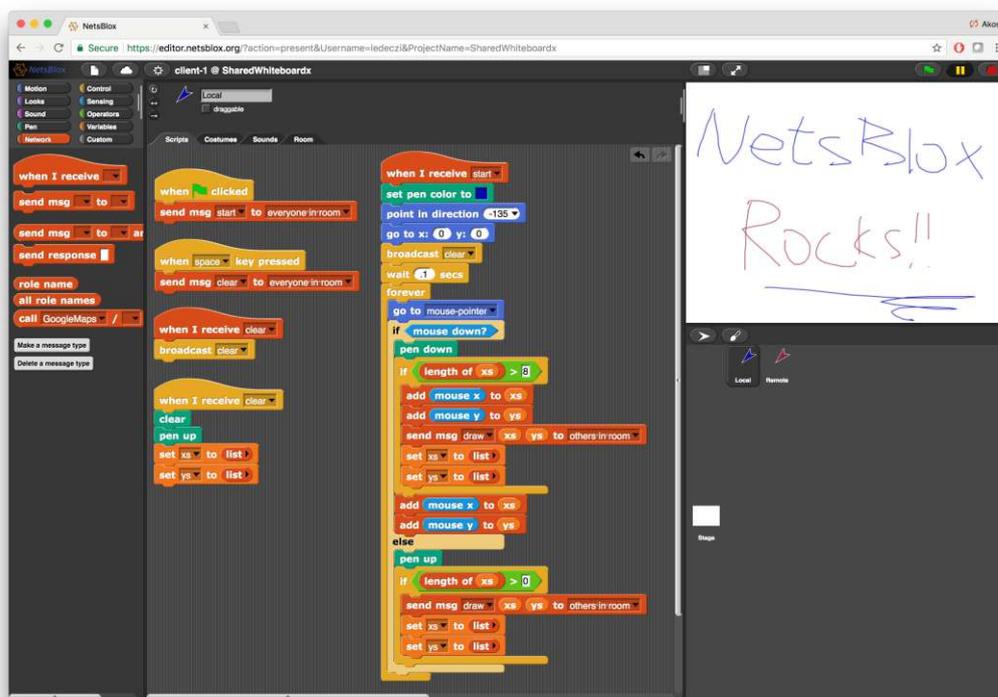


Figure 4. The Shared Whiteboard project

Mesh Network: participants will form a mesh network where everybody is only allowed to send messages to one other person (e.g., right-hand neighbour around a circular table). The goal of the project is to be able to send messages to anybody in the room by forwarding messages not meant to the given person. This will be a good opportunity to use the built-in visual network debugger of NetsBlox.

Multi-player game: attendees will select a simple, turn-based multi-player game such as Tic Tac Toe, 21 Pebbles or Ghost. Participants will have the option to utilize the N-player Game service that provides a set of RPCs and messages that facilitate the creation of turn-based games or implement the game with custom messages only.

The last part of the workshop will be a discussion in which the presenters will solicit feedback on the design of the environment and what directions its development should take.

Conclusion

The distributed programming abstractions in NetsBlox provide access to vast arrays of data on the Internet directly in the visual programming environment empowering students to create innovative science projects and simultaneously bring STEM concepts into CS education. The ability to create multi-player games and other engaging networked programs provides increased motivation and encourage young learners to be creators and not just consumers of digital entertainment. The workshop will expose attendees to the large number of possibilities created by opening the Internet to a block-based programming environment via a small set of simple but powerful abstractions.

References

Broll, B., Lédeczi, A., Zare, H., Nguyen Do, D., Sallai, J., Völgyesi, P., Maróti, M., Brown, L., Vanags, C. (2018) A visual programming environment for introducing distributed computing to secondary education, *Journal of Parallel and Distributed Computing*, ISSN 0743-7315, <https://doi.org/10.1016/j.jpdc.2018.02.021>.

WS7: The ER4STEM Repository for Educational Robotics

Annalise Duca, *annalise@acrosslimits.com*

Angele Giuliano, *angele@acrosslimits.com*

AcrossLimits Ltd, Hilltop Gardens, Triq L-Inwkina, Naxxar, Malta

Sofia Nikitopoulou, *sophieniki@ppp.uoa.gr*

Nikoleta Yiannoutsou, *nyianoutsou@ppp.uoa.gr*

Chronis Kynigos, *kynigos@ppp.uoa.gr*

University of Athens, Athens, Greece

Abstract

Type: Workshop

Using robotics in education is an engaging method for student motivation towards STEM subjects and more. Teachers, educators and researchers who are newly experimenting with the use of robots in the classroom are all asking a very similar question. “Where can I find inspiration to introduce constructionism in my teaching?”, “What can I do to teach my subject using robotics?” The answer to this is “The ER4STEM Repository” which will be full of educational resources, activity plans and suggestions for educators. “The ER4STEM Repository” has been underpinned by the basic pedagogical theory underlying its’ design in constructionism. This happens through an Activity Plan Template. This template is aligned with the Repository and provides a generic design instrument that identifies critical elements of teaching and learning with robotics based on theory and practice and is expected to contribute to the description of effective learning and teaching with robotics.

Keywords

repository; educational robotics; STEM; robots; sharing ideas; collaboration; educational robotics for STEM; educational activities; constructionism; OER

Introduction

Top EdTech stories are constantly reporting how and why educators need to start educating and preparing our kids to be able to tackle future jobs that are already changing in nature. This includes understanding how one will have to work with robots and technology in different industries, may it be medicine or in manufacturing companies.

The Educational robotics for STEM (ER4STEM) Project

The ER4STEM project is an European Funded project, and some of its’ activities are workshops on robotics for approximately 4000 students in four countries, evaluation and the development of an educational robotics repository to address the needs of the market.

The Repository

Different educational repositories already exist, however none of them are focused on providing specific educational robotics lesson plans. Further to this, many robotics kits suppliers, promote their robotic kit for the use of STEM education, however most of the time, one can only find few examples on their website. These are normally not enough for educators for many reasons. One needs to consider that in most cases, the educator need to master how to use the robotic kit first before teaching it or using it to reach a particular educational goal. Creating activity plans take a considerable amount of time. Additional to this, more resources and effort is needed to ensure usability.

The ER4STEM repository is a user-centred repository and it is based on a pedagogical activity plan template which was developed by the University of Athens. The offline version of the activity plan has been already used by universities, robotic centers and training centers for the creation of various robotics workshops and the refined version is now available as an interactive online repository.

Accessing the Repository

The repository is a user-friendly and guided portal that will encourage users to both search for existing resources while also sharing their own. The portal can be accessed from repository.er4stem.com. Figure 1 shows the landing page for the repository. Users can sign up for a free account using either Facebook Single sign-on or using their email address. The portal is available in 7 languages (English, Bulgarian, Czech, German, Greek, Turkish and Italian).



Figure 1. Landing page for the ER4STEM repository

The Activity Plans in the Repository

Logged in and guest users can easily search for different open resources. This can be done with any keywords, or else using an advanced search which gives the additional opportunity to the user to search by: subject, age and language. All users can share any activity plan via social media, and also download an offline version in PDF format. When logged in, users also have the option to mark any activity as a favourite, so they can easily find it again later on.

The activity plans are split structurally in different sections, to ensure ease of use. Each activity plan is split in; Outline Information; Objectives and Skills; Artifacts; Who? Where? How long?; Interactions; Technology; Learning and Teaching; How To; Assessments; Supporting Materials and Feedback (from other users). Simple terminology was used to ensure that even those whose English is not their native language, find it easy to navigate and use the repository.

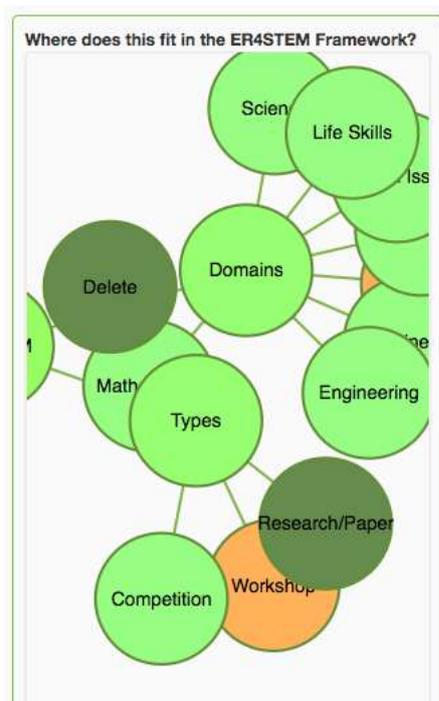


Figure 2. ER4STEM Framework on the ER4STEM Repository

Figure 2 also illustrates how with the use of an interactive diagram, one can visually see where the activity plan fits in relation to the ER4STEM framework.

Submitting an Activity Plan

Once logged in, users will have the option to create and submit their own activity plans. This process is split into 9 different steps. In each step, detailed information is given to the user to ensure that the activity plan is written in line with the ER4STEM framework. Our aim for the creation of an activity plan into the repository is to find balance between a) a level of abstraction that will make the template adaptable to different settings and b) a level of detail that will demonstrate the influence of a specific pedagogical approach, it will address the particularities of Robotics and it will augment the affordances of the specific robotic kit used in each activity plan. Therefore, when the users create their own activity plan they are tacitly and implicitly guided to think as pedagogical expert. This method ensures that the pedagogy aspect of the activity plan is not lost, yet at the same time users do not need to be familiar with the framework per se to use it as it will guide their input step by step. Some of the fields required to create the activity plan include: Learning Outcomes: Subject Related, Skills Learning Outcomes, Actions, Relationships, Roles in the group, Support by the tutor/s, Digital Artifacts, Robotic artifact, Gender, Age Group, Class Size, Group Size, Grouping Suggestions, Prior Knowledge, Special needs and abilities, Environment, Style of Room, Duration, Technology Used, Price per Kit, Programming Languages, Technology Needed, Activity Blocks and any Supporting Material.

When a user fills these fields we ensure that once available, anyone reading the activity plan, can follow it and implement the activity plan accordingly in their own classroom environment. All activities submitted are reviewed by our team of experts to ensure that the quality is maintained.

The Blocks

Activity blocks are a new construct focusing mainly on the practical aspect of the activity plan. Thus, the role of Blocks is to help users create activities in the classroom by giving them concrete ideas on the section of activity plan “how to”. In several cases, activity blocks are accompanied by relevant worksheets that are designed to facilitate the implementation of the activity. Consequently, the activity blocks were designed as a result of the first and second year evaluation of the project, aiming to support

the fostering of creativity, collaboration and reflection and to offer multiple entry points for the activity plans.



Figure 3. Activity Blocks that can be added to the activity plan

As in figure 3 below, previously used and tried “blocks” are presented to the user, to either arouse curiosity or implement as is. These blocks are filtered according to their type. Types found are Introductory, Reflection, Assessment, Evaluation, Exploration, Experimentation and Construction.

The Workshop

During this 90-minute workshop, we will be doing an introductory session to show how the repository functions, following by an interactive session where participants will be brainstorming on activity plans that they can create. Participants should ideally have a laptop or tablet during this session.

Conclusion

The ER4STEM repository answers a need in the educational market for a simple one stop online shop to find teaching materials and inspiration for teachers that want to use robotics in their subjects. All materials are open and free of charge, and with constant inputs from the teaching community they will continue to increase in number and variety spanning all subjects and ages.

Acknowledgements

This work is funded by the European Commission through the Horizon 2020 Program (H2020, Grant agreement no: 665972). Project Educational Robotics for STEM: ER4STEM.

References

- <https://www.inc.com/greg-satell/we-need-to-educate-kids-for-the-future-not-the-past-heres-how.html>
- <http://www.edtechupdate.com/2017/robotics/trends/?open-article-id=7667685&article-title=-robots-are-coming-for-your-children-&blog-domain=hackeducation.com&blog-title=hack-education>

Papert, S. Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc. (1980).

Nikoleta Yiannoutsou, Sofia Nikitopoulou, Chronis Kynigos, Ivaylo Gueorguiev & Julian Angel-Fernandez. "Activity Plan Template: a mediating tool for supporting learning design with robotics". 7th International Conference on Robotics in Education, 14-16 April 2016, Austria.

Demonstrations

The Relationship between Computer Programming and English Language Skills

Nevin Akcay, *nevin.akcay@yahoo.com*

Gumuspala Vocational and Technical High School, Istanbul, Turkey

Hulya Avci, *hulyaavci@beykent.edu.tr*

Beykent University, School of Foreign Languages, Istanbul, Turkey

Ali Gungor, *ali.gungor@bau.edu.tr*

Bahcesehir University, Vice President, Istanbul, Turkey

Tufan Adiguzel, *tufan.adiguzel@es.bahcesehir.edu.tr*

Bahcesehir University, Faculty of Educational Sciences, Istanbul, Turkey

Abstract

Computational thinking is a problem-solving method applying computer science techniques and encourages programming or coding skills among children. A programming or coding language is a special language for describing computation and expressing a set of instructions on what tasks a computer needs to execute. There has been an ongoing discussion in the literature whether programming languages with a vocabulary of keywords are based on English. However, there is not much literature on how computer programming affects other aspects of life beyond the skills acquired. In this study, we demonstrated the current literature on the relationship between programming and English language skills and the methods of teaching both languages.

Keywords

Computer programming, coding, English language skills, learning

Introduction

Information technology has gained value and recognition widely in the fields of education, economics, health, agriculture, social life and entertainment (Uzunboylu, 2005). In the last few years, with the rapid development of information technology in almost all areas, having coding and programming skills is expected to become more fundamental than ever for individuals of all sectors in the 21st century (Uzunboylu, 2005; Sayin & Seferoglu, 2016). "Today, computer programs are genetic code of our world and many educators are starting to think that students need more than a passing knowledge of computer coding" (Gow, 2015, p. 20). While the internet continues to expand, computer programming and coding continue to play an important role in education (Uzunboylu, 2005; Balanskat & Engelhardt, 2015).

According to the Grand Coalition for Digital Jobs, there might be a shortage of 900.000 ICT professionals in Europe and in the ICT related sectors in the next ten years (Burke, 2013). However, over the past year, a number of non-profit organizations have attempted to investigate coding with innovative training approaches and looked for people who can write code to meet their urgent needs in many workplaces (Sayin & Seferoglu, 2016). In recent years, many governments around the world have begun to modify the curriculum in schools to foster the improvement of computational thinking of primary and secondary students through computer programming or coding (Moreno & Robles, 2015). The purpose of integrating coding into curriculum is to equip students with skills such as problem-solving and logical thinking skills that are increasingly crucial within the current digital community (Balanskat & Engelhardt, 2015). Additionally, these skills are beneficial not only for computer scientists but for anyone, regardless of personal features, interests or profession (Resnick, 2013). The use of computer programming in schools as an educational tool to improve learning in other disciplines is becoming highly prevalent at all levels of education in many countries (Moreno & Robles, 2015).

The language used in programming is not understood by an untrained person (Computer Programming) In the process of learning coding, people not just learning the coding, but they are also coded to learn. In addition to that, they have the opportunity to learn many other abilities, for example, problem solving strategies, programming, logical thinking, algorithmic and analytical thinking, and the ability to transmit ideas regardless of the programming language selected (Ersoy et al., 2011; Resnick, 2013). "Teaching of programming logic is the first and most important phase of programming teaching" (Arabacioglu et al., 2007, p. 193). The primary goal is to promote coding rather than teaching and using it as a tool to improve other skills (Sayin & Seferoglu, 2016). Factors such as student attitudes and perceptions, consistency of selected programming language and aims of foreign language education programs might influence students' success in programming and transfer of skills to other domains (Sayin & Seferoglu, 2016; Arabacioglu et al., 2007; Ersoy et al., 2011).

As English is the dominant language, but isn't a prerequisite, in the IT world, the majority of programming languages are based on keywords in English (King, 2015; Basak, 2016), some of which are declarative and mark-up languages like HTML and XSLT, and query languages like SQL. Java and C++ are also English-based computer programming languages which are provided in introductory programming courses in many colleges (Krcan & Bilobrk, 2011). Therefore, many programmers can learn little English while studying programming (Basak, 2016). Students who are competent in many subjects are sometimes incapable of succeeding at programming. On the other hand, some students who perform well in computer programming classes can also become good at English courses (Moreno & Robles, 2015).

Most programming languages look like Romaic to an untrained person at first glance (Malan & Leitner, 2007). Although they have been generated in a rigorous and artificial way instead of naturally evolving, they precisely embody linguistic knowledge as well as other features shared with human languages (King, 2015). Unless prospective programmers have adequate knowledge about these linguistic varieties including the study of meaning (semantics) and sentence structure (syntax) as well as terminology in their native language, the complexity of programming increases promptly. As for experienced programmers who are quite competent in the first language, their second language acquisition must have been easier (Riker, 2010). To illustrate, considering the use of prepositions including "from," "through," "to," and so on, COBOL language could be notably similar to English language. However, the most legible and English-like language is Smalltalk (Kenneth, 2016). The syntax of Smalltalk consists of `<object> <message>`, where `<message>` which is a command or operator the user sends to an object, which can be anything from numbers and strings to classes and code blocks. The `<message>` can be unary, binary, or keyword, the latter taking on a parameter as following:

```
renderer: = three WebGLRenderer new.
```

```
renderer setSize: (window innerWidth) height: (window innerHeight) (Kenneth, 2016).
```

Scratch, another simple object-oriented programming language generated by MIT Media Lab's Lifelong Kindergarten Group (2013), has aroused interest among many teachers aspiring to promote coding in their English language classes as it might facilitate the second language acquisition (Quan, 2015). In a research project, students participated in a story writing activity using Scratch in a writing workshop. The findings suggested that students were motivated to progress in learning English and developed positive perception towards English, besides enhancing their digital literacy and language content areas (Quinn & Kafai, 2012). According to a study on coding, most students consider that coding has a positive effect on not only learning English but also improving other essential skills such as teamwork (Moreno & Robles, 2015). Furthermore, computer programming may have a positive impact on students' learning outcomes in English language skills.

Conclusion and Discussion

As a result of literature review, as the programming languages are mostly English, and especially share certain similarities in terms of English syntax, one (Scott, 2015) says that English learning skills of students improve while there are also studies and interpretations (Moreno & Robles, 2015; Kenneth,

2016) that indicate otherwise. In the literature review, several results of the studies were found as following:

As students who know the programming language learn programming, they can easily comprehend English sentences (Moreno & Robles, 2015).

Learning other languages besides English gets more practical for learners of programming (Galvin, 2016).

Computer programming promotes the abilities such as logical thinking and algorithm to solve problems in many areas, regardless of the language used, and even the ability of analytical thinking (Basak, 2016; Malan & Leitner, 2007).

Even if programming languages have a specific semantics and syntax, which is written in English, the words used in programming languages will be almost equal to those who speak English and who do not speak English (Resnick, 2013).

It should be noted that some students who take a programming course might encounter certain challenges due to the programming language. Further research needs to be conducted to reinforce the learning process and to determine the underlying causes of the problems about programming languages confronted by students (White & Sivitanides, 2002). Future studies should reveal the relationship between programming and English language skills through qualitative and quantitative studies. In addition, the interrelation between programming and other fields (e.g. science and mathematics) should be examined. Thus, we can conclude there is a need for studies which determine whether the study of English and other skills is affected by the development of calculation thinking and whether the students should be able to generalize and transmit this problem-solving process to other areas.

In conclusion, the first phase of interdisciplinary study between programming and English language skills was revealed in this study. Regarding current technological innovations, information-age in 21st century, and digital traces that produce massive data in online environments which are constantly and exponentially increasing, it is thought that more work will be done in programming and coding education.

References

- Arabacioglu, T., Bulbul, H.I., & Filiz, A. (2007). A new approach to computer programming teaching. *Proceedings of the 9th Academic Informatics Congress* (pp. 193-197). Dumlupinar University, Kutahya, Turkey.
- Balanskat, A., & Engelhardt, K. (2015). *Computer programming and coding priorities, school curricula and initiatives across Europe*. European Schoolnet (EUN Partnership AIBSL), Brussels, Belgium.
- Basak, S. (2016, April 28). Re: Do people in non-English-speaking countries code in English? [Online forum comment]. Retrieved from <https://namasteui.quora.com/Do-people-in-non-English-speaking-countries-code-in-English>
- Burke, A. (2013). *Grand coalition for digital jobs aims to fill 900,000 predicted European ICT vacancies*. Retrieved from <https://www.siliconrepublic.com/jobs/grand-coalition-for-digital-jobs-aims-to-fill-900000-predicted-european-ict-vacancies>.
- Computer Programming. (n.d.). In online business dictionary. Retrieved from <http://www.businessdictionary.com/definition/computer-programming.html>
- Ersoy, H., Madran, R.O., & Gulbahar, Y. (2011). A model proposed for teaching programming languages: Robotic programming. *Proceedings of the 13th Academics Informatics Congress* (pp. 731-736). Inonu University, Malatya, Turkey.
- Galvin, G. (2016, October 13). Some say computer coding is a foreign language [Online news article]. Retrieved from <https://www.usnews.com/news/stem-solutions/articles/2016-10-13/spanish-french-python-some-say-computer-coding-is-a-foreign-language?context=amp>

Gow, P. (2015). *Teaching computer programming is back. Why now? A new culture of coding*. Retrieved from https://propertibazar.com/article/2015-computing-our-future-future-classroom-lab-european_5a31a244d64ab24ed47f3677.html

Kenneth, R. (2016, March 7). Re: What computer language is most similar to human spoken language? [Online forum comment]. Retrieved from <https://www.quora.com/What-computer-language-is-most-similar-to-human-spoken-language>

King, E. (2015). Java as a second language: Thoughts on a linguistically informed transition to typing languages IB Design. *Proceedings of the 2015 IEEE Blocks and Beyond Workshop* (pp.11-12). IEEE Computer Society, Washington, D.C., USA.

Krpan, D., & Bilobrk, I. (2011). Introductory programming languages in higher education. *Proceedings of the 34th International Convention*. MIPRO, Opatija, Croatia.

Lifelong Kindergarten Group, MIT Media Lab (2013). SCRATCH 2.0 (programming language).

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *SIGCSE Bulletin*, 39, 1, 223-227.

Moreno, J., & Robles, G. (2015). Computer programming as an educational tool in the English classroom. *Proceedings of the Global Engineering Education Conference (EDUCON) IEEE* (pp. 961-966). Tallinn University of Technology, Tallinn, Estonia.

Quan, G.C. (2015). Student teachers evaluating and assessing SCRATCH in the applied linguistics classroom. *Social and Behavioral Sciences*, 174, 1450-1456. doi:10.1016/j.sbspro.2015.01.774

Quinn, B., & Kafai, Y. B. (2012). The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 433-438). ACM, New York, NY, USA.

Resnick, M. (2013). Learn to code, code to learn. How programming prepares kids for more than math. Retrieved from <https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn>

Riker, A. (2010). *Natural language in programming an English Syntax-based approach for reducing the difficulty of first programming language acquisition* (Doctoral dissertation). Retrieved from <http://hdl.handle.net/10192/23861>

Sayin, Z., & Seferoglu, S.S. (2016). Coding education as a new 21st century skill and effect of coding on educational policies. *Academic Informatics*. Adnan Menderes University, Aydin, Turkey.

Scott, B. (2015, April 9). Is the English language too important to learn a programming language? [Online forum comment]. Retrieved from <https://www.quora.com/Is-the-English-language-too-important-to-learn-a-programming-language>

Uzunboylu, H. (2005). *The effectiveness of web-assisted English language instruction on the achievement and attitude of the students* (Unpublished doctoral dissertation). Ankara University, the Institute of Educational Sciences, Ankara.

White, G. L., & Sivitanides, M. P. (2002). A theory of the relationship between cognitive requirements of computer programming languages and programmers' cognitive characteristics. *Journal of Information Systems Education*, 13, 1, 59-66.

Synthesizing the Mesh: Using Constructible Authentic Representations to Gain Intuitive Understanding of Bayesian Reasoning

Monica Chan, monica.chan@tc.columbia.edu
Teachers College, Columbia University, USA

Gary C. F. Lee, garyleecf@mit.edu
Massachusetts Institute of Technology, USA

Abstract

In recent years, there has been a massive push towards building basic computer science knowledge and skills at K-12 levels. However, less has been done towards introducing more complex and application-based fields of computer science, such as machine learning, and connecting these fields with mathematics and statistics education. This game *Synthesizing the Mesh* has been developed to provide a constructionist learning platform that helps upper middle to high school students gain intuition and understanding for Bayesian reasoning, an inherent algorithm of machine learning that is traditionally perceived as an advanced topic. The design principle Constructible Authentic Representations (CAR) has been used to guide the game design process.

Keywords

game design; middle school; high school; constructionism; problem-solving; strategy; bayesian reasoning; technology

Introduction

Bayesian reasoning is one of the most fundamental algorithms of machine learning methods, and has been a powerful approach for describing uncertainty and conditional probability. However, the topic of Bayesian reasoning is typically avoided in introductory undergraduate mathematics (statistics) and computer science education (Satake & Murray, 2014), and hardly taught at all in high school mathematics or computer science classes, due to the perception that Bayesian reasoning is an advanced topic. However, the applications of Bayesian methods are widespread in everyday life, found in fields such as healthcare, law, manufacturing, etc.

Although there has been a larger push for K-12 computer science education in the recent years (K-12 Computer Science Framework; Next Generation Science Standards, 2018), there is still a lack in the use of mathematics and statistics at K-12 levels to teach and learn essential concepts related to fundamental applications of computer science. This proposed educational game named *Synthesizing the Mesh* uses constructible authentic representations as a design principle. The game aims to provide a constructionist learning platform, where upper middle to high school students can gain an intuition about the overarching concept of how Bayesian reasoning works. This is an effort to spur students' interests not only in learning about basic programming methodology, but also in delving more deeply into the logic behind computer science algorithm development.

Literature Review

Prior Work

In the past years, researchers have begun developing constructionist approaches to teach STEM concepts that are traditionally perceived as higher-level or advanced. For example, *MaterialSim* was developed as a computational constructionist learning environment that aided undergraduate students in learning materials sciences concepts (Blikstein & Wilensky, 2009).

In the field of mathematics, and more specifically statistics, *Equidistant Probability* was a NetLogo microworld with a series of computer-based probability experiments targeted at helping middle school students gain intuition in probability and statistics (Abrahamson & Wilensky, 2003). Weintrop and colleagues (2016) had also conducted a study on the development of computational thinking through constructionist online/video games. Such games combine features of traditional video games with constructionist learning and design theories to create a core playing experience around building artifacts in the game environment (Weintrop et al., 2016). We have used the discoveries about player/learner experiences from these previous studies to create *Synthesizing the Mesh*, with a focus on Bayesian reasoning as the choice of the advanced level topic that this educational game concentrates on.

Constructible Authentic Representations

Constructible Authentic Representations (CAR) is a design principle that may be incorporated into educational game design to engage players in constructing artifacts that are epistemologically aligned to representations used in the target domain (Holbert & Wilensky, 2014, p. 53). A main objective of this design principle is to encourage the player to connect seemingly disparate knowledge resources to gain a higher probability that in-game experiences would be co-activated with formal representations in non-game contexts (Holbert & Wilensky, 2014, p. 57). Building upon the principle of concept integration (Cheng, 2011; Kafai, 2012), games designed and guided with CAR emphasize on the player's interaction with the game in creating meaning out of the concepts that the game accentuates.

CAR is also a build-on from the concept of computation-based restructurations (Wilensky & Papert, 2010), whereby computational objects (agents) replace traditional mathematical or scientific representations to extend the learner's understanding of a certain topic or idea. These computational objects that are restructured typically have "power properties" that attract domain-experts, but also have "cognitive, affective, social and diversity properties" that make it accessible to the wider public (Wilensky & Papert, 2010, p. 4).

Synthesizing the Mesh Game Design

Currently, *Synthesizing the Mesh* has been built using a HTML/CSS/JavaScript platform, guided by the CAR design principle. Specific game design features and mechanics include the use of different types of sensors with varying costs and coverage range, different types of bugs that multiply at various rates or probabilities, and different player choices on whether or not to perform predictive maintenance that significantly reduces chances of bugs resurfacing. These decisions to implement certain choices are governed by the amount of cash units each player has.

The premise of Bayesian reasoning hinges upon a concept in probability, Bayes' theorem, where one's belief about a hypothesis (or probability that the hypothesis is true) is updated as more evidence is observed (D'Agostini, 2003). The game mechanic models this form of reasoning, where players attempt to identify locations of the bugs based on the observations of the sensors in the neighborhood. Further, strategizing the locations and types of sensors may also help provide evidence about the guesses to differing degree. The game is collaborative yet also competitive – players work together to unearth all the bugs in the game, but also aim to accurately find the most bugs while optimizing their sensor placements and minimizing cost.

Learning Goals

Listed below are the learning goals of *Synthesizing the Mesh*:

1. To cultivate a more holistic intuitive understanding of the method of Bayesian Reasoning: based on observations (sensors in the game), figure out the probability of outcomes/guesses (bugs in the game)
2. To gain more confidence about approaching thought processes that may be unfamiliar
3. To learn to collect data strategically and collaboratively to achieve a communal goal

Game Play

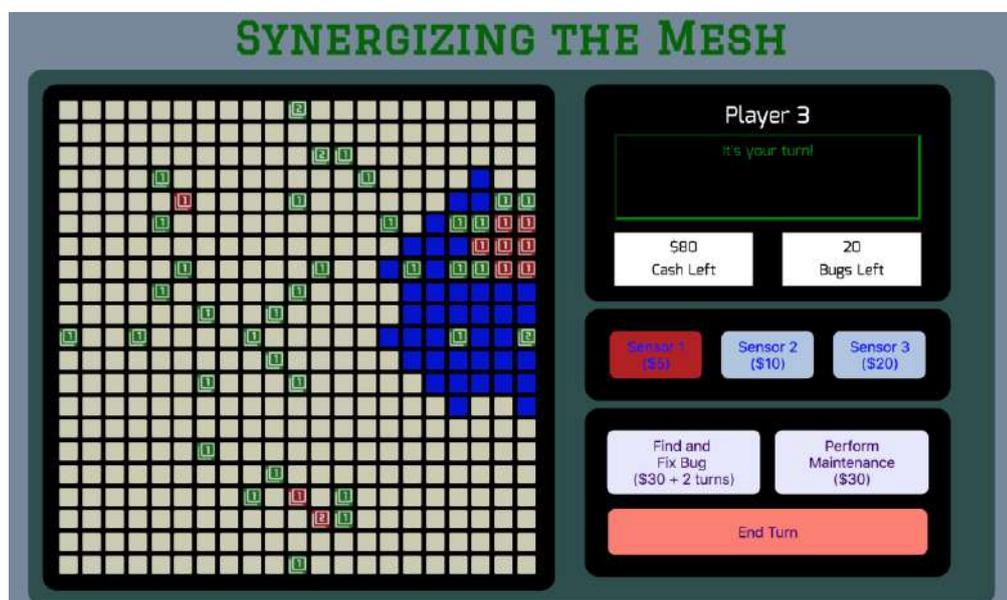
Each player starts with \$100, and begins by setting sensors of his/her choice on the blue spots on the game board. Player 1 begins from the left, Player 2 from the top, Player 3 from the right, and Player 4 from the bottom of the 21x21 square grid. The sensor placed on the board will turn green if there is no bug (machine bug) in its scope, and will turn red if there is a bug present in the vicinity. Each player is able to end his/her turn anytime and then the next player goes.

Various game mechanics are listed below:

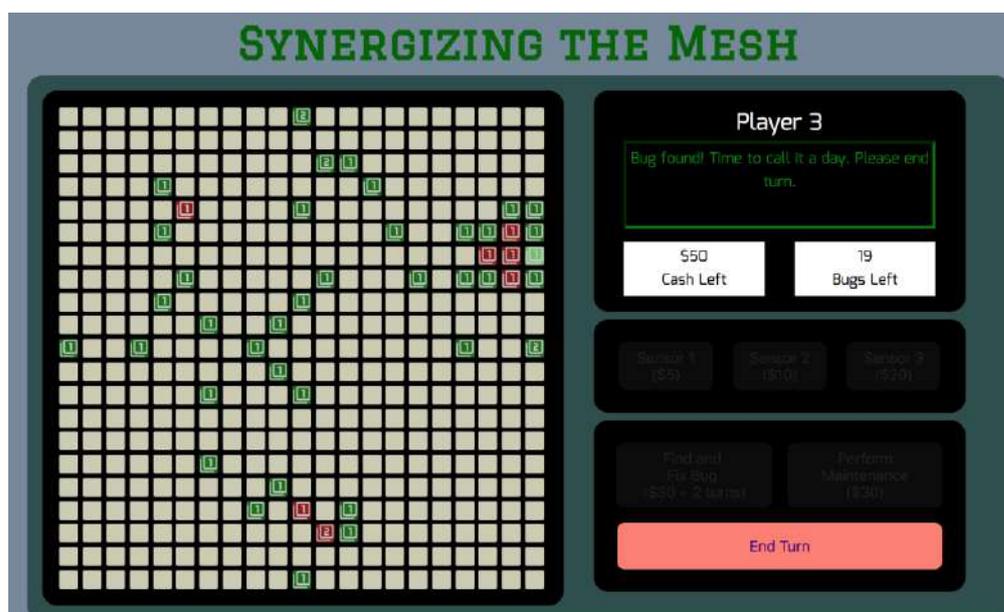
- **Players** need to spend money implementing sensors to find where bugs are, but they have a limited amount of money
- **Sensors** come in 3 different levels of functionality and scope (Sensor 1 has the smallest scope and is the cheapest, while Sensor 3 has the widest scope and is the most expensive)
- **Find and Fix Bug** option fixes the bug temporarily - the bug could reappear
- **Perform Predictive Maintenance** option reduces chances of the bug reappearing
- **Goal:** Reach “bug-less” state (~less than 5% chance of bug reappearing)

While designing the mechanics, Squire and Barab’s study of urban underserved students learning history via *Civilization III* (2004) informed our game design. Squire and Barab stated that “Learning occurred through recursive cycles of failure and revising strategies, which led to frustration, engagement and learning” (2004, p. 1). This connection between the cycles of strategies (which we considered as game mechanics in this case) and the frustration and engagement (which we considered as game dynamics) is a great way of anticipating strategies and implementations that other players might do during the game. Just as how Squire and Barab had guided the students in their study through historical simulation and discussion about hypothetical history via the game *Civilization III*, we intended to add mechanics that allow *Synthesizing the Mesh* to not only be a tool for understanding the underlying concept of Bayesian reasoning, but also to contain mechanics that affect interactions and anticipation amongst players’ thoughts, players’ moves, and any further relationships players might think of while identifying and fixing the “bugs” in the game to extend their fundamental conceptual understanding. This is not an emulation of mechanics between *Civilization III* and *Synthesizing the Mesh* because they have gaming natures and methods that are inherently different from each other, but we tried to follow the expansion of educational scope beyond the game simulation, which Squire and Barab had demonstrated in their research study (2004, p. 5).

Below is how the current game board prototype looks, with an example of how a bug is found:



This figure shows red sensors, meaning a bug is in the vicinity of the red sensors. These clues that the player has to generate will point the player closer to identifying the bug. The blue squares on the grid indicate where Player 3 can put a sensor next.



This figure shows that the player (Player 3) has successfully identified this bug (in light green). The dashboard also changes to read “Bug found!”

Future Work

Future work includes completing building out the game aesthetics of *Synthesizing the Mesh*, to include possible real-world settings in the game, in order to connect the player’s in-game experience to representations that would also be personally meaningful outside the game. Upon completion of *Synthesizing the Mesh*, we have plans to implement this in an after-school supplementary program to examine how students interact with the content and representations in the game. We are interested in studying how students gain intuition and understanding for traditionally complex and advanced topics in machine learning concepts, specifically Bayesian reasoning in this case, thus we will perform a pre- and post-test outside the game, and collect in-game observational data from this research project.

References

- Abrahamson, D., & Wilensky, U. (2003, February). The quest of the bell curve: A constructionist approach to learning statistics through designing computer-based probability experiments. In *Proceedings of the Third Conference of the European Society for Research in Mathematics Education*.
- Blikstein, P., & Wilensky, U. (2009). An atom is known by the company it keeps: A constructionist learning environment for materials science using agent-based modeling. *International Journal of Computers for Mathematical Learning*, 14(2), 81-119.
- Cheng, P. C. H. (2011). Probably good diagrams for learning: representational epistemic recodification of probability theory. *Topics in Cognitive Science*, 3(3), 475-498.
- D'Agostini, G. (2003). *Bayesian reasoning in data analysis: A critical introduction*. World Scientific.
- Holbert, N. R., & Wilensky, U. (2014). Constructible authentic representations: Designing video games that enable players to utilize knowledge developed in-game to reason about science. *Technology, Knowledge and Learning*, 19(1-2), 53-79.
- K–12 Computer Science Framework. (n.d.). Retrieved March 29, 2018, from <https://k12cs.org/>
- Kafai, Y. B. (2012). Learning design by making games: Children's development of design strategies in the creation of a complex computational artifact. In *Constructionism in practice* (pp. 87-112). Routledge.
- Next Generation Science Standards. (2018, March 08). Retrieved March 29, 2018, from <https://www.nextgenscience.org/>

Satake, E., & Murray, A. V. (2014). Teaching an Application of Bayes' Rule for Legal Decision-Making: Measuring the Strength of Evidence. *Journal of Statistics Education*, 22(1).

Squire, K., & Barab, S. (2004, June). Replaying history: Engaging urban underserved students in learning world history through computer simulation games. In *Proceedings of the 6th international conference on Learning sciences* (pp. 505-512). International Society of the Learning Sciences.

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning (IJGBL)*, 6(1), 1-17.

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulations of knowledge disciplines through new representational forms. *Constructionism*.

Teaching Computational Thinking with Minecraft & Microsoft MakeCode

Stephen Howell, stephen.howell@microsoft.com
Microsoft Ireland

Abstract

In this demonstration, we present Microsoft MakeCode, an open-source visual programming language that can be used to teach Computational Thinking concepts with Minecraft. With MakeCode, we will show how Minecraft can be used as the microworld for exploring constructionist ideas. Students can design shapes and structures to be built algorithmically. Programmable agents can be instructed to navigate, act and explore, and it all takes place in a virtual world that many students find comfortable and familiar because of the success of Minecraft as a game.



A house created in Minecraft with a snippet of code that builds the walls using a nested for-loop

Keywords

computational thinking; microworlds; visual programming languages

Demonstration

In Minecraft, players must survive by collecting resources and carefully utilising them to ensure safety when the monsters, or 'mobs', come (they mostly come at night). Luckily for educators, Minecraft also has a 'creative mode', where there are no monsters and teachers can bring students to play together without the ability to injure each other. This version of Minecraft is called the 'Education Edition', and this edition was the first to add the ability to use a visual programming language to build and program agents. In this demonstration, we will show the *agent* and *builder*.

Agents

When MakeCode is enabled by the student or educator in Minecraft (by issuing the command 'code' in game), the agent appears. Designed to be cute and friendly to younger players, the agent can be programmed to protect the player, or find and return resources, or even to fight on behalf of the player.



The agent and a code that make the agent 'jump' up and forward in an arc

Builder

Perhaps of more interest to educators wishing to explore constructionist concepts is the *builder*. The builder is invisible but can be instructed to quickly build complex structures, such as castles with moats, battlements, floors and drawbridges (O'Sullivan et al., 2017).



A castle from the inside with the code that generated the floor and the door

References

O'Sullivan, M., Robb, N., Howell, S., Marshall, K., Goodman, L. (2017). Designing inclusive learning for twice exceptional students in Minecraft. *International Journal of E-Learning & Distance Education*, 32, (2).

Interpolating (and Extrapolating) 3D turtle Programs in Beetle Blocks

Ken Kahn, toontalk@gmail.com

Department of Education, University of Oxford, UK

Abstract

Turtle programs can be treated as objects to manipulate. In this demo a program takes two turtle programs as input and creates a new program that is the interpolation between the input programs. An input of .25, for example, will behave like one-fourth of the first program and three-fourths of the second. An input greater than 1 will extrapolate beyond the second program in the direction from the first program. This idea was explored in (Kahn 2007) for two-dimensional turtle programs. Here we generalise it for Beetle Blocks (Romagosa et al 2016), a 3D version of Snap! (Harvey & Mönig 2010).

Keywords

Program interpolation; Snap!; turtle programming; 3D turtles; Beetle Blocks; codification;

Interpolating 2D Logo turtle programs

(Kahn 2007) describes a program interpolator that can create a new program from two Logo programs that are defined using FORWARD, RIGHT, REPEAT, SETPENCOLOR, PENUP, and PENDOWN. As the simplest example consider two programs that draw different length lines:

<pre>to short forward 40 end</pre>	<pre>to long forward 100 end</pre>
------------------------------------	------------------------------------

The interpolated program is:

<pre>to short_to_long :x forward interpolate :x 40 100 end</pre>	<pre>to interpolate :x :a :b output :a + :x * (:b - :a) end</pre>
--	---

SHORT_TO_LONG 0 behaves just as SHORT, SHORT_TO_LONG 1 behaves as LONG, SHORT_TO_LONG .5 averages them, and SHORT_TO_LONG 2 extrapolates beyond LONG starting from SHORT.

The difficult step is canonicalising input programs that repeat a sequence of turtle commands a different number of times. (Kahn 2007) explains this in detail.

Moving to 3D

Beetle Blocks (Romagosa et al 2016) is a well-designed 3D version of Snap! (Harvey & Mönig 2010). Before one can begin to create interpolations of Snap! or Beetle Blocks programs we need a way to treat a block program as a data structure that can be manipulated programmatically. Fortunately, Snap! has a “codification” feature (Ball et al 2015; Harvey & Mönig 2018). Codification has been used to define how Snap! blocks can be translated to Python, C, Smalltalk, JavaScript, or other languages. We used this feature to translate Snap! blocks into JSON strings that are then converted into Snap! lists.

The problem of programming a Snap! program that constructs another program (the interpolation program) was resolved by defining the constructed program as a list of closures that can be run to

execute a sequence of commands. The values or expressions in corresponding commands in the two input programs are handled as variables closed over by functions.

FORWARD (or MOVE as it is called in Beetle Blocks) is treated in a similar manner to how the 2D turtle interpolator worked. RIGHT (or ROTATE as it is called in Beetle Blocks) has an additional argument that specifies whether the rotation is around the x, y, or z axes. As long as both programs rotate in the same dimension in the corresponding program locations it is straight-forward to generalise to 3D. To make a more generic interpolator other Beetle Block commands are also supported including “go to x: y: z:” and a list of “set” commands that change coordinates, hue, saturation, lightness, and opacity. Beetle Block command for extruding curves and lines are also handled. Furthermore, arithmetic expressions involving addition, subtraction, multiplication, and division are supported. Support for user reporters can be easily added.

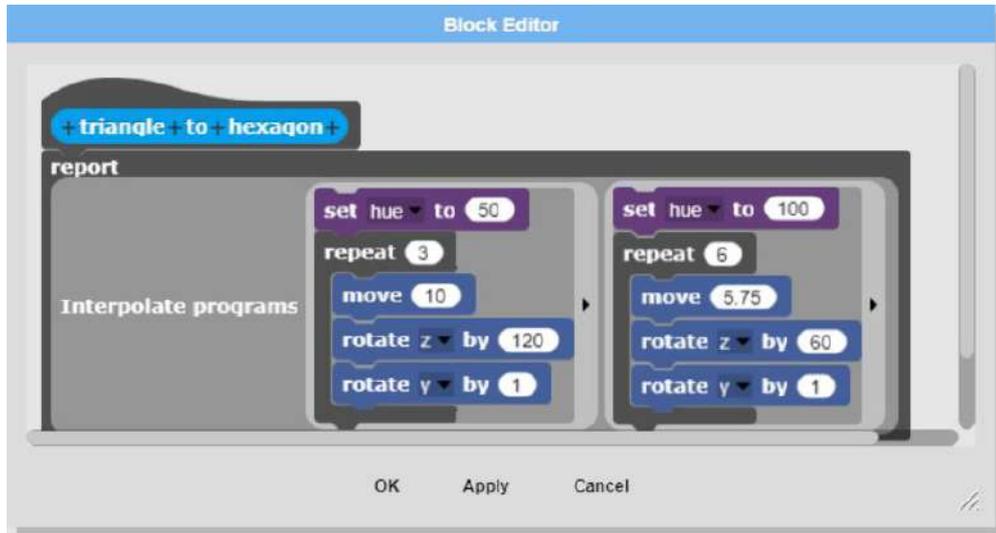


Figure 1. A simple example of interpolating a yellow triangle to green hexagon

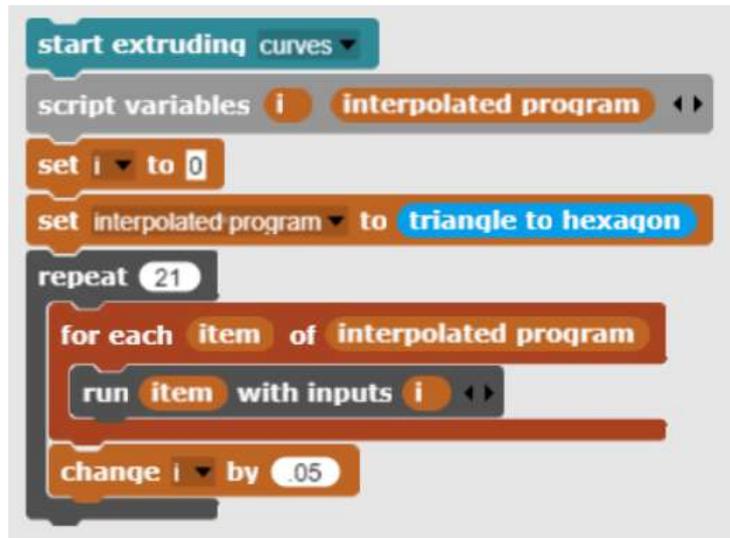


Figure 2. A program to run the triangle to hexagon interpolation program with 21 values from 0 to 1

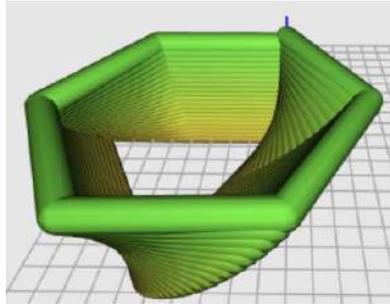


Figure 123. The result of running 21 interpolations between the triangle and hexagon

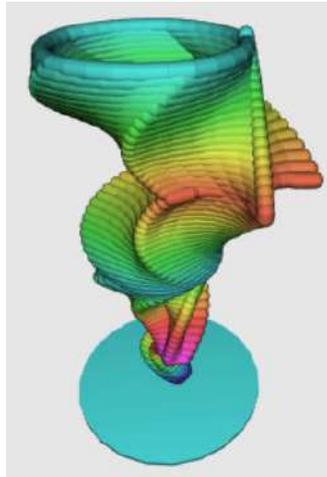


Figure 4. Interpolating and extrapolating between circles, stars, and pentagons

A very nice feature of Beetle Blocks is that the output of 3D turtle programs can be created on 3D printers. Figure 4, for example, is an attempt to make a vase. This program (and the interpolation program generator) can be found at <https://tinyurl.com/circle-star-pentagon>.

Conclusion

A turtle program is more than a shape. A circle, for example, can be drawn clockwise, counter-clockwise, or multiple times. The consequences of how a shape is drawn can result in dramatically different interpolations and extrapolations. Students tinkering with program interpolation and extrapolation are entering a mathematically and computationally rich area. And the creative and aesthetic possibilities of interpolating and extrapolating between turtle programs are many.

References

- Ball, M., Mock, L., McKinsey, J. Machardy, Z., Garcia, D., Titterton, N., Harvey, B. (2015) Oh, Snap! Enabling and Encouraging Success in CS1. In: SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education. pp 691-691.
- Harvey, B., Mönig, J., (2010) Bringing “No Ceiling” to Scratch: Can One Language Serve Kids and Computer Scientists? In Proceedings: Constructionism, Paris, France.
- Harvey, B., Mönig, J., (2018) Snap! 4.1. Reference Manual. <https://snap.berkeley.edu/SnapManual.pdf>.
- Kahn, K (2007) A Program to Interpolate (And Extrapolate) Between Turtle Programs. International Journal of Computers for Mathematical Learning. December. Volume 12. Issue 3. pp 255–262.
- Romagosa, B., Rosenbaum, E., Koschitz, D. (2016) From the Turtle to the Beetle - The Beetle Blocks programming environment. <http://hdl.handle.net/10609/52807>. Universitat Oberta de Catalunya.

Hedgehog: A Versatile Controller for Educational Robotics

Markus Klein, *klein@pria.at*
Practical Robotics Institute, Austria

Clemens Koza, *koza@pria.at*
Practical Robotics Institute, Austria

Wilfried Lepuschitz, *lepuschitz@pria.at*
Practical Robotics Institute Austria

Gottfried Koppensteiner, *koppensteiner@pria.at*
Practical Robotics Institute Austria

Abstract

We describe Hedgehog, an educational robotics controller designed to foster interest in STEM subjects. Hedgehog allows building robots out of common actuators and sensors, and can be combined with Lego for a beginner-friendly experience. Through building robots, students can learn and apply engineering, electronics, planning and teamwork skills. The controller facilitates learning programming at different age levels through both textual and visual programming support. For advanced students, Hedgehog's open source ecosystem allows delving into subjects such as microcontroller programming or cooperative robots as well. Hedgehog has been used in numerous workshops and also in robotics competitions with great success.

Keywords

Robotics; programming; visual programming; STEM

Hedgehog Controller

Hedgehog is a robotics controller: a computer aimed at controlling robotic components to allow building and programming robots. It is general-purpose, meaning that it was designed not only for a single task but to support whatever robotic project one might envision, and designed with educational use cases in mind. For example, Hedgehog controllers connect to a central WiFi, meaning that network congestion is less of a problem even in classrooms with multiple robots.

The Hedgehog Educational Robotics Controller was designed to offer learning capabilities for people of different ages and skill levels. As Hedgehog itself is only the controller, a lot of its features relate to software development, but also the hardware was designed with this in mind. The Raspberry Pi utilized by Hedgehog allows for experimentation, and connectors of the Pi and microcontroller not directly used for Hedgehog's core functionality are accessible as well. The case exposes the Pi's USB, HDMI, SD card slot, etc., and has holing compatible to Lego to allow children to build robots out of familiar components. Although mechanically compatible with Lego, Lego's motors and sensors are not supported due to their proprietary plugs. Instead, Hedgehog uses standard 0.1" pin headers and is compatible to easily available and relatively cheap RC-Servos.

Software Platform

Hedgehog supports out-of-the-box text-based programming with Python as well as a visual programming environment based on Google's open source library Blockly. Especially the latter

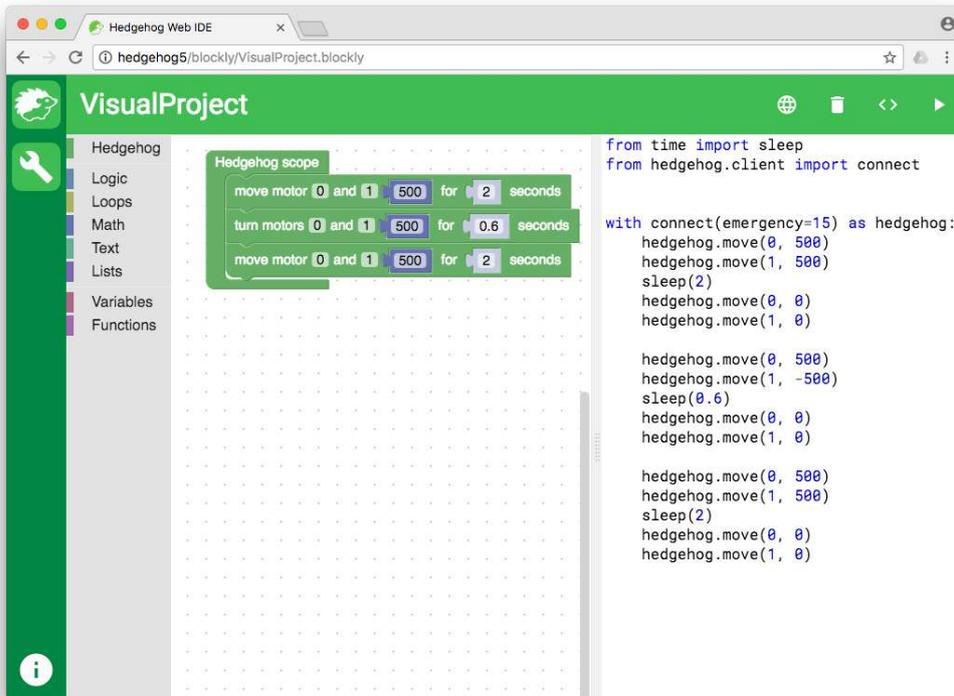


Figure 1. Screenshot of the visual programming environment with the visual program on the left and the generated Python code on the right.

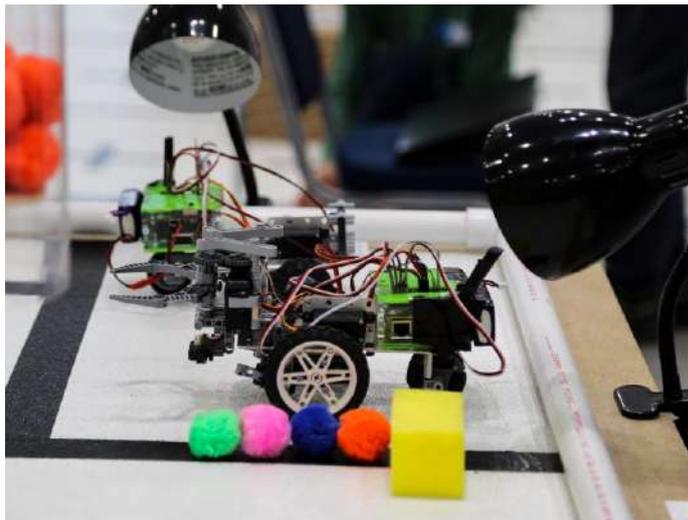


Figure 2. Hedgehog robotics controller in use at a robotics competition.

option allows beginners without any prior experience to dive into the world of robotics as it introduces an easier entry and less error potential in comparison with a text-based language. The design concept of the controller puts much emphasis on extensibility and versatility. Thus, adding new programming languages or integrating third-party libraries is well supported.

In order to allow children to quickly and easily get started with programming the Hedgehog controller, a web-based development environment called the Hedgehog IDE was implemented.

Programs are stored on the controller and users simply use their computers or tablets to connect to the controller via a browser and do not need to install any software on their machines which is a big advantage in workshop situations where time is always precious.

Real-World Usage

The Hedgehog controller has been used in a number of contexts, including workshops for middle school aged students, extracurricular activities with minimal supervision, and robotics tournaments. The controller was also used in the construction of an underwater robot, described in Grabler et al. (2018), which was in turn used in multiple educational robotics activities. Anecdotal evidence from these activities hints towards good acceptance of the Hedgehog system with regard to hardware compatibility (electronics connectors and mechanical connection points), usability of the development environment described in the previous section, and programming means - the libraries and the Python and Blockly languages.

For a subset of activities, in particular four workshops with 70 students aged 11 to 16, Koza et al. (2017) analyzed quantitatively and compared them with four earlier workshops using different controllers. All eight workshops were part of the same project and used a comparable style and setting. The reference workshops' controllers used the C programming language instead of Python.

The statements "working with robots was interesting/difficult/fun," and "overall, I would rank this course with ... stars," were looked at. Regarding difficulty, the results showed no significant difference. In the other categories (interesting, fun, overall stars), low ratings (1 or 2 stars) were almost absent from the Hedgehog workshops' results, while they did show up more frequently in the reference workshops. Although factors such as different instructors and students were not taken into account, it can be interpreted that students were more engaged in the workshops using Hedgehog.

Conclusion

Hedgehog shows a lot of promise for the education domain and has already been used successfully in different settings. We are confident that Hedgehog's ease of use and versatility makes it a good choice for various educational use cases. Through this, the Hedgehog controller helps reaching diverse audiences to foster their STEM interests.

Acknowledgements

The authors acknowledge the financial support from the European Union's Horizon 2020 research and innovation program under grant agreement no. 665972.

Demonstration

During this 30-minute demonstration, we will present the Hedgehog educational robotics controller, including its hardware capabilities and the software platform. The demonstration will also introduce means to facilitate constructionist learning by building and programming robots using the controller.

References

- Koza, C., Wolff, M., Frank, D., Lepuschitz, W. and Koppensteiner, G. (2017) Architectural Overview and Hedgehog in Use. In Proceedings: International Conference on Robotics and Education RiE 2017. p. 238-249.
- Reinhard Grabler, Markus Klein, Thomas Fellner, and Gottfried Koppensteiner. (2018) Development of a Low-Cost Maritime Educational Robotics Platform. *International Journal of Materials, Mechanics and Manufacturing*, Vol. 6, no. 3, pp. 208-214.

Working groups

WG1: Constructionist Approaches to Computational Thinking

Bernhard Standl, *bernhard.standl@ifs.tuwien.ac.at*

Gerald Futschek, *gerald.futschek@tuwien.ac.at*

Vienna University of Technology, Austria

Jane Waite, *jane.waite@computingschool.org.uk*

Queen Mary, University of London, UK

Andrew Paul Csizmadia, *A.P.Csizmadia@staff.newman.ac.uk*

Newman University Birmingham, UK

Lina Vinikienė, *lina.vinikiene@mii.vu.lt*

Vilnius University, Lithuania

Abstract

This working group elaborated an approach to evaluate computational thinking activities with learning in a constructionist way. As widely known, Wing stated in her refined definition of Computational Thinking (CT) (Wing 2008), that CT is an approach for solving problems that draws on concepts fundamental to computing. Later, Aho (2012) described the term CT as including algorithm-design and problem-solving techniques that can be used to solve common problems arising in computing. As Yadav et. al. (2011, 2016) reminded Wing's initial paper (2006) points out that CT involves three key elements Algorithms, Abstraction, and Automation. The term CT has been grown since then to a variety of interpretations. Ackerman (2001) compared Piaget's constructivism and Papert's development of this in a constructionist way and drew the two views together as learning in a constructionism way 'as Piaget and Papert do, that knowledge is actively constructed by a child in interaction with its world, then we are tempted to offer opportunities for kids to engage in hands-on explorations that fuel the constructive process.'. Papert's core message that the learner is 'projecting out our inner feelings and ideas is a key to learning. Expressing ideas makes them tangible and shareable which, in turn, informs, i.e., shapes and sharpens these ideas, and helps us communicate with others through our expressions.' This means, that new insights are the sum of single experiences made by applying existing knowledge for enhancing it. Considering both parts discussed above, constructionism and computational thinking, this working group's intentions are based on the combination of both for selecting and evaluating classroom activities. Therefore, we designed a matrix, where aspects from both, computational thinking and a constructionist learning approach, can be analysed. The matrix is designed to identify, categorize and evaluate such classroom activities and encompasses three parts: **Computer Science Concepts, Problem-Solving Concepts, Levels of Abstraction.**

		Computer Science Concepts					Type of resource	Paid or free	popularity, used prior studies, enforced by educational stakeholders, other)	Activity/Approach to Map	Computational Thinking Skills				
Coder	Date	ALP	DSR	CPH	C&N	ISS					ABS	ALT	DEC	EVA	GEN
		9	1	0	2	3					4	8	5	5	0
Andrew (sample)		1					Game	Paid		Activity: Cody Roby Description: Cody Roby is an unplugged code game which either allow two teams to play an unplugged coding game or a player to design their own game for others to play. URL: http://codeweek.it/codyrobby/	1	1	1	1	
Jane	May 2018	1?					Lesson plan	Free		Activity: Barefoot crazy characters - unplugged activity teaching about algorithms.		1	1	1	

This working group’s paper presents our approach of a systematic evaluation of classroom activities in terms of constructionist learning and discusses first results of our evaluation process, where we coded such activities corresponding to the matrix.

Keywords

computational thinking; constructionist learning; classroom activities

1. Introduction

This working group elaborated an approach to evaluate computational thinking activities with learning in a constructionist way. As widely known, Wing stated in her refined definition of Computational Thinking (CT) (Wing, 2008), that CT is an approach for solving problems that draws on concepts fundamental to computing. Later, Aho (Aho, 2012) described the term CT as including algorithm-design and problem-solving techniques that can be used to solve common problems arising in computing. As Yadav et. al. (Yadav, Gretter, Hambrusch, & Sands, 2017; Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011) reminded Wing’s initial paper (Wing, 2006) points out that CT involves three key elements Algorithms, Abstraction, and Automation. The term CT has been grown since then to a variety of interpretations. Ackerman (Ackermann, 2001) compared Piaget’s constructivism (Piaget & Duckworth, 1970) and Papert’s development of this in a constructionist way and drew the two views together as learning in a constructionism way ‘as Piaget and Papert do, that knowledge is actively constructed by a child in interaction with its world, then we are tempted to offer opportunities for kids to engage in hands-on explorations that fuel the constructive process.’. Papert’s core message that the learner is ‘projecting out our inner feelings and ideas is a key to learning. Expressing ideas makes them tangible and shareable which, in turn, informs, i.e., shapes and sharpens these ideas, and helps us communicate with others through our expressions.’ This means, that new insights are the sum of single experiences made by applying existing knowledge for enhancing it. Considering both parts discussed above, constructionism and computational thinking, this working group’s intentions are based on the combination of both for selecting and evaluating classroom activities. Therefore, we designed a matrix, where aspects from both, computational thinking and a constructionist learning approach, can be analysed. The matrix is designed to identify, categorize and evaluate such classroom activities and encompasses three parts: Computer Science Concepts, Problem-Solving Concepts, Levels of Abstraction. This working group’s paper presents our approach of a systematic evaluation of classroom activities in terms of constructionist learning and discusses first results of our evaluation process, where we coded such activities corresponding to the matrix.

In our approach, first, we tried to identify a suitable research question to our paper's topic. One of the earlier ones was: How could be CT learned in constructionist way in different academic disciplines / different tasks? Following further discussion, the group agreed on investigating existing classroom activities as a starting point to start to formulate and evaluate a toolset which might prove useful in investigating constructionism and computational thinking in K-12 classrooms.

Leading by the research questions: *In what way do K-12 classroom activities teach computational thinking in a constructionist way? How useful is our model in evaluating CT in K-12 classroom activities? How useful is our model in evaluating constructionism in K-12 classroom activities?* we continued with a discussion about our ideas and how we could create a system for a classification of Constructionist-Computational-thinking Classroom Activities.

The idea was agreed to design a mapping tool, which we could use to record and code classroom activities to start to investigate and compare activities for their computational thinking and constructionist attributes.

This working group paper is structured as follows. First, we are describing the background of our work in literature and further describe an overview on computational thinking, constructionism and its concepts respectively combination of both parts. In a further section, we will describe the methods we used to gather and evaluate the classroom activities. This includes a detailed description of the mapping tool we have developed. This is followed by a presentation of the results and a final discussion of this working group paper.

2. Background

Computational Thinking (CT) has been widely discussed since Jeannette Wing published her article "Computational Thinking" in 2006 (Wing, 2006). There have been several attempts to define this concept more precisely since then with the discussion converging to a handful of skills which characterize the thinking concepts associated with CT. These concepts include abstraction, decomposition, algorithmic thinking, generalization and evaluation. Algorithmic thinking education has a long tradition in constructionist education. Logo, Scratch and other programming tools invite creative learning of programming and algorithmic thinking. But CT is considered very broad, it covers not only programming and algorithmic skills, but also activities like problem formulation, system modelling and solution evaluation.

An international group collaborated on this working group paper to consider computational thinking and constructionism. In identifying an underpinning theory to our working group's topic, we first investigate the two parts of the title of our paper: Constructionism and Computational Thinking. Despite both topics have their own deep background, of which a description would go far beyond this paper's scope and word length, the next two subsections will give a short overview and in particular connections to our work.

2.1 Constructionism

Ackerman (Ackermann, 2001) in her comparison of Piaget's constructivism and Papert's development of constructivism in a constructionist way, drew the two views together describing learning in a constructionism way as *'that knowledge is actively constructed by the child in interaction with her world, then we are tempted to offer opportunities for kids to engage in hands-on explorations that fuel the constructive process.'* She further asserted Papert's core message that the learner *'projecting out our inner feelings and ideas. is a key to learning. Expressing ideas makes them tangible and shareable which, in turn, informs, i.e., shapes and sharpens these ideas, and helps us communicate with others through our expressions.'* Similar to Piaget, Papert identifies learning by constructing and reconstructing knowledge through experience. In particular, Papert's constructionism sets a focus from learning in situations *'rather than looking at them from a distance, that connectedness rather than separation are powerful means of gaining understanding'* (page 8). This means, that new insights are the sum of single experiences made by applying existing knowledge for enhancing it. Therefore, *'hands-on activities are the best for the classroom applications of constructivism, critical thinking and learning.'* (http://www.teach-nology.com/currenttrends/constructivism/classroom_applications/)

We summarise the significant feature of constructionism for our study as being that learners make something in order to learn. However, what that something is, and what degree of autonomy is associated with the process of making is a focus of our study along with the relationship of constructionism and computational thinking.

2.2 Computational Thinking

Computational thinking has become a popular term in computer science education with, definitions varying depending on perspective (Tedre & Denning, 2016). Previous works present at least three types of approaches to defining CT: it is a set of skills to help solve problems e.g. (Wing, 2006), it is a thought process e.g. (Aho, 2012), or it is a problem-solving process e.g., (Voogt, Fisser, Good, Mishra, & Yadav, 2015). As widely known, Wing (Wing, 2008) stated in her refined definition of CT that it is an approach for solving problems that draws on concepts fundamental to computing. Later, Aho (2012, p. 832) described the term CT as including “algorithm-design and problem-solving techniques that can be used to solve common problems arising in computing”. As Yadav (Yadav et al., 2017, 2011) reminded Wing's initial paper (Wing, 2006) points out that CT involves three key elements Algorithms, Abstraction, and Automation. The term CT has been grown since then to a variety of interpretations. Settle and Perkovic (Perković, Settle, Hwang, & Jones, 2010), who proposed seven principles for CT across the curriculum, added that CT also involves computation, communication, coordination, recollection, evaluation, and design. For Lee (Lee et al., 2011) CT involves defining, understanding, and abstraction. Barr et al. (2011) suggested that CT involves the design of solutions, implementation of designs, testing, running analysing, reflecting, abstraction, creativity, and group problem solving. Grover et al. (Grover2013; Grover & Pea2018) stated that CT should include among others abstraction, information processing, structured problem-solving decomposition as modularization, iterative recursive thinking, and efficiency. Again, for Lee et al. (Lee, 2011) CT involves defining, understanding, and solving problems, reasoning at multiple levels of abstraction, understanding and applying automation, and analysing the appropriateness of the abstractions made.

According Brennan & Resnick (2012), CT involves three dimensions such as computational concepts (the concepts designers employ as they program), computational practices (the processes of construction), and computational perspectives (the perspectives designers form about the world around them and about themselves).

2.3 Constructionism in Computational Thinking

Considering both parts discussed above, constructionism and computational thinking, this working group's intentions are based on the combination of both for selecting and evaluating classroom activities. Therefore, we designed a matrix, where aspects from both, computational thinking and a constructionist learning approach, can be analysed. The matrix is designed to identify, categorize and evaluate such classroom activities and encompasses three parts:

1. Computer Science Concepts
2. Problem-Solving Concepts
3. Levels of Abstraction matrix

Below we are discussing these three dimensions in detail:

a) Computer Science Concepts

From the viewpoint of computer science education, teaching and learning of computer science (CS) concepts is more important than learning how to use computer systems. Dagienė, Sentance and Stupurienė (2017) categorized in their paper “Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics” the CS concepts in following 5 categories:

- ALP: Algorithms and Programming
- DDS: Data, Data Structures and Representation
- CPH: Computer, Processes and Hardware
- C&N: Communications and Networks
- ISS: Interactions, systems and society

These categories are so designed that each concept fits in one of the categories. Since CS tasks often involve more than one concept, a task may be assigned to more than one category.

b) Computational Thinking Concepts (CT)

Computational Thinking Skills are independent from CS concepts. Computational thinking originates from solving CS tasks but the essence of these thinking skills can be applied also in all other disciplines. Based on the work of Selby and Woollard (2013), Dagiene et al. (2017) categorized also the Computational Thinking Skills into 5 categories:

- ABS: Abstraction
- ALT: Algorithmic Thinking
- DEC: Decomposition
- EVA: Evaluation
- GEN: Generalisation

Simple definitions of each of these concepts are (Selby 2013, Standl 2017)

- Abstraction: Ignoring unnecessary detail. When abstracting a problem in a way that helps to solve it. If we had to keep all the details in our heads, we could never get anything done. As we have described above, abstraction is mentioned as a key-stone of CT and Grover and Pea (2018) identified it as a core element, which differs CT from other types of thinking.
- Algorithmic thinking: Considering the sequence of steps. This includes a design of an algorithm to develop the step-by-step instructions for solving the problem. Starting from what already is known and working outward from there by making a plan how to approach to solve the problem.
- Decomposition: breaking a problem down into parts. Decomposition involves finding structure in the problem and determining how the various components will fit together in the final solution. Doing decomposition well makes it easier to modify the solution later by changing individual components, and also enables the reuse of components in solutions to other problems.
- Evaluation: comparing alternatives and how does the solution work in practice and are there alternatives? Trying to give the problem different inputs to look how the solution works.
- Generalisation: Creating things that can be reused in more than one scenario. Is the solution to similar problems also applicable and what is needed to do so? How can the solution be generalized? Which parts turned at the evaluation out to be not necessary?

c) Levels of abstraction (LOA) matrix

The degree of autonomy of learners to make choices about their constructed artefacts was viewed as an important aspect to capture in order to evaluate the opportunity for learners to engage in constructionist learning (reference needed) in computational thinking activities. Three frameworks have been combined and further developed to create a levels of abstraction matrix which provide an opportunity to evaluate autonomy for the different 'levels' or 'stages' for a programming project. The work that we build upon is the Levels Of Abstraction (LOA) framework (Perrenet et al. 2005; Perrenet & Kaasenbrood 2006), the Abstraction Transition (AT) Taxonomy (Cutts et al. 2012) and the Use - Modify- Create approach (Lee et al. 2011).

A levels of abstraction (LOA) model (Perrenet et al. 2005; Perrenet & Kaasenbrood 2006) has been suggested to support novice university students in thinking about algorithms in programming of: problem, object, program and execution. This model has been situated for high school learners by renaming the object level as algorithm (Armoni 2013) and with younger K-5 learners by renaming levels as: task for problem; design for object; code for program and running the code for execution (Waite et al 2016, Waite et al 2017).

Cutt's et al. (2012) investigated novice undergraduate 'talk' of programming activities in response to peer instruction questions and proposed the three leveled Abstraction Transition Taxonomy of English, CS speak and Code, where English is used to define the goal, 'CS Speak' for the technical description and code to accomplishes the goal.

The Use, Modify, Create approach has been suggested as a learning framework, where learners first use products created by other which are not ‘theirs’, move on to modify products and finally create their ‘own’ products (Lee et al. 2011).

3. Method

3.1 The Mapping Tool

A mapping tool was required to provide a means to classify and evaluate CT and constructionism. A simple google doc spreadsheet was used to facilitate sharing of the mapping tool. Following email discussion of a range of opportunities for classifying computational thinking a modification of the 2-dimensional model of computational thinking skills and computer science concepts for individual tasks (Dagiene et al 2017) was agreed to be used to evaluate the intended teaching and learning objectives related to computer science concepts and computational thinking(CT).

3.2 Computer Science (CS) Concepts

The working group adopted the computer science concept categorization of school computing as proposed by Dagiene et al (2017) as that of:

1. Algorithms and Programming
2. Data, Data Structures and Representation
3. Computer Processes and Hardware
4. Communications and Networking
5. Interactions, Systems and Society

In addition, the working group adhered to the guidance from Dagiene and el (2017) that for practical purposes, a precise definition of each category is required and this can be achieved by the usage of keywords as shown in [Table 1](#).

Table 1 Computer Science (CS) Concept Categories and Keywords (Adapted from Dagiene et al (2017))

CS Concept Categories	Code	Keywords
Algorithms and Programming	APL	Algorithm; Binary search; Boolean algebra; Breadth-first search; Brute-force search; Bubble sort; Code; Coding; Computational complexity; Constants; Constraints; Debugging; Depth-first search; Dijkstra’s algorithm; Dynamic programming; Divide and conquer; Encapsulation; Function; Greedy algorithm; Heuristic; IF conditions; Inheritance; Iteration; Kruskal’s algorithm; Logic gates; Loops; Maximum flow problem; Objects; Operations AND, OR, NOT; Optimization; Parameters; Prim’s algorithm; Procedure; Program; Programming Language; Program execution; Quick sort; Recursion; RSA algorithm; Shortest path; Selection; Sequence; Sorting; Steps; Traveling salesman problem; Variables Steps, sequence, algorithm, code, program
Data, Data Structures and Representation	DDS	Array; Attributes; Biconnected graph; Binary and hexadecimal representations; Binary tree; Character encoding; Databases; Data; Data mining; Eulerian path; Finite-state machine; Flowcharts; Fractals; Graph; Hash table; Integer; Information; Linked list; List; Queue; Record; Stack; String

Computer Processes and Hardware	CPH	Cloud computing; Deadlock; Fetch-execute cycle; Grid computing; Image processing; Interpreter; Memory; Multithreading; Operating system; Parallel processing; Peripherals; Priorities; RAID array; Registers; Scheduling; Sound processing; Translator; Turing machine
Communications and Networking	C&N	Client/server; Computer network; Cryptography; Cryptology; E-commerce; Encryption; Parity; Protocols; Security; Topologies
Interactions, Systems and Society	ISS	Classification; Computer use; Design; Ethics; Graphical User Interface; Human Computer Interaction (HCI); Legal issues; Robotics; Social issues;

Apart from assisting in the categorization of tasks, keywords are helpful to teachers who wish to find tasks that assist in introducing, teaching or formative assessing a specific computing topic (Dagiene and Sentance 2016; Yang and Park 2014).

3.3 Computational Thinking (CT): Skills Level

The working group adopted as its categorization of computational thinking which followed the work of Selby and Woolard (2013) and has been adopted by Computing At School in the UK in developing guidance for teachers on computational thinking (Csizmadia et al 2015). This approach was adopted by both Giordano et al (2015) and Dagiene et al (2017) in design a framework for classifying computational thinking skills and computing concepts. The following table summarizes the five categories of computational thinking: Abstraction, Algorithmic Thinking, Decomposition, Evaluation and Generalization, and describes aspects of computational thinking as exhibited by learners as they work their way through a task:

Table 2 CT Skills and ways to identify them (adapted from Dagiene et al.(2017))

CT kills	Code	How to identify the use of this skill
Abstraction	ABS	Removing any unnecessary detail Identifying key elements in the problem Choosing an appropriate representation of a system
Algorithmic Thinking	ALT	Thinking in terms of sequences Thinking in terms of rules Creating an algorithm Executing an algorithm
Decomposition	DEC	Breaking down tasks into sub-tasks Thinking about problems in terms of their component parts Making decisions about dividing into sub-tasks with integration in mind

Evaluation	EVA	Find an appropriate solution Finding the best solution Deciding whether the solution is fit for purpose Deciding whether the solution is the most efficient one
Generalization	GEN	Identify patterns, similarities and connections Solving new problems based on solutions to similar problems Utilizing the general solution, i.e. induction

Classifiers need to be able to know how to identify how a particular skill might be used to solve a given task (Table 2). One of the difficulties encountered by the members of the working group was that they could only presume to know how an individual learner solves a specific task which may differ from the way that either the task setter or the classifier would solve the task. In practical terms, there may be more than one computational thinking skill associated with each task. Therefore, the working group followed the guidance suggested by Dagiene et al (2017) of recording a maximum of three computational thinking skills.

3.4 Developing the Constructionism Matrix

A similar discussion was undertaken to find a framework to evaluate classroom activities for their ‘degree’ of constructionism features. A new framework was devised, which will be evaluated as part of the working group activity.

Waite et al. (2016) suggested that the LOA model might be mapped to the AT taxonomy, with Problem being matched to English, and object to CS speak and Program and Execution to Code. Similarly, we have combined the Program and Execution level, as our focus is on the autonomy of learners at each level, and they can have no influence on how the code runs to suggest a modified LOA suitable for K-12 of:

- Problem or task (English)
- Object, algorithm or design (CS Speak)
- Program or code and Execution or running the code (Code)

This provides the first dimension of our constructionism matrix to which we added the dimension of Use, Modify, Create (UMC) (Lee et al. 2011) to provide an indication of learner ownership and autonomy for each level. Descriptions were added for each intersection of modified LOA and UMC matrix. An iterative process was undertaken to develop these descriptions. As classroom activities were coded, the scale was reviewed and adapted to enable differences between the activities to be highlighted and then further subdivided to a 1 to 5 scale to describe a gradation of learner autonomy in the construction process. Our final constructionism matrix is shown in Table 3.

Table 3: Constructionism Matrix

Adapted LOA	Problem or Task (English)	Design/Object/Algorithm (CS Speak)	Code/Program/Running the code (Code)	Use/Modify/ Create
Scale 1	No independence e.g. the teacher or activity states the problem, a copy task activity	No independence using a pre-existing design, or no design	No independence, copying pre-defined code	Use

2	Limited independence, can only change superficial aspects of the problem e.g. adding extra questions to a quiz, changing the characters of an animation (broadly a minimal remix)	Limited independence, can only change some aspects of the design, objects, algorithm, for example reordering sequence of events, changing the data used	Limited independence, can only change some elements of the modelled or pre-defined or example code	Modify
3	Moderate independence, can change a broader range of characteristics of the problem but limited to the genre and context of the	Moderate independence, can change a broader range of characteristics of the design but changes are within the genre and context of the original exemplified design.	Moderate independence, can change a broader range of characteristics of the code but limited to the genre and context of the task and also to the hardware and software defined by the task or teacher.	Modify
4	Increasing independence, cannot change the 'Genre' but has full control of the context e.g. must be a quiz, or a physical computing problem but can be any context.	Increasing independence, cannot change the design approach to be used, but can adapt the objects, algorithm etc to meet needs of the problem	Increasing independence, limited by type of hardware and software to be used, but could choose a different input type or type of microcontroller or different block based programming language.	Modify
5	Full pupil independence the pupil has full control over the problem to be considered e.g. can be a quiz, game, physical product, unplugged etc	Full pupil independence the pupil has full control over the design to be considered can choose format and approach to be taken	Full pupil independence the pupil has full control over the hardware, software and implementation choices	Create
6	Levels 1 and 2 seen in a unit of work	Levels 1 and 2 seen in a unit of work	Levels 1 and 2 seen in a unit of work	Use/Modify
7	Levels 1, 2 and 3 seen in a unit of work	Levels 1, 2 and 3 seen in a unit of work	Levels 1, 2 and 3 seen in a unit of work	Use/Modify
9	Not Applicable	Not Applicable	Not Applicable	

As the mapping tool was used further attributes were added, these included:

- Type of artefact made (Physical computing, onscreen, unplugged, concept)

- Type of resource (Lesson plan, game etc)
- Cost (Paid for, free)
- Activity/Approach URL and description
- Target age range (see [Table 4](#))

Table 4: Target Age Range

Years old	US grades	English year groups
5-6	K-1	Key Stage 1 Years 1-2
7-11	2-5	Key Stage 2 Years 3-6
12-14	6-8	Key Stage 3 Year 7-9
15-16	9-10	Key Stage 4 Years 10-11
17-18	11-12	Key Stage 5 Years 12-13

Finally, each activity was double blind coded by two members of the group.

3.5 Selection of activities.

Where information was available related to the popularity of classroom activities this data was used to select these activities. In the UK, a review of computer science education (Royal Society, 2017) was used to select the top 5 resource providers and a selection of their materials was included. To select material provided by one supplier of resources, we contacted them and asked which were the most popular and used these. For some providers material is only available with payment, here the materials were not included as there would be no means to then share and compare the approach taken with readers of the research. Our study is not rigorous in sampling, but our intention was to trial the approach for review of materials to suggest next steps for a more complete review.

For Lithuania, a number of anecdotally popular resources were selected, one from Italy and the rest from the US. One activity (CodeMonkey) was chosen because this game-based environment was awarded as the Best Coding and Computational Thinking Solution⁷⁷ in 2018. This classroom activity is recommended in informatics educational content in primary education in Lithuania as a tool that engages student to learn informatics concepts by practice not only during informatics lessons, but also during other subjects such as math, etc. (firstly, students solve problem without computer (eg. measure the distance) and then repeat the same task using CodeMonkey tool). Other activities are applied or recommended as classroom activities in primary or middle school in Lithuania as tools to teach computer science concepts or programming. Several activities were the most popular during the Hour of code⁷⁸ (Code Event) in 2015. In addition, some resources (Scratch, Khan Academy, Code.org) are mentioned as the best free resources for teaching youngsters to code.⁷⁹

⁷⁷<https://www.playcodemonkey.com/blog/2018/06/20/codemonkey-awarded-best-coding-computational-thinking-solution/>

⁷⁸ <https://blogs.sas.com/content/sascp/2015/12/07/our-favorite-hour-of-code-resources-for-csedweek15/>

⁷⁹ <https://codakid.com/top-5-free-kids-coding-websites-of-all-time/>



Figure 1. Students solve problem without computers and then solve it with computers, discusses about coding with CodeMonkey (Taurage „Saltinis“ Progymnasium, Lithuania)

3.6 Inter-reliability of codings & quantitative analysis

As a resource was added by a researcher to our main spreadsheet, a second sheet had been set up that automatically inherited the link to the resource. No other data was copied across. A second coder could then pick up the activity and code it without having seen the prior codings of their peer. Therefore we had two authors independently coding each resource. The results were copied into SPSS for interreliability evaluation using Cohen's kappa (Cohen, 2011) where each activity was a 'case'. In using SPSS there is a known limitation, that if one coder only uses one value for a variable, then SPSS evaluates this as a constant and will not report a reliability statistic. An documented acceptable workaround is to add a distinct pseudo case with a very small weighting (0.0001) as opposed to all other cases which are allocated a weighting of 1 (UCLA, 2018). This technique was required to be used as we were using dichotomous variables (yes/no), such as for the computer science concepts which had not coded for any activity.

As we were working as a distant group SPSS syntax was used and a shared excel spreadsheet so that authors could validate the quantitative data analysis techniques used. The syntax and input data is available on request from the first author.

Our results are predominantly non-parametric data; therefore, we have used the Mann-Whitney U test to compare two independent groups such as concepts being selected as taught or not (Cohen, 2011). To grade effect sizes, we used Cohen's classification: if r is 0.1 to 0.3 there is small effect, if r is 0.3 to 0.5 there is moderate effect and 0.5 and above is large effect. For variables with more than 2 possible values, such as the activity type or the artefact type created, the independent Kruskal-Wallis test was used to investigate any significant differences (Cohen, 2011). We report the χ^2 statistic for this statistic and include post-hoc analysis through the Bonferroni-correction for pairwise comparison of categories. For paired dependent testing, such as comparing trends of coders ratings of the constructionism matrix across the problem, design and code dimensions we have used the Wilcoxon Signed-Rank test (Cohen, 2011). For this statistic we report the number of cases tied and the increases or decreases of ranked responses to report on the underlying data.

4. Results

We first report on descriptive statistics for the coding of activities, followed by intercoder-reliability reporting and finally with inferential statistics as we look for relationships between the variables coded. Twenty-one activities were coded as shown in Appendix 1. 57% ($n=12$) of the activities were for 7 to 11 year olds, 23% ($n=5$) for 5 to 6 year olds, 2 were for any age from 5 years, 1 was for any age between 5 and 11 and 1 activity for students from 10 years onwards. Therefore, all activities were judged to be suitable for some set of pupils in the K-5 (primary age range in England).

Over 75% of the activities selected were lesson plans ($n=16$), the remainder were online activities ($n=4$) except for one board game. Two thirds of the activities originated from England ($n=14$), 29% from the US ($n=6$) and one from Italy. Nearly 30% of the activities employed Scratch ($n=6$), 15% ScratchJr ($n=3$), there were 2 route based programming languages where the student moved an onscreen character or programmable toy with direction keys and there was one activity for each of the programming languages

of blockly, CoffeeScript, Python, Java and a physical computing software called Crumble. There was also one activity that used Google Forms and 6 activities which used no programming language or specific software to make something.

In order to give a % for each concept type we have taken the average number of coded activities for each concept across the two coders and present this as a percentage out of the 21 activities as shown in [Table 5](#). On average across the coders, the most popular CS concept taught in the sample of activities codes was ALP (Algorithms and Programming) with 90% of the activities coded to this concept. No activities were coded to Data, Data Structure and Representation concept. For CT concepts, the most popular coded concept was ALT (Algorithms) with, 83% of the activities coded as teaching this concept, the second most popular CT is DEC (decomposition) with 45% of activities coded for this. The most frequent artefact type coded was concept at 79%, followed by 64% for OnScreen artefact types.

Table 5. CS concept, CT concept and artefact type counts by coder and % out of 21 activities

	CS concepts					CT concepts					Artefact type			
	ALP	DDS	CPH	C&N	ISS	ABS	ALT	DEC	EVA	GEN	Physical	OnScreen	UnPIugged	Concept
Coder 1	19	2	0	2	3	4	19	9	9	1	2	13	10	16
Coder 2	19	1	0	2	4	5	16	10	7	3	2	14	7	17
Average	19	1.5	0	2	3.5	4.5	17.5	9.5	8	2	2	13.5	8.5	16.5
%	90%	7%	0%	10%	17%	21%	83%	45%	10%	10%	10%	64%	40%	79%

Cohen’s κ was run to determine if there was agreement between the authors coding of n activities as shown in [Table 6](#). There was a substantial agreement ($\kappa = .778$, $p < 0.005$) across all the groups of variables.

Table 6 Cohen’s κ for coding of activities.

Group of variables	Variable	Cohen’s κ	p	Notes
Computer science (CS) concepts	ALP: Algorithm and Programming	1	$p < 0.000$	All coder agreed on every case
	DDS: Data, Data Structures and Representation	.644	$p = 0.002$	
	CPH: Computer, Processes and Hardware	not reported		No cases were recorded for this
	C&N: Communications and Networks	1	$p < 0.000$	All coders agreed on every case
	ISS: Interactions, systems and society	.829	$p < 0.000$	

	Average κ for all CS concepts	.868		Almost perfect agreement
CT concepts	ABS: Abstraction	.696	p=0.001	
	ALT: Algorithmic Thinking	.877	p<0.000	
	DEC: Decomposition	.704	p=0.001	
	EVA: Evaluation	.690	p=0.001	
	GEN: Generalisation	not reported		17 agreements 2 disagreements but 1 coder only recorded one value
	Average κ for all CT concepts	.74175		Substantial agreement
Constructionism scale	Problem or Task	.535	p<0.000	
	Design/Object/Algorithm	.563	p<0.000	
	Code/Running the code	.432	p=0.001	
	Average κ for all Constructionism Scale	.51		Moderate agreement
Artefact type	Physical Artefact Created	1	p<0.000	
	On-screen Artefact Created	0.901	p<0.000	
	Unplugged Artefact Created	0.807	p<0.000	
	Concept Artefact Created	0.604	p=0.002	
	Average κ for all Activity Type	0.82		Almost perfect agreement
Overall Average κ		.7784		Substantial agreement

As shown in [Table 7](#), the majority of activities were coded at level 1 scale for Problem or Task level at 52% and 61% for coders, followed by 24% to 29% at level 2 and 9% to 10% for level 3 with only 1 activity assigned a 4 or 5 level. The majority of designs were at scale 2 at 53% to 71%, with 14% to 24% at level 1 and only 5% at level 3. As with the design level, the majority of activities were rated at a level 2 scale for the code dimension at 57% and 61% with between 5% and 9% at level 1, and one activity assigned a level 5. For the coding dimension a quarter, 24% to 29% were not assigned a level (for example as they were unplugged activities).

Table 7. Percentage and count (n) of coded activities by constructionism scale.

Adapted LOA Scale	Problem or Task (English)		Design/Object/Algorithm (CS Speak)		Code/Program/Running the code (Code)	
	Coder 1	Coder 2	Coder 1	Coder 2	Coder 1	Coder 2
1	61% (13)	52% (11)	24%(5)	14% (3)	9%(2)	5% (1)
2	5% (1)	0	28%(6)	33% (7)	52%(11)	48%(10)
1 to 2	19% (4)	29% (6)	24% (5)	38%(8)	5% (1)	13%(3)
2 or 1 to 2	24% (5)	29% (6)	53% (11)	71%(15)	57% (12)	61%(13)
3	5% (1)	0	5% (1)	5% (1)	5% (1)	0
1 to 3	5% (1)	9%(2)	0	5 (22%)	0	5% (1)
3 or 1 to 3	10% (2)	9%(2)	5% (1)	5%(1)	5%(1)	5% (1)
4	0	0	0	0	0	0
5	0	5% (1)JS	0	0	0	5% (1)JS
4 or 5	0	5% (1)	0	0	0	5%(1)
coded as not applicable	5% (1)	5% (1)	19% (4)	10%(2)	29%(6)	24%(5)
n	21	21	21	21	21	21

Our data is predominantly nonparametric. We have dichotomous (yes/no) variables for the CS concept and CT concepts and ordinal variables for the constructionism scales for each of the dimensions of problem, design and code.

Summary Dimensions: To simplify reporting of the constructionism scale we have combined the ‘1 to 2’ response with the 2 response and the ‘1 to 3 response’ with the 3 response for each of the dimensions. We have also removed the 4 and 5 scale responses as there was only 1 activity coded at the 5 scale by one coder, and the second coder placed this same activity as 3, so we have classified this as an outlier and will further consider this case in discussion. In doing this we have created Summary Dimensions with scales of 1, 2 and 3.

We have used the Mann-Whitney U statistic to investigate whether there was any statistically significant relationships between the constructionism scale and its CS concept or CT concept. Here the cases, rather than being an activity, were the codings of each activity by each researcher therefore the maximum population was 42 (n=42) for these tests.

The null hypothesis was that for each CT and CS concept there was no statistically significant difference between whether an activity was coded with that concept and the constructionism scale.

The null hypothesis for all CS and CT concepts could not be rejected as all tests showed no statistically significant difference. The test statistics are shown in Appendix 2. A cross tabulation of the data is shown in [Table 8](#).

Table 8 Cross Tabulation of CS & CT concepts by constructionism matrix dimensions for both coders showing % of type and (n) using Summary Dimensions, with most popular per concept & dimension highlighted.

		Problem			Design			Code		
		1	2	3	1	2	3	1	2	3
CS con cept s	ALP	55%(18)	33%(11)	12%(4)	23%(8)	71%(24)	6%(2)	23%(8)	71%(24)	6%(34)
	DDS	100%(2)	0	0	33%(1)	67%(2)	0	0	0	0
	CPH:	0	0	0	0	0	0	0	0	0
	C&N:	100%(4)	0	0	0	0	0	0	0	0
	ISS:	57%(4)	43%(3)	0	29%(2)	71%(5)	0	0	100%(5)	0
CT con cept s	ABS:	86%(6)	0	14%(1)	0	67%(2)	33%(1)	0	0	100%(1)
	ALT:	61%(17)	25%(7)	14%(4)	24%(7)	69%(20)	7%(2)	12%(3)	80%(20)	8%(2)
	DEC:	43%(6)	43%(6)	14%(2)	25%(4)	63%(10)	12%(2)	8%(1)	75%(10)	17%(2)
	EVA:	54%(7)	15%(2)	31%(4)	14%(2)	72%(10)	14%(2)	0	82%(9)	18%(2)
	GEN:	100%(2)	0	0	50%(1)	50%(1)	0	50%(1)	50%(1)	0

Similarly, we performed the same statistic on the artefact types reported by the coders, as there was only one board game we removed this from the test and used a Mann-Whitney U to compare those activities which were lesson plans and those which were online student activities for each of the dimensions of the constructionism matrix, namely problem, design and code, for the assigned constructionism scale of pupil autonomy. The null hypothesis for these tests was that there would be no statistically significant differences in the gradings of the constructionism scale for lesson plan based resources or online student resources. There were two statistically significant differences, as shown in Appendix 2, for the Design level ($p=0.018$, $n=25$, $U=54.5$, $r=3.98$) and the Code level ($p=.048$, $n=29$, $U=37$, $r=.398$) both of medium effect size, meaning we do not accept the null hypothesis. A cross tabulation of the data is shown in [Table 9](#) and shows that the scale of autonomy was recorded at a higher level for lesson plans than for online resources. This may be significant in that it is may not the CS or CT concept that is having a bearing on the constructionism aspect of a task, more what type of activity it was.

Table 9 Cross Tabulation of activity types by constructionism matrix dimensions for both coders showing % of type and (n) using Summary Dimensions.

		Problem			Design			Code		
		1	2	3	1	2	3	1	2	3
Activity Type	Lesson Plan	61%(19)	29%(9)	10%(3)	14%(4)	82%(23)	4%(1)	4%(1)	92%(21)	4%(1)
	Online Student Activity	71%(5)	29%(2)	0	57%(4)	43%(3)	0	33%(2)	67%(4)	0
	Board Game	0	0	100%(1)	0	0	100%(1)	0	0	100%(1)

We also compared the constructionism matrix scales to different types of artefacts created by activities, the results are shown in Appendix 2. The null hypothesis was that there would be no difference in the reported autonomy of students for whether the artefact was of a particular type or not. A Kruskal Wallis with bonferroni correction pairwise test showed ‘unplugged compared to physical’ (p=0.013, n=17. U=5.678, r=.694) and ‘on screen compared to physical’ (p=0.016, U=5.328, n=29, r=.517) had a statistically significant differences at the problem level both with large effect size, but no statistically significant difference for the others. However, caution is urged as there were only 2 activities creating this artefact type, each coded by a different coder. A cross tabulation (Table 10) showed that physical activities had a higher autonomy scale than non-physical for the problem dimension. This indicates that the type of artefact may be having a bearing on the degree of constructionism of a task, rather than the CT or CS concepts.

Table 10 Cross Tabulation of artefact types by constructionism matrix dimensions for both coders. showing % of type and (n) using Summary Dimensions

		Problem			Design			Code		
		1	2	3	1	2	3	1	2	3
Artefact type	Unplugged	75%(9)	17%(2)	8%(1)	12.5%(1)	75%(6)	12.5%(1)	25%(1)	50%(2)	25%(1)
	Onscreen	65%(15)	31%(7)	4%(1)	29%(7)	67%(16)	4%(1)	9%(2)	86%(19)	5%(1)
	Physical	0	50%(2)	50%(2)	0	4(100%)	0	0	100%(4)	0
	Concept	65%(19)	21%(6)	14%(4)	19%(5)	73%(19)	8%(2)	10%(2)	80%(16)	10%(2)

To investigate if there was any statistically significant difference in coders assignment of scales for each of the constructionism dimensions we performed the pair statistic of the Wilcoxon Signed Rank test with results shown in [Table 11](#). The null hypothesis was that there would be no statistically significant difference in scales across the problem and design, design and code and problem and code dimensions. There was statistically significant evidence to reject the null hypothesis, indicating that there was a relationship between coders allocation of scales across the dimensions. In our population coders rated the design scale as higher than the problem scale with moderate effect size, and the code scale higher than the design and the code scale higher than the problem (see [Table 11](#) for test results). This indicates that our coders rated activities as being more 'constructionist in the dimensions' of coding, then design and then lowest in problem.

Table 11: Comparing the changes in coders rating of scale across the constructionism matrix dimensions using the Wilcoxon Signed Rank test.

Test statistics	Comparing constructionism scales Problem to Design	Comparing constructionism scales Design to Code	Comparing constructionism scales Problem to Code
Increase in scale	12 (sum of ranks 90)	4 (sum of ranks 10)	14 (119)
Decrease in scale	2 (sum of ranks 15)	0 (sum of ranks 0)	2 (17)
Tied	20	26	14
n	34	30	30
p	p=0.008	p=0.046	p=.003
Z	-2.673	-2.00	-3.00
r	-.46	-.37	-.55

During review of activities, coders added notes to the mapping tool of their justification for their allocation of concepts and constructionism scale, these are shown in Appendix 1. We have not undertaken a qualitative analysis of this data, but draw upon these notes in discussion. In future studies these notes could be analysed more rigorously to inform & improve the suggested tool.

5. Discussion

An initial point to note was the difficulty in creating a method for evaluating the degree to which constructionism is incorporated in activities, a matrix has been suggested, but this requires further validation as a useful approach. When creating the matrix there was discussion of what were the most important features of a constructionist activity. According Papert (1980), important is learning experience that engage students in constructive activities that are meaningful to them. Furthermore, activities should be accessible to students with different styles of thinking and learning. Harel stressed that "students become deeply involved and gain deeper understanding... through the process of constructing, programming, and explaining their own representations" (Kafai, 1994, p. 24). Our focus led to the attribute of learner autonomy in creating artefacts at each of the main stages of a product development. Here an adapted levels of abstraction framework was suggested with dimensions of

problem, design and code. A scale of autonomy from levels 1 to 5 was suggested with 1 representing the least amount of learner control of the artefact being created to 5 representing the learners having complete choice over what to make and how.

During the process of review of activities, it appeared that the scale criteria description was not sufficient to distinguish between the sample of activities. Some coders rated activities at '1 to 2' or '1 to 3', this was because the activities, rather than being a single small activity were a unit of work across several lessons, or within a single lesson there was a 'graduation' of pupil autonomy. This 'graduation' may have been over the course of the lesson as different 'tasks' occurred, such as an initial closely controlled task, followed by a task where there was more student control or it could be that over several lessons pupils were moved from use to modify (Lee et al. 2011a).

In order to deal with this added requirement, two extra levels were added to the scale of ranges of '1 to 2' or '1 to 3' as coding value options. However, a % or degree of each level may have been more useful. Within the qualitative data of 'Justification of codings' variable' one coder said " I was loathe to allocate a level 2, as this only occurred in the final 20th challenge and up to this point there had been absolutely no student autonomy at all " (Coder 2)

However, during reporting on the data, these two extra 'range' scales were merged back in with their respective highest level, this was because of the small number of activities which were surveyed and the need for broader groups for statistical analysis. In further work, a larger sample of activities is needed and the more granular levels can then be used to draw out clearer distinctions between activities.

Turning our attention to the quantitative data reported, the coding of computer science concepts was reliable. There was a high level of inter-reliability agreement reported as being almost perfect ($\kappa = .868$, $p < 0.005$) in [Table 5](#).

For CT concepts the agreement was less but was still a substantial agreement ($\kappa = .74175$, $p < 0.005$). However, for the Constructionism matrix agreement was only moderate (.51, $p < 0.005$). Agreement was almost perfect on artefact type ($\kappa = 0.82$, $p < 0.005$).

Selection of activities to sample was problematical. In the UK there are a large number of resources available for educators to select from and recent surveys which have reported on the most popular (Royal Society, 2017). However, in other countries the task of finding resources to evaluate was not so easy. In other countries teachers can use resources that are internationally available, such as the code.org materials but working group members reported that in some countries teachers are more likely to create their own activities and that these are not then shared. For these countries, a generic set of activities were created to represent these countries resources. Whether or not these are representative of what is being actually used in class is difficult to assess. Similarly, in the UK, it is not clear as to whether teachers are using resources in their published form or if they are adapting them, in a recent survey of 207 teachers (data not yet published) 40% of teachers reported they created their own resources, including adapting resources from over 70 specifically named resource sets. The most popular three resource mentioned in this survey was the Barefoot materials (Berry et al. 2015) with 34% of the teachers mentioning using this resources followed by and 16% mentioning CodeIt and 13% Switched on, of which we have reviewed sample material in our survey here. Although, teachers choose the most popular computer science or computational thinking activities in the world, but sometimes the diversity of activities depends on policies, curriculum, grades, etc. For example, robotics has become very popular in secondary schools in Finland; computer science equipment is provided by the States and the local communities in Germany, programming with Scratch is involved as activity in primary school in Ireland; learning objectives include programming skills and knowledge of computer hardware in secondary schools in Lithuania; a course that includes programming with Scratch or Kodu is available in Portugal computer science education (7th-8th grade) (in Passey, 2017). Teaching of computer science integrated to other educational subjects is started in primary school in Lithuania in 2017.

In some cases, teachers use computer science activities as a way to demonstrate how the same problem or task of real life could be solved by using computers. For example, students have to solve problem/task during mathematics lesson, they try to solve its own or in pairs and at the end of the lesson solve the same problem using the particular tool, like Scratch, code.org, Codemonkey, etc. It gives

opportunity for students to understand the problem or task deeper or practice in different ways, engage discussion between students.

Similarly, there was debate as to whether to include activities in which no physical 'artefact' was constructed rather than learners 'constructed conceptual understanding. For one activity, a board game, the learners constructed their learning, as the 'used' others games, but then went on to create their own version, very much transitioning through a use, modify, create approach (Lee et al. 2011). This transition from using to making seemed to be the salient point in which constructivism (Piaget 1970) switches to constructionism (Papert 1970) and as long as activities included this transition then they could be included. Therefore, activities such as a Bebras task (Dagienė & Sentance 2016) would not be included, as they support development of a CT concept but do not include an element of then making or constructing of an adapted or brand new version of a task. This is a problematical issue as a teacher could use a Bebras task as a starting point and then adapt this context into a constructionist activity. How teachers are using activities in practice was not captured by our survey of materials. But our approach could be used as a starting point to further investigate teachers transition of resources from constructivist to constructionist activities.

In the case of creating Bebras tasks, Dagienė et al. (2016) mentioned a constructionist and deconstructionist learning ways. Constructive way of learning is the creation of computer science task and deconstructionist way of learning is that an computer science concept is analyzed and deconstructed in its main aspects (discovering why it is computer science).

Our review was of single activities rather than of sets of activities sitting with a progression of development of knowledge, skills and understanding. This is a limitation of our study. Similarly, some coders had considered the use of a construct such as a procedure or function was sufficient to warrant generalisation could be assigned, whereas for other coders, only if learners had themselves generalised rather than copied or used someone else's generalisation could that attribute be assigned to an activity. There are opportunities to review the progression of the constructionism matrix and CT concepts through time. For example, the Solo taxonomy (Biggs, 1982) starts with the learner having no experience of a concept and moves to them being able to apply the concept in new and novel scenarios. There is opportunity to map this taxonomy, or other similar to the constructionism matrix.

Looking at each of the dimensions for our activities, the fact that 52% to 62% of the problems were assigned a scale of 1

The majority of activities were coded at scale 1 for Problem or Task level at 52% and 61% for coders. The majority of designs were at scale 2 at 53% to 71%. Similarly, the majority of activities were rated at scale 2 for the code level at 57% and 61% as shown in Table 7.

Looking at the overall statistics for the constructionism matrix and its dimensions and autonomy scales, as shown in Table 7, there is an overall pattern whereby autonomy for learners is most restricted at the problem stage with around half of activities coded at scale 1. Autonomy for learners is slightly better during design and code where scale 2 was assigned for around half to three quarters of the time. What this indicates, for our sample, is that learners have little control over the context of the activities they are undertaking, they have some opportunity to start to modify the design and the code, but very few opportunities to be involved in a more free form create where they have control over what they might make it and how. Therefore, the level of constructionism might be considered to be LOW if we measure it based on student autonomy, learners are not moving to the point where they might feel ownership (Lee et al. 2011) as they are not yet making their own new artefacts.

Considering each of the computational thinking concepts and their relationship with constructionism and relating this to our findings.

ALT- Algorithmic Thinking

Despite having no statistically significant statistics related to our data for the algorithmic thinking CT concept, there are interesting features of the descriptive data. ALT was our most popular CT concept coded for the 21 activities, 16 to 19 activities (Table 5) were coded as teaching this concept by our 2 coders. When analysing the activities that were matched by the coders to this concept, the majority

of these were related to level 1 problem activities, level 2 design and level 2 coding activities ([Table 8](#)). The inter-reliability for ALT was almost perfect agreement ($\kappa=.877$ $p<0.05$) ([Table 6](#)).

Considering whether there might be specific reasons why algorithmic thinking might be particularly suited to constructionism then we turn our attention ideas on progression for this concept and the work by Rich, Strickland, Binkowski, Moran, & Franklin (2017) who suggested a learning trajectory for developing sequence. Rich et al. grouped initial learning about sequence into everyday activities related to ordering and precision, suggesting that learners transitioned from understanding the world around them for a particular concept to then applying this in a programming context with an intermediate phase of computational thinking. An example of a goal on this trajectory is “During this trajectory they need to learn that programs are made by assembling instructions from a limited set”(page 187). Whether this objective is more effectively met through CodeMonkey activities with no choice of the problem, the design or the code (as there is only one possible solution to a puzzle), or with Lightbot, with similar restrictions or through a more open exploration of a ScratchJr activity to draw a square, which still is a level 1 problem, but which allows learners to select the character that draws the square and the learner can choose where exactly to start, how big the square might be and can embellish this activity in next steps, so this becomes a level 2 design and code activity has not been evidenced by our study.

DEC - Decomposition

Just less than half of our activities were assigned to the decomposition concept (dependent on coder per [table 5](#)).These were relatively evenly split between level 1 and level 2 on the problem dimension with a couple of activities at level 3. For design most were at level 1 and most level 2 for the coding activities ([table 8](#)). The inter-reliability was at a substantial agreement ($\kappa=.704$ $p<0.05$) ([Table 6](#)). Why decomposition might have lower pupil autonomy at the design level is interesting, whether this is because design is lacking as a clear and identified step may be a contributing factor. A recent review of resources for teaching computing, concluded that there was a lack of resources with design (Falkner & Vivian, 2015) and a study matching learning goals against research cited design as the most unmatched goal (Rich, Strickland & Franklin, 2017). We did not specifically ask coders to reflect on this, however one coder noted that ‘Design element was STRONG’(coder 2) in activity 19 but nonexistent in the puzzle based activities which have a predetermined problem and solution algorithm. In undertaking design students are required to break their problem down into parts (decompose it), they need to consider the level of detail that is appropriate (and so are also abstracting) and order these items (and so are practising algorithmic thinking). Considering how students are experiencing design they may be copy an existing design, or working in a more constructionism way by creating their own, perhaps activities could be improved by increasing the element of independent design by learners in the same way that they are required to do when they are undertaking other subjects such as when learning to write (Waite, Curzon, Marsh, Sentance, Hawden- Bennett, 2018).

ABS - Abstraction

Only 4 to 5 activities were identified as teaching abstraction and the inter reliability was moderate ($\kappa=.696$, $p=0.001$), yet abstraction is seen by many as the cornerstone of computational thinking. Wing wrote “The abstraction process, deciding what details we need to highlight and what details we can ignore, underlies computational thinking” (Wing 2008, p.3718). The activities which have abstraction coded for them are predominantly at level 1 for problem and level 2 for design and code (see [table 8](#)).

In our study we have developed upon a particular use of abstraction through our constructionism matrix and the levels of abstraction, where the dimensions of our matrix represent levels of detail for different purposes of an activity, the problem, design and code, however that does not relate to how activities engaged with the teaching and learning of abstraction.

In looking at our coded activities, as with decomposition, learners are using abstractions as they are given a problem by their teacher or an online system, this is coded at scale 1 and what we saw for most of our small sample. If learners then follow a predefined design (as with the puzzle activities) they are using an abstraction and if they are copying or figuring out a pre-defined code solution, again they are using an abstraction at scale 1.

Despite learners using abstractions, few of our activities were coded with this concept. Perhaps this is because the concept is hidden from both the teacher and the learner, and only when a specific process of abstraction is mentioned would it be coded? We have 2 activities coded level 2 for design and 1 for level 3 at code and design. This is for activity 19, a ScratchJr which the coder justifies her allocation for by saying

'The design element is STRONG. I wish I could have given this a much higher score - maybe this needs to be reflected - but the genre was fixed and the language used. I allocated more CT aspects as kids were deciding on their own designs so abstracting and decomposing as they were designing. I am not sure they did generalisations - maybe because they were doing repeats ... need to discuss more about how we allocated each of the CT concepts - need a clearer definition.' (Coder 2)

However, for this same activity coder 1 disagreed and only gave the activity a level 2 for design and coding and did not allocate decomposition or abstraction as concepts taught.

The comments made by coder 2 implies a constructionism approach was being taken, as learners were 'deciding on their own designs' however, whether this means that learners will make more progress rather than using someone else's is yet to be robustly evidenced for this age group of learners.

GEN - Generalisation

This concept was assigned the fewest times from all of the CT concepts, only 1 to 3 activities were coded as GEN and the inter-reliability was not reported by SPSS as there were no agreements and numbers were small. Even where it was assigned there was a lack of certainty of the allocation, one coder remarked in the justification notes (Appendix 1) about Activity 21 (Lightbot) "Although this teachers procedures - is this generalisation or is it decomposition??? As the purpose is not to reuse for a logical reason but to reduce code blocks used." (Coder 2) The same coder raised the same question about Activity 19' I am not sure they did generalisations - maybe because they were doing repeats ... need to discuss more about how we allocated each of the CT concepts - need a clearer definition.'" (Coder 2)

Again this was raised for Activity 24:

'If we are considering repeats as decomposition then this is used, I am assuming then as it moves to functions then we can say that generalisation is used HOWEVER the level of use of these is very low. On solo it's really just first level as using what is provided - maybe unistructural, whether we think this use of a procedure that someone else made is generalisation I am really not sure it will be interesting to see what the other coders thought and whether the learning of generalisation here has been done in a constructionist manner at all.'" (Coder 2)

Two issues are raised by these comments, firstly, is a function or procedure which is used solely to reduce the number of lines of code for something that can be repeated a generalisation or a decomposition used within a repeat? Does generalisation need to be where we are "transferring a problem solving process to a wide variety of problems" (CSTA & ISTE, 2011)? Has this confusion arisen as the term pattern has been associated with generalisation? This term may have been introduced to simplify language and as an instructional approach in a progression of learning for generalisation to encourage the spotting of patterns that might then become generalisations. However, during this progression reusing code to simplify code within a single solution, such as in the activity 19,21 and 24 is this generalisation?

Secondly, does using elements created by someone mean we are learning about the process involved in creating those elements?

EVA - Evaluation

Evaluation according to Bloom is the highest order thinking skill (Bloom, 1957) which learners encounter as they gradually spiral through a succeeding progression of more complex material. In our review of the 21 activities, 7 to 9 activities were identified as teaching evaluation with substantial inter-rater reliability ($\kappa=0.69$ $p=0.001$). Of those activities assigned this concept by the two coders, over half were

rated as level 1 (54%) and a third (31%) level 3 for the problem, nearly three quarters (72%) were rated level 2 for design and 82% at level 2 for coding. This is quite a wide spread, with some tendency to the higher levels for code and design. Perhaps implying that evaluation of the code was more prevalent than evaluation of a design. But this is a broad assumption which needs more careful analysis particularly around what we might mean by an activity having evaluation being assigned as teaching that CT concept. In our definition of evaluation the following phrases were provided to help coders decide if evaluation was a skill USED in the activity 'Find an appropriate solution' Finding the best solution' 'Deciding whether the solution is fit for purpose' 'Deciding whether the solution is the most efficient one'. This list implies an order of progression. Surely the first of which must have been seen in any activity that had some kind of solution. Therefore all activities, if they solved any kind of task or activity should have had an element of evaluation. It may be that coders discounted this first example and only assigned where there was a 'best solution' considered, or if there was a design to which the solution to could be compared to decide if it was fit for purpose. There appears to be more work to be done in defining the rationale for assigning a concept as being associated with an activity, perhaps as a grading of association.

Our quantitative data should be viewed with much caution as our sample size was small at only 21 activities, and only 3 authors coded, each doing a mixture of 1st and 2nd coding. However, this small number of activities became 42 cases to review researchers views of the attributes of activities. Our approach for classification appears to have some reliability, as there was substantial agreement across the variables and coders. However, for some activities there was very different allocation of the constructionism scale, such as for the activity 25 the JavaScript onscreen activity which was the only activity to be awarded more than a scale of 3. It was allocated a 1-2 by one coder for all three dimensions but a 5 for just the problem and coding and a 1 for design by the other coder. The rationale for this may be that the activity sat within an overall complete unit of work, which eventually might lead students to be given a chance to create a new artefact using any product, in any context, but within the activity reviewed this was not the case. This indicates an issue with how an instrument to measure the constructionist aspect of activity might scope the boundaries of an activity.

Despite the limitations of our quantitative work, a number of interesting data results have been revealed, such as there being no statistically significant differences between activities with each of the CT and CS concepts across the scales of each of our problem, design and code dimensions of our constructionism matrix and yet there were differences for other aspects, including the activity type, such as lesson plan compared to onscreen student activity and also the artefact created such as a physical artefact compared to an onscreen activity or an unplugged activity. A resulting suggestion might be that CT & CS concepts are merely the content that is being delivered by a constructionism approach, the success of this approach is less impacted by the underlying material being delivered but more by the techniques applied to deliver it, such as through human (teacher) mediated activity as opposed to a symbolic (system) mediated (Kozulin, 2003) event, or through the creation of a physical artefact compared to a onscreen or unplugged artefact.

Our contribution to this field is to have suggested and trial a new mapping tool incorporating a new framework, called the constructionism matrix, for reviewing activities which are used in the teaching and learning of computing in terms of constructionism and computational thinking. We have identified limitations with our framework and with our trial, but from its development and using it with a small sample of lesson plan and online student activities (n=21) targeted at K-5 learners, we have been able to report the trial results of its first use, discuss problems encountered and suggest opportunities for improvement of both the mapping tool and the matrix.

We suggest next steps should be to further refine the constructionism matrix, by adding % of use of each scale and more closely defining the scope of the activities reviewed, it should then be used to survey a larger population of resources. Similarly, more work is needed to reflect on the depth of learning of CT & CS concepts perhaps in a similar scale as the student autonomy but perhaps using Solo taxonomy or other more finely grained frameworks of a degree of learning. Our mapping tool will then have a combined framework, which might provide more insight into the potential relationship between constructionism and computational thinking.

Acknowledgements

We would like thank you **Janne Fagerlund**, University of Jyväskylä for contributing to this working group paper.

References

- Aho, A., 2011. Computation and Computational Thinking. Ubiquity, 2011 (January), Article No. 1.
- Armoni, M., 2013. On Teaching Abstraction in Computer Science to Novices. *Journal of Computers in Mathematics and Science Teaching*, 32(3), pp.265–284.
- CSTA & ISTE (2011). Operational Definition of Computational Thinking for K–12 Education. Retrieved from <http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf>
- Bloom, B. S. (1956). Taxonomy of educational objectives: The classification of educational goals, handbook 1 cognitive domain. New York: David McKay Co. Inc.
- Bloom, B., Anderson, L., & Sosniak, L. (1994). Bloom ' s taxonomy, a forty year retrospective. Chicago, IL: University of Chicago.
- Berry, M., Woollard, J., Hughes, P., Chippendal, J., Ross, Z., & Waite, J. (2015). Barefoot computing resources. Retrieved from <http://barefootcas.org.uk/> Last accessed 8th August 2018
- Biggs, J. & Collis, K., 1982. Origin and description of the SOLO taxonomy. *Evaluating the quality of learning: The SOLO Taxonomy*. New York: Academic Press Inc, pp.17–30. Available at: <https://doi.org/10.1016/B978-0-12-097552-5.50007-7>.
- Bloom, B. S. (1956). Taxonomy of educational objectives: The classification of educational goals, handbook 1 cognitive domain. New York: David McKay Co. Inc.
- Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.
- Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. and Woollard, J., 2015. Computational thinking: A guide for teachers. *Computing at Schools*.
- Dagienė, V., Futschek, G., Stupurienė, G. (2016). Teachers' Constructionist and Deconstructionist Learning by Creating Bebras tasks. Constructionism in Action. Conference Proceedings. <http://e-school.kmutt.ac.th/constructionism2016/Constructionism%202016%20Proceedings.pdf>
- Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, 28(1), 23-44.
- Louis Cohen, Lawrence Manion, and Keith Morrison. 2011. Research methods in education. Vol. 7th Edition. Routledge.
- Cutts et al., 2012. The abstraction transition taxonomy: developing desired learning outcomes through the lens of situated cognition. In *Proceedings of the ninth annual international conference on International computing education research*. ACM, pp. 63–70. Available at: <https://doi.org/10.1145/2361276.2361290>.
- Dagienė, V. and Sentance, S., 2016, October. It's computational thinking! Bebras tasks in the curriculum. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (pp. 28-39). Springer, Cham
- Katrina Falkner and Rebecca Vivian. 2015. A review of computer science resources for learning and teaching with K-12 computing curricula: An Australian case study. *Computer Science Education* 25, 4 (2015), 390–429.

- Giordano, D., Maiorana, F., Csizmadia, A.P., Marsden, S., Riedesel, C., Mishra, S. and Vinikienė, L., 2015, July. New horizons in the assessment of computer science at school and beyond: Leveraging on the viva platform. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 117-147). ACM.
- Grover, S., Pea, R.: Computational Thinking in K12 A Review of the State of the Field. *Educational Researcher* 42(1), 38–43 (2013)
- Lee, I. et al., 2011. Computational thinking for youth in practice. *ACM Inroads*, 2(1), pp.32–37. Available at: <https://doi.org/10.1145/1929887.1929902>.
- Grover, S., & Pea, R., 2018. Computational Thinking: A Competency Whose Time Has Come. In S. Sentence, E. Barendsen, & C. Schulte (Eds.) *Computer Science Education: Perspectives on teaching and learning in school* (pp. 19–37). London: Bloomsbury Academic.
- Kafai, Y. B., 1994. *Minds In Play: Computer Game Design as a Context for Children's Learning*.
- Kozulin, A., 2003. Psychological tools and mediated learning, in: A. Kozulin, B. Gindis, S. Ageyev, Vladimir, and M. Miller, Suzanne, eds., *Vygotsky's educational theory in cultural context*, Cambridge University Press, 15–38.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- Passey, D. (2017). Computer Science (CS) in compulsory education curriculum: Implications for future research. *Education and Information Technologies*, March 2017, Volume 22, Issue 2, pp 421–443.
- Piaget, J. (1970). *Genetic epistemology*. New York: Columbia University Press
- Royal Society, 2017. After the reboot: computing education in UK schools. The Royal Society, 6-9 Carlton Terrace, London, SW1Y 5AG. Available at <https://royalsociety.org/topics-policy/projects/computing-education/>
- Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 Learning Trajectories Derived from Research Literature: Sequence, Repetition, Conditionals. *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 182–190). ACM.
- Kathryn Rich, Carla Strickland, and Diana Franklin. 2017. A Literature Review through the Lens of Computer Science Learning Goals Theorized and Explored in Research. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 495–500.
- Teague, D. and Lister, R., 2014. Programming: Reading, writing and reversing, in: *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, 285-290.
- Selby, C.C., Woollard, J.: Computational thinking: the developing definition, pp. 5–8 (2013)
- Settle, A., & Perkovic, L., 2010. Computational Thinking across the Curriculum: A Conceptual Framework. *Technical Reports, Paper 13*. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.910.8295&rep=rep1&type=pdf>
- Standl, Bernhard, 2017. Solving Everyday Challenges in a Computational Way of Thinking. In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, Cham, 180-191.
- Tedre, M., & Denning, P., 2016. The Long Quest for Computational Thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120–129). New York, NY: ACM.
- UCLA: Statistical Consulting Group. How can I calculate a Kappa Statistic for variables with unequal score Ranges? SPSS FAQ Available at <https://stats.idre.ucla.edu/spss/faq/how-can-i-calculate-a-kappa-statistic-for-variables-with-unequal-score-ranges/> Last accessed 8th Augusty 2018
- Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A., 2015. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715-728.

Jane Waite, Paul Curzon, D Marsh, Sue Sentance, and A Hawden-Bennett. 2018. Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal Of Computer Science Education In Schools* (2018).

Waite, J. et al., 2016. Abstraction and common classroom activities. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. ACM, pp. 112–113. Available at: <https://doi.org/10.1145/2978249.2978272>.

Waite, J et al. 2018. Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. In *International Journal of Computer Science Education in Schools*, Jan 2018, Vol. 2, No. 1 ISSN 2513-8359 pp. 14-40 Available at: DOI: 10.21585/ijcses.v2i1.23

Yang, S. and Park, S., 2014. Teaching Some Informatics Concepts Using Formal System. *Informatics in Education*, 13(2), 323-332.

Appendix 1 Coded activities & Qualitative notes

Activity Id	Activity/ Approach to Map	Rationale for choice	Activity Type	Description/Justification /Additional Comment(s) coder 1	Description/Justification /Additional Comment(s) coder 2
1	<p>Activity:Cody Roby URL:http://codeweek.it/codyroby/</p>	<p>Developed as an unplugged resource for teaching programming concepts, initially for Code Week Italy and has been adopted by CodeEU for Code Week</p>	Game	<p>This activity allows learners to either play a predetermined unplugged coding game or design/construct their own unplugged coding games.</p> <p>When learners play the game they have to think and abstract their strategy (conceptual artefact) and breaking down moves into potential smaller moves. Once a move is made, they a player will need to evaluate when has occurred and think about their next move. This becomes an iterative process as the game is play to an outcome.</p> <p>When a learner designs a game they create rules for playing that games and may create new cards (physical artefact).</p> <p>Academic Reference(s): 2in English</p> <p>Ferrari, F., Rabbone, A. and Ruggiero, S., 2015, September. Experiences of the T4T group in primary schools. In The Proceedings of International Conference on Informatics in Schools: Situation, Evolution and Perspectives-ISSEP.</p> <p>Klopfenstein, L., Fedosyeyev, A. and Bogliolo, A., 2017. Bringing an Unplugged Coding Card Game to Augmented Reality. INTED PROCEEDINGS, pp.9800-9805.</p>	<p>Many of the example games are algo rather than decomposition. But you could create games that also supported decomp. There is this issue of the 'surface' level understanding and depth that could be brought about by discussion, and also by creating games. I am finding it hard to say anything is NOT concept... as it is taking you to 1st Solo stage of just introducing a concept by experience. Also there is this whole problem of whether this is coding or an algorithm. The information processing agent is the person who executes it... This could be taken to the next level by actually implementing these in scratch... or other block based programming languages... it would be probably be</p>

					harder than I think.
2	Activity: Barefoot crazy characters - unplugged activity teaching about algorithms.	Top 10 of resources according to Royal Society report. Funded by DfE for resources for primary teachers to teacher new computing curriculum. Selected as specifically about teaching algorithms using an unplugged approach	Lesson plan	This activity introduces algorithms by teachers modelling the creation of a simple set of instructions (algorithm) to draw a character. Pupils follow the teacher's algorithm and then create their own. The teacher models how to evaluate the precision and completeness of the algorithm to introduce the idea of debugging. Children then construct their own algorithm for their own crazy character. They test out their friends algorithms to help them debug them. One could say that there is abstraction, as pupils have to consider what is most important to their character, and that they think of terms which are an 'abstraction' of how how to draw that part of the object e.g. draw a triangle. However, abstraction is not the focus of the activity. The focus is on introducing the vocabulary and concepts of algorithm and debugging and making a link between these terms and familiar instruction writing in English and Art lessons. Pupils are decomposing the task as they think of the individual steps and what to do next. They are sequencing their instructions, therefore one could link to this programming construct. But again this explicit link is not made. One could argue that the introduction of the term algorithm and debugging is constructing an abstract concept. However we need to consider whether this is going too far with the idea of constructionism (which we may need to further explore in our paper). The process of evaluation is explicitly built into the lesson through a box on the	

				worksheet requiring another pupil to test the algorithm out - so they are becoming the computational agent (could link to Wing and Denning argument). A physical artefact is created by the learner when they construct their algorithm.	
3	Activity: Barefoot beebot basics	Top 10 of resources according to Royal Society report. Funded by DfE for resources for primary teachers to teacher new computing curriculum. Selected as specifically about teaching programming using programmable toys	Lesson plan	In this activity pupils construct and solve programming challenges for a simple programmable toys. One could say that learners are constructing concepts (conceptual artefacts) in that they are constructing the concept of an algorithm and then constructing the concept of a program as they implement the algorithm as code. But as stated about I am not sure about whether this is all learning. There is a question here as to whether the algorithm is in fact code. As there is often a 1:1 mapping between the written commands and the buttons pressed (commands) which make up the code. This is suggested to be a group activity, with children taking on different roles of the algorithm creator and the implementer of the algorithm (the programmer), this is an early introduction of pair programming. The requirement of the pupil's to talk about their algorithm and whether it will or will not work through the use of a fake bot is significant. As they are using a tangible manipulative that represents the programmable device. Only when they have tested and debugged their algorithm do they then implement it as code. The use of the fakebot is important perhaps in interrupting a trial and error (random) debugging approach and could be argued to support the development of the 'conceptual artefact' of an algorithm (but I am not sure). The important	

				<p>thing for constructionism is that they are creating their own challenges - get the bot from the light green square to the yellow diamond avoiding all the circles. The use of challenge cards is start to develop further algorithmic thinking concepts, such as longest route, shortest route. The pause on, avoid encourage development of the skill of decomposition. Evaluation is encouraged through explicit use of the fakebot and reference to debugging of the implemented code. However, the amount of 'freedom' that pupils have in the context of the task will depend on the choices made by the teacher. If they use pre-made mats and no challenge cards, there is very limited pupil 'choice', if they ask learners to create their own mats (as suggested in the extension) then learners can have personally meaningful contexts. In terms of the design the choice is limited as is the implementation.</p>	
4	<p>Activity Barefoot Viking Raid Animation</p>	<p>Top 10 of resources according to Royal Society report. Funded by DfE for resources for primary teachers to teacher new computing curriculum. Selected as specifically about teaching programming using online programming language focused on teaching sequence and repetition</p>	<p>Lesson plan</p>	<p>In this introductory programming activity pupils create a simple animation using sequence. The algorithm is developed by learners taking a predefined background and set of sprite and deciding on their movement (action) and what they say (display) on a paper based design. Therefore there is very limited learner involvement in the definition of the task, or the design as they are constrained by the context and the commands being used. The code is developed by the learners being shown a command at a time by the teacher modelling its use. The learners then tinker with the commands that have been modelled. This is guided exploration and then implement</p>	

				<p>their design. Again the learners have limited choice on what they implement beyond the bounds of the commands introduced. For extension further commands (repeats) are introduced for learners to extend their design and implementation within the context.</p>	
5	<p>Barefoot maths quiz</p>	<p>Top 10 of resources according to Royal Society report. Funded by DfE for resources for primary teachers to teacher new computing curriculum. Selected as specifically about teaching programming using online programming language focused on teaching selection and variables</p>	<p>Lesson plan</p>	<p>In the introduction to selection and variables activity learners create the algorithm with a partner, using their knowledge of how simple quizzes work. The task is preset as a maths quiz with the requirement broadly to ask questions, check the answer and keep a score. The context is pre-set. To explore the commands that might be used to implement the algorithm learners are provided with a scratch file that has useful commands in the scripting area. BUT they are not snapped together. Learners then explore these commands to discover how they might work. Learners are 'constructing' their understanding as they explore. However, this is guided as they are given the commands to start with. Again one could say they are constructing knowledge of the selection construct as a conceptual artefact. Learners decided on the questions they want to include in their quiz, but the context and genre is constrained. For extension a buggy program is provided with a repeat and use of random variable block.</p>	
6	<p>Barefoot one of network ones</p>	<p>Top 10 of resources according to Royal Society report. Funded by DfE for resources for primary teachers to teacher new computing</p>	<p>Lesson plan</p>	<p>In this activity pupils create the internet by physically enacting the role of clients, routers and servers passing packets of data around. The concept of what is the internet, packets of data and the role of the components in a network are being learned. So here the pupils are constructing</p>	

		curriculum. Selected as specifically about teaching networks		an unplugged representation of a network, but the objective is constructing conceptual understanding. The data that is passed around is supplied by the teacher as are the components within the network. Therefor the pupils have no choice on what the context or genre or data is. The planning states that the lesson supports the development of abstraction as the learners are creating an abstraction of the real internet. However, they are broadly acting out someone else's view of an abstraction.	
7	Code-IT I am special ScratchJr	Top 20 of resources according to Royal Society report. Very popular resource created by a CAS master teacher, teacher trainer. This item selected as it is for programming with youngest age groups with scratch jnr	Lesson plan	In this introduction to programming activity learners create a simple animation about what makes them special. The idea of an algorithm being the sequence of statements that they have written, or had scribed for them on a simple planner, with the order shown by numbering. The context for the activity is set, as is the genre, the planning approach and what hardware and software is being used. This similar to other introductory activity. However, because the context is about the children's personal lives and something that is relevant to them this activity may be seen as being more constructivist? Is this true? Therefore even though there is no more/ or less freedom in terms of choices available, the relevance may be the thing that influences what is constructivist?	
8	Code-IT Jam Sandwich Bot	Top 20 of resources according to Royal Society report. Very popular resource created by a CAS master teacher, teacher trainer. This item selected	Lesson plan	In this activity the teacher pretends to be a robot and is instructed how to make a jam sandwich by the pupils. The teacher shows how instruction must be precise for the enactment to be successful. The task, genre, etc. are all predetermined. The learners are	

		as it is for teaching algorithms - very popular.		constructing their understanding of an abstract concept - a program for a computer to follow.	
9	Code-IT Crumble activity	Top 20 of resources according to Royal Society report. Very popular resource created by a CAS master teacher, teacher trainer. This item selected as it is for teaching physical computing with a programmable microcontroller	Lesson plan	In this activity learners make a simple display for any scene they are interested in and program the lights using a simple programmable controller called the crumble. This is a very simple design and technology, science and computing cross curricular lesson (plus art). The design is recorded as series of sentence which a says what the program will do in terms of controlling the lights. The teacher models the sentence - such as 'My program will 'make lights flash by replacing the stars in my picture with programmable lights. The user will see the stars twinkling in the sky and the moon lit up with a bright white light'. The teacher models each step of the addition of the components to create the circuit, and help cards are provided for pupils to follow. An example script is provided for pupils to predict what the code will do, and then to run and then to investigate before writing their own code. The physical wiring, and programming is limited but the context is not.	
10	Code-IT How an internet search works	Top 20 of resources according to Royal Society report. Very popular resource created by a CAS master teacher, teacher trainer. This item selected as it is for teaching using search in an unplugged way	Lesson plan	In this activity pupils complete a worksheet that requires them to find common classroom items e.g. scissors and write down where they find them. They then rank the locations. This unplugged activity introduces the idea that search engines present information in an order, a rank, that is theoretically useful for the person wanting to find things. It is constructing an abstract concept, through a familiar activity. The ranking is tested by a pupil from another class who is not familiar to the	

				<p>class environment. The pupils have no control over the context, genre or objects they are finding and the locations they are ranking, but they do decide on what the criteria for ranking is. As with the network activity you could argue that an abstraction is being learned about.</p>	
11	<p>Code-IT Magic Carpet</p>	<p>Top 20 of resources according to Royal Society report. Very popular resource created by a CAS master teacher, teacher trainer. This item selected as it is for teaching programming.</p>	<p>Lesson plan</p>	<p>In this introductory scratch activity an example game is provided for a specific context, but teachers could choose - or possibly pupils too - to create their game in different context (examples are provided). This simple offering of a range of other contexts may be significant in terms of choice given to pupils for their design and then build. 'Cross Curricular Focus this can be adapted to fit in with many topics or projects. It could be a lost cave dweller trying to find their way home without falling from the rock walkway, a bee that has to pollinate each flower or a spaceship that has to visit every planet. (See Travel Europe for Geography version)' http://code-it.co.uk/wpcontent/uploads/2017/01/MagicCarpetPlanning.pdf Had the other products given this simple option then the constructivist opportunities would have been increased. The general approach for the planning is to ask learners to USE the example program and spot the main features which they then list, this is a decomposition or abstraction perhaps of the main features. The teacher then models creating each feature and pupils then explore the commands associated with each feature and implement this part and tick off their list. The actual choice for implementation is low, as is the design. However, the context may be varied (if elected</p>	

				by the teacher)	
14	Switched on Computing We are Detectives	Top 10 of resources according to Royal Society report. Funded by a private education publisher. A popular and free sample.	Lesson Plan	In this activity pupils read emails, create emails and record information in a spreadsheet to solve a mystery. The emails are all pre-prepared by the resource creators, pupils construct reply emails and summarise data in a spreadsheet. Teachers can adapt the context for the activity but pupils are constrained by the flow of the events of the activity. They are constructing an understanding of email address formats and other ideas on emails. No specific reference is made to computational thinking, but in extracting key information one could say that abstraction is being practised (tenuous), and also the order in which things are happening could be sequence and CT (tenuous). This brings in play the whole discussion about what is CT and if it is thinking skills if you are not then moving on to use those skills in order to produce a program. And if the skills are not revealed to learners - are they CT. Else all learning where you have some dependency of task could be ALT.	
15	Switched on Computing Learn to code sample	Top 10 of resources according to Royal Society report. Funded by a private education publisher. A popular and free sample.	Lesson Plan	In this introduction to Scratch activity pupils follow step by step instructions to create a simple animation. At the end pupils are asked to change specific simple aspects and add extra suggested commands. A set of questions at the end ask a series of yes/no questions to encourage thinking about the task.	

16	Switched on Computing We are adventure gamers	<p>Top 10 of resources according to Royal Society report. Funded by a private education publisher. A popular and free sample.</p>	Lesson Plan	<p>In this introduction to Python pupils create an a simple interactive story which prints statements then the user answers a yes/no answer to move onto the next statement. The context for the story is open to the teachers and pupils to decide upon but the structure of the code that will be written is provided. The pupils replace displayed text in the example code given, which is modelled by the teacher with their text from their design. The concept of text based programming is introduced here by pupils being given and then typing in python commands. It is not quite copy code, but it is close. Having their own story context increases the degree of personal choice.</p>	
18	https://www.playcodemonkey.com	<p>This game is recommended for primary school in Lithuania according the project “Informatics in primary education“ https://informatika.ugdome.lt/en/about-project/</p> <p>CodeMonkey Awarded Best Coding & Computational Thinking Solution (https://www.playcodemonkey.com/blog/2018/06/20/codemonkey-awarded-best-coding-computational-thinking-solution/)</p>	Online activity	<p>In this game students learn to write code in programming language called CoffeeScript. The goal for student is to help monkey to catch bananas by writing lines of code. The game cover this computer science concepts: loops, objects, function calls, boolean conditions, function, conditions (if, if else).</p> <p>In the paid version teacher could get the curriculum explanation (teaching progress step-by-step explanation), track student progress. In the free version only 30 challenges, curriculum explanation are available. Curriculum has lesson plans, workshops</p>	<p>This actually teaches a misconception in terms of CT as it encourages learners NOT to decompose a problem, as if you want to try out 1 or 2 steps it says that is wrong. The task is set there is NO mention of design and the code has to be more or less identical to what was required by the developer. You may be out by 2 or 3 steps, but this is not particularly about choice. At challenge 14 we have the first code reading with code that is in the wrong order. Again this is teaching sequence (flow of</p>

					control) but not much else. At challenge 16 it suddenly introduces the term argument - it does not explain this AT all. At challenge 21 they introduce repeat loops I don't know if there is a sandbox at any point or when this activity stops
19	<p>Animated Genres: https://www.scratchjr.org/curricula/animatedgenres/full.pdf</p>	<p>This tool is recommended for primary school in Lithuania according to the project "Informatics in primary education" https://informatika.ugdome.lt/en/about-project/ In addition I looked to this article: https://ase.tufts.edu/devtech/publications/Portelance-2015-Constructing-ScratchJr.pdf</p>	Lesson plan	<p>During the curriculum, students created projects within three "animated genres:" collages, stories, and games During lessons students get to know about the main ScratchJr features and programming by blocks. Students are able to create their own projects by applying concepts learned in module lessons. In the first module during teacher and students discussion, teacher explain the meaning of instructions and sequences. After discussion students learn to move or use blocks in the scripting area, select blocks category. In the second module students learn to create sequences using variety of different motion blocks. Teacher demonstrate several example with different sequences. Students learn to use the start on green flag and end block, choose new characters. Teacher demonstrate the situation as unplugged activity and later ask student to write a program.</p>	<p>Lessons 1 is 1,1,1 but lesson 2 already becomes 2,2,2 but with very very close confines. As moves to 3 then pupils can choose their characters - but still tightly controlled on programming blocks. The design is through physical enactment. Lovely use of getting kids to predict what code might be used to implement the running of the code. Then there is a lovely open challenge where they make a collage... an animation. Then next few lessons introduce further concepts - guided exploration and again an open project where learners have more control to create a story type animation. The activities are</p>

					<p>creative - even with the jungle speed where learners have opportunity to choose an animal and then make them fast or slow and then race them. This planning is so different to codemonkey, or the khan java script activity. The design element is STRONG. I wish I could have given this a much higher score - maybe this needs to be reflected - but the genre was fixed and the language used. I allocated more CT aspects as kids were deciding on their own designs so abstracting and decomposing as they were designing. I am not sure they did generalisations - maybe because they were doing repeats ...</p>
20	<p>Learning ScratchJr via Playground Games</p>	<p>https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8363498</p> <p>https://dl.acm.org/citation.cfm?id=2532751</p>	Lesson plan	<p>In this curriculum (lesson plans) teacher shows features of ScratchJr and students create a specific playground game using presented features. Lessons plans is based on the discussion and practical programming ScratchJr environment. Teacher can adapt prepared curriculum or extend lessons.</p>	<p>Starts with very simple activities 1,1,1 draw a square, then make the cat walk diagonally and do a cartwheel 1,1,1/2 By lesson for it is slightly less 'copy my design' - which is verging on copy code to become sing the Hokey Pokey and a</p>

					dancing cat - this is harder than it looks - they could have made it have far more choice by saying any song. The amount that is covered in these 8 sessions is quite ridiculous. It forces copy code by having far too much covered - particularly as this is for such very young learners. I feel as though the lack of creativity until the later lessons is not being reflected in the coding of the activity for the research. Those less able pupils are very much being restricted here - and actually I don't believe that young learners would not go of and do their own thing much earlier on.
21	LightBot is a puzzle game based on coding. URL: http://lightbot.com/index.html	According https://venturebeat.com/2014/06/03/12-games-that-teach-kids-to-code/ and https://dl.acm.org/citation.cfm?id=3017728	Online activity	Lightbot is an educational game for kids that introduces several principles of programming. Children will practice concepts like sequence, conditions, and loops without typing or coding. Use problem solving skills to complete the puzzles (http://www.abcya.com/lightbot.htm)	There is NO choice or creativity here the problem, design and code are all predetermined. It will be interesting to see what the other coder has coded. Although this teachers procedures - is this generalisation or is it decomposition?? ? As the purpose is not to reuse for a logical reason

					<p>but to reduce code blocks used... it is dependent on you knowing your right and left and having good spatial awareness - but I like the music - the sprite box is not in lightbot now - this had will loops be associated with decomposition and procedures with generalisation - again just because I used one for someone else's problem does that mean I can generalise the concept of decomposition etc... we should have used solo with CT.</p>
23	<p>Choose Your Own Adventure</p>	<p>It focuses on excellent instruction with group and independent practice activities that build creativity, communication, and collaboration.</p> <p>Their goal is to reach all students and see computer science become part of a complete elementary education (URL: https://repositorio.grial.eu/bitstream/grial/1068/1/TEEM_2017_paper_03_preprint.pdf)</p> <p>https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnum</p>	Lesson plan	<p>Students tell a story by following a sequence, using statement. Brainstorm is used to find out the meaning of computer science and technology connecting daily life. Students create an anchor chart with CS words. Teacher introduce concept "conditions" by code.org video example and using decision tree on a an anchor chart or board. Discussion about story, teacher introduce Kodable world storyline and decision tree template, present how to fill in the decision tree graphic organizer to guide their writing process. Teacher present google forms.</p>	<p>This is interesting as the implementation is onscreen but via google forms. The focus is on learning flow of control and selection. design is strong - it's a shame that the google form video does not exemplify using the design. I think evaluation is used - but this is probably true now I come to think of it for any</p>

		ber=7860198 According https://www.sciencedirect.com/science/article/pii/S2212868917300338			
24	Frozen	Code.org provides sequences of videos and puzzles where users control characters from popular games like Rovio's Angry Birds or movies like Disney's Frozen with drag-and-drop programming. Using programming by demonstration to seamlessly integrate gameplay and educational content (citation from https://dl.acm.org/citation.cfm?id=3102106)	Online activity	In this activity teacher introduce these concepts: code, debugging, program. "Using programming by demonstration to seamlessly integrate gameplay and educational content" (https://dl.acm.org/citation.cfm?id=3102106)	A pure copy code activity, what is interesting is the age groups of Frozen and required reading vocabulary and maths. If we are considering repeats as decomposition then this is used, I am assuming then as it moves to functions then we can say that generalisation is used HOWEVER the level of use of these is very low. On solo it's really just first level as using what is provided - maybe unistructural, whether we think this use of a procedure that someone else made is generalisation I am really not sure it will be interesting to see what the other coders thought and whether the learning of generalisation here has been done in a constructionist manner at all. I am not sure ... this learning is so

					'hidden' as an experience rather than anything that is noted or over. The amount of reading skill is high. It's such a shame there is not more code reading. YOU HAVE TO DO 19 activities before you are finally given a sandbox - I was loathe to allocate a level 2, as this only occurred in the final 20th challenge and up to this point there had been absolutely no student autonomy at all , The design really is none existent and is not at all exemplified during the activities 1 to 19 except from defining what you have to reproduce. You can't create your own procedures... missed opportunity
25	Intro to JS: Drawing & Animation	According Khan Academy answer about the most popular course	Online activity	This is JavaScript course. During this course students have to work through the tutorials, solve challenges, try some projects (start with an existing project or modify it). Writing code canvas is updated according written code. This course runs through the following sections: Intro to Programming (which contains a video explaining their approach), Drawing Basics, Coloring, Variables, Animation Basics, Interactive Programs, Text and Strings,	There are very tightly controlled initial tasks where you have to reproduce what they want you to code. It's not precisely copy code. But it is very close. It does suggest to create your design of the drawn image on graph paper first. You have no choice on the

				Functions, Logic and If Statements, Debugging Programs, Looping, Writing Clean Code, Arrays, Objects, Object-Oriented Design, Becoming a Better Programmer.	problem and very little choice in terms of the design or the code. Algorithms are not discussed explicitly but sequence is brought to attention of the learner. At the end of the sequence you then have a completely free option. It is very boring watching someone code. At the end of the copy code activities it then gives you a set of predefined tasks to choose from and a starter design and code to then amend. I chose 10 onwards for age as the maths and reading requirement is quite high as it is text based. I was loathe to allocate a level 2, as this only occurred in the final challenge.
--	--	--	--	---	---

Please note that not all coders completed the justification part.

Appendix 2 Mann Whitney U statistics comparing concepts, artefact and activity types to constructionism scale for each of the constructionism matrix dimensions.

Note scales where a coder had rated as 1 to 2 this has been added to scale 2 and where a coder rated 1 to 3 this has been coded as 3. Scales 4 and 5 have been removed as only 1 activity was coded to this scale by one coder, and the second coder rated this same activity as 3.

		Problem	Design	Code
Computer science (CS) concepts	ALP	p=.083 n=39 U= 144	p-.714 n=36 U=28	p was not computed
	DDS	p=.389 n=39 U=22	p-.665 n=36 U=41.5	p was not computed
	CPH:	Not computed	not computed	not computed
	C&N:	p=.179 n=39 U=40	not computed	not computed
	ISS:	p=1. n=39 U=112	p=.611 n=36 U= 88.5	p=.914n=30 U= 65
CT concepts	ABS:	p=.322 n=39 U=84.5	p=.199 n=36 U=72.5	p=.067 n=30 U=28.5
	ALT:	p= .701 n=39 U=166.5	p=.876 n=36 U=97.5	p=.914 n=30 U=60
	DEC:	p=.149 n=39 U=225	p=.814 n=36 U=168	p=.439 n=30 U=127
	EVA:	p=.268 n=39 U=206.5	p=.253 n=36 U= 190	p=.171 n=30 U=137
	GEN:	p=.389 n=39 U=22	p=.495 n=36 U=23	p=.331 n=30 U=15
Artefact type (Mann Whitney U)	Physical	p=0.007 n=39 U=125 Z = -2.952 r=.04 (medium effect size)	p=.576 n=36 U=76	0=.930 n=30 U=54
	Onscreen	p=.489 n=39 U=159	p=.280 n=36 U=111	p=.872 n=30 U=84.5
	Unplugged	p=.329 n=39 U=149	p=.327 n=36 U=174	p=.441 n=30 U=97
	Concept	p=.692 n=39 U=132	p=.475 n=36 U=151	p=.713 n=30 U=109

Artefact type (Kruskal Wallis with Bonferroni Correction comparing Physical onscreen and unplugged)	Overall	p=.012 n=39 $\chi^2=8.876$ d.f=2	p=.367 n=36 $\chi^2=2.007$ d.f=2	p=.968 n=30 $\chi^2=0.065$ d.f=2
	Pairwise	unplugged to physical p=0.013 Z=2.862, n=17 U=5.678 r=.694 (large) onscreen to physical p=0.016 Z=2.785 U=5.328 n=29 r=.517 (large)		
		unplugged- on screen p=1 Z=3.502 n=39 U=.402		
Activity Type (only comparing Lesson plans and Online as there was only one board game activity) Mann Whitney U	Lesson Plan / Online Student Activity	p=.535=38 U=94.5	p=.018, n=35 U=54.5, Z=-2.358 r=.398(medium)	p=.048, n= 29 U=37, Z=-1.978 r=.367(medium)
Activity Type with Kruskal Wallis with Bonferroni correction Overall Comparing Lesson Plan, Online and Board Game)	Overall	p=0.164 ($\chi^2=3.619$, n=39, d.f =2)	p=0.009 ($\chi^2=9.363$, n=36, d.f =2)	p=0.009 ($\chi^2=9.373$, n=30, d.f =2)
	Pairwise		Student Onscreen - Board game p=0.021 Z=8.816 U=2.690 n=36	Student Onscreen - Board game p=0.010 Z=6.166 U=2.946 n=30
			lesson plan board game p=.173 Z=3.485 U=2.234 Student Onscreen- lesson plan p=.076 Z=3.485 U=2.234	lesson plan board game p=.061 Z=35.831 U=2..946 Student Onscreen- lesson plan p=.228 Z=2.617 U=1.775

WG2: Developing Constructionism, or a New Learning Concept, across the Ages

Don Passey, *d.passey@lancaster.ac.uk*
Lancaster University, Lancaster, UK

Loice Victorine Atieno, *atienomunira04@gmail.com*
Eötvös Loránd University, Budapest, Hungary

Wilfried Baumann, *baumann@ocg.at*
Austrian Computer Society, Vienna, Austria

Valentina Dagiene, *valentina.dagiene@mii.vu.lt*
Vilnius University, Vilnius, Lithuania

Abstract

Curricula in many countries are adopting computing (or informatics) as a subject or discipline. Computing learning practices in these 'new' curricula can involve pupils from 5 years of age, either within a discrete subject, or integrated into other subject topics across the curriculum. However, the learning concept, framework or theory that such curricula are based on, is not clear in curriculum documentation.

Our curriculum concepts of learning progression are largely based on Piaget's research, who described learning as a form of cognitive constructivism, developing over the age span of young people, and progressing through a series of stages: sensorimotor; preoperational; concrete operational; and formal operational. The new computing curriculum and teaching and learning practices could well be placed within this conceptual framework. However, Vygotsky's research added a more social dimension of learning, a concept of social constructivism.

Our contribution will review the constructionism approach by Papert that is based on uses of digital and computing-based resources within a constructivist approach to learning, and will investigate what learning concept, framework or theory should underpin computing curricula to ensure that practices and outcomes are as effective as possible.

Keywords

Constructivism, Activity theory (Lev Vygotsky), Learning Theories

Background

Curricula in many countries are adopting, or have recently adopted, computing as a subject or discipline (such as the national curriculum for computing in England, 2013), although the subject may be given alternative names and have somewhat different concerns in different countries (computer science in the United States, Canada, or New Zealand, or informatics in Germany, Poland, or Lithuania, for example). Computing learning practices in these 'new' curricula can involve pupils from 5 years of age, perhaps within a discrete subject, or perhaps integrated into other subject topics across the curriculum. The learning concept, framework or theory that such curricula are based on, and therefore which can underpin practices and outcomes across the age span of learners, is not clear in curriculum documentation, however.

Western curriculum concepts of learning progression are largely based on Piaget's research (1936), who described learning as a form of cognitive constructivism, developing over the age span of young people, and progressing through a series of stages or phases: sensorimotor; preoperational; concrete operational; and formal operational. The new computing curriculum and teaching and learning practices could well be considered and placed within this broad conceptual framework. However, Vygotsky's research (1978) added a more social dimension of learning, a concept of social constructivism. The social constructivist approach to learning may be important to consider also when implementing a

computing curriculum. With the advent of digital and computing-based resources, further concepts of learning have been developed. From the perspective of computing, the most significant of these is perhaps the concept of constructionism developed from Papert's research (1991). In summary, over the previous 80 years, as digital resources have continued to be developed and used in education, so our concepts of learning have been reconsidered - from a cognitive individual perspective, to a social perspective with others (including teachers and peers), and then with digital resources (Passey, 2013). Our current contexts, however, are different from those previous times when researchers developed their learning concepts or theories.

This paper will consider this issue: what learning concept, framework or theory should underpin computing curricula to ensure that practices and outcomes are as effective as possible across the age span of that curriculum.

Research questions

- Can a concept such as constructionism still be reliably adopted and developed across the age range from 5 to 18 years?
- Do we need a new conception of learning that accommodates our current context?
- Is this concept or theory something that we can create from a research perspective, or can we do it from practical know-how and experience?
- What should we do in the future to underpin learning development in computing?

Approach

In this paper, we start to explore the research problem by taking a systematic analytic approach:

- Initially, we describe and detail the skills and competencies that are required of a contemporary computer programmer or software developer.
- We then take a number of relevant learning theories, describing and detailing their sources, and their main features.
- Using the features of each learning theory, we identify whether and to what extent there is a match to the approaches of skills and competencies required of a contemporary software developer, and whether certain skills or competencies would not be easily developed.
- We critically map the findings from across all the learning theories we have explored, to indicate where there are matches and where there are gaps, and consider whether a new learning theory is required, and how this relates to previous learning theories.
- From the outcomes of the mapping, we identify the rationale for, a possible name for, and detail the nature of any new learning theory proposed.

This is clearly an ambitious project. Hence, we do not foresee that we will provide in this paper a picture that cannot be further developed and questioned. It is more our intention to open up the research questions, which we see as being important strategically for the future of computing (informatics) development in education. Learning theories continue to evolve in parallel with social constructs; we see this paper as a contribution within that evolution.

Skills and competencies required of a contemporary software developer or computer programmer

According to the United Kingdom (UK) National Careers Service (2018), a software developer (programmer) could:

“work in a wide range of businesses and industries, public services, utilities, defence and research. ...work closely with project managers, business analysts and graphic designers, to find

out what the client wants and the best way to achieve it. Usually ...work in a team. ...could work on a wide variety of projects, from financial databases to robotics to apps for phones and tablets. ...may use a number of programming languages or project management tools. ...day-to-day tasks may include: talking through requirements with the client and the development team; taking part in technical design and progress meetings; writing or amending computer code; testing software and fixing problems; keeping accurate records of the development process, changes and results; carrying out trials and quality checks before release; maintaining and supporting systems once they're up and running. As an experienced developer ...supervise a programming team and provide feedback on coding work.” (National Careers Service, 2018, p.n.p.)

There are clear links for an individual involved in this work to a computing (informatics) curriculum. The skills and competencies of a contemporary software developer can be considered as an end product of computing or computer science education. But, in terms of developing those skills and competencies, they should adequately match or be enabled through learning activities that are consistent with any appropriate learning theory or theories that traverse the age span of learner and learning development. So, we need to detail these skills and competencies, within a contemporary rather than historic context.

The skills needed by a software developer can be analysed through nine different elements or processes:

- Conceiving – taking user, market, technical and end-product requirements into consideration, producing ideas or drafts of a software process or product;
- Planning – exploring scope and defining specific user requirement, outlining and detailing the overall process and time plan leading to a final outcome;
- Designing – generating an overview or high-level design of the outcome, identifying specific elements or modules of a program, how they integrate, and what language, operating system and hardware components might be involved;
- Developing – prototyping to consider proof-of-concept or trials of possible alternatives, leading to implementation that involves programming the code;
- Documenting – detailing the internal design, so that the software and process can be reviewed, revised and maintained in the future;
- Debugging/testing – finding and resolving problems that stop efficient and effective use of the software, and judging the quality of the outcome compared to initial requirements;
- Deploying – releasing the software for use, with additional concern for customisation;
- Evaluating/improving – beyond the initial deployment period, taking ongoing feedback into account, and exploring how to improve, perhaps integrating further or new software facility; and
- Maintaining software – picking up on problems or issues over longer periods of time, and exploring ways to resolve these within contemporary situations.

Skills are developed through experience, and experiences can be divided according to the form of interaction taking place – whether from the individual’s interest or concerns, whether through formal education routes, or whether through work and employment. Skills might arise, therefore, through (although specific skills are not necessarily associated with each of the following dimensions):

- Talent; such skills are difficult to ‘teach’ or ‘acquire’, but it is important that they are recognised and discovered. Some people appear to naturally engage with certain processes of software development, exhibiting specific talents, while others do not naturally engage in the same way. How such skills arise, whether from other experiences, or from role models, for example, is not easy to identify, but it is important for educators to be able to identify these. But persistence and experience can in many, but possibly not all, cases make up for differences in skills arising from talent.
- Education; those skills are abilities associated with a formal education.
- Practical experience; these skills are acquired on-the-job when working in the field. They not only encompass technical skills, but also organisational skills, such as planning, structuring and emotional organisational intelligence.

Skills can be categorised in a number of different ways. Lamb, Maire and Doecke (2017) categorised 21st century skills as: critical thinking; creativity; metacognition; problem solving; collaboration; motivation; self-efficacy; conscientiousness; and grit or perseverance. Clearly, this categorisation focuses on what might be regarded as ‘soft skills’, without emphasising skills that might be regarded as operational or technical (manipulative and mechanical, for example). SkillScan (2012) takes an alternative approach, categorising skills as: relationship; communication; management/leadership; analytical; creative; and physical/technical. For this paper, the skills are broadly categorised into three different groups that are formed from a coalescence of these sources:

- Technical skills; these involve having a good understanding of the theoretical background of the field as well as of the common practices used and discussed by the community. As SkillScan (2012) state, these skills include those that enable effective interactions between the individual with physical objects including machines and technological systems.
- Analytical skills; these include the abilities to collect and analyse information, problem-solve, and make decisions. Such skills are concerned with logical thinking, mathematical reasoning, and structural reasoning (planning). Critical thinking is an important basis for this category. People with this skill see trouble-shooting as a challenge rather than a nuisance. The skill to have great attention to detail and to consider overall context at the same time is important, but often found difficult to master. People with good analytical skills need to quickly switch between abstract and practical viewpoints. They must be able to master complexity. In particular, they should be able to estimate project cost and understand the scope of projects.
- Social and emotional skills; software developers should have very good communication skills when communicating with team members, customers, their team leader, etc. Writing and understanding documentation like user manuals, training materials, tutorials, and trouble-shooting guides are also part of the overall and necessary communication effort. It has been found that effective developers are willing and able to share knowledge and expertise with fellow team members. Depending on their work situation (whether a lone warrior, team player, etc.) they need to be able to work either independently or within groups. They should be able to analyse user needs and have a solid understanding of a company’s needs. They should be able to scrutinise all the information they are given, anticipate events that are hard to predict, and handle both pressure and failure.

Some of the skills listed above are more clearly obvious as critical requirements, and they are often easier to assess. Some of these are more related to short-term success, and those would be expected to be in high demand in job advertisements. Some skills are more subtle, and are more related to long-term success.

In summary, and taking the range of required skills that will be used in the next section of the paper, our framework for analysis is shown in Table 1. Using this framework, we consider whether each learning theory we have selected can underpin the development of learning activities required to develop each of these skills. We answer the question: is it possible to understand how this learning theory relates to or explains or underpins the development of learning activities to support each of these skills?

Table 1. Framework of skills required by contemporary software developers

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
Conceiving			
Planning			
Designing			
Developing			
Documenting			
Debugging/testing			
Deploying			
Evaluating/improving			
Maintaining			

Matching learning theories to skills and competencies of contemporary software developers

In this section, we take a number of learning theories, chosen because they either often underpin educational curricula and practices, or are associated in certain ways with computing (informatics) education and practices. For each learning theory, we consider their source(s), their main features, the match to the development of learning activities that can support the skills and competencies of a contemporary software developer (taking the skills of a software developer identified in the previous section), and highlight those skills and competencies not clearly supported through this learning theory approach.

Cognitive constructivism

Source: Cognitive constructivism is a theory that emerges from Piaget’s studies on cognitive development (Piaget, 1936; Piaget & Cook, 1952). Piaget did not examine the development of learning per se, and his theory focused only on early learners (up to 11 years, and from beyond 11 years of age, the development identified was considered to go across the lifespan). His evidence was gathered from a small, discrete sample (3 of his own children, and some of his colleague’s children in his later work).

Main features: His theory identifies three important components of development: schemata (building blocks in memory that can develop arenas of knowledge or understanding); adaptation (how an individual moves from one stage of development to another); and stages of cognitive development. He described schemata as sequences of actions that can be held in memory, providing blocks for the retention of and building of knowledge, ideas and understanding. He described adaptation in three stages: assimilation (using an existing schema to explain or work with a new idea or piece of knowledge); accommodation (realisation that an existing schema does not apply and needs to be adapted); equilibration (when a schema needs to be replaced, and the challenges associated with doing this). He described the stages of development as: sensorimotor (up to 2 years, when mental representations can be held in memory or mind); preoperational (2 to 7 years, when an object or word can be symbolically associated with something else); concrete operational (7 to 11 years, when internal thinking can occur without having to do this physically); and formal operation (from 11 years, when thinking can be logical and about abstract concepts). His later work (Piaget, 1958) indicated that adaptation required an active learning involvement, and that problem-solving needed to be experienced rather than learnt. Whether stages do exist, and what the influences of social and cultural factors might be, were not explored within his research. The role of language, and the concept of schemata, have both been questioned by other researchers.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 2 (at this stage, interpreted by members of the working group only).

Table 2. Framework of skills related to cognitive constructivism

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
Conceiving	√ from 2 years of age	√ from 11 years of age	?
Planning	√ from 11 years of age	√ from 11 years of age	?
Designing	√ from 7 years of age	√ from 11 years of age	?
Developing	√ from 7 years of age	√ from 11 years of age	?
Documenting	√ from 7 years of age	√ from 11 years of age	?
Debugging/testing	√ from 11 years of age	√ from 11 years of age	?
Deploying	√ from 7 years of age	√ from 11 years of age	?

Evaluating/improving	√ from 11 years of age	√ from 11 years of age	?
Maintaining	√ from 11 years of age	√ from 11 years of age	?

Skills and competencies not covered: Taking Piaget’s stages of development as a guide, it is possible to see how some technical skills could be developed from 2 years of age, while others could be developed from 7 years of age. If all of the skills could not be developed until 11 years of age, then this places a wide of learning activities out of the central context of contemporary practice. Additionally, the stages of development do not allow a clear understanding of how necessary social and emotional skills might be developed.

Social constructivism

Source: Social constructivism is a theory of knowledge in sociology and communication theory that examines the development of jointly constructed understandings of the world that form the basis for shared assumptions about reality. Social constructivism has been studied by many educational psychologists, who are concerned with its implications for teaching and learning. The social theory of knowledge concerns human growth that is socially organised and knowledge created through cooperation (McKinley, 2015). The approach has been influenced by Vygotsky’s work and is centred on the social context of learning, in which he believes that knowledge is collectively created and assembled (Bodrova & Leong, 2012; Gaurain, 2008). Through interaction, learners get the chance to express their thoughts, hence creating a common understanding associated with the idea (Kalpana, 2014). Gredler (2008) believes that as we move from the theory developed by Piaget towards that of Vygotsky, there is a shift of ideas from individualism to collaboration, aided performance, social communication and sociocultural activity. According to this theory, learning is often socially constructed, perhaps as a model. Constructing a model can constitute making a sequence of ideas that are public, for discussion and change; this is similar to programming activity, where Papert indicated parallels between professional programmer work and children’s programming actions.

Main features: Conversation is the most important means of maintaining, modifying and reconstructing subjective reality (according to Berger & Luckmann, 1991). Vygotsky’s theory is based on 3 major themes namely: social interaction, the More Knowledgeable Other (MKO) and the Zone of Proximal Development (ZPD). Social interaction enables the process of cognitive development where learners exhibit both social and individualistic functions; learning through The More Knowledgeable Other (MKO) concerns learning through someone more familiar with the subject under study. They may be a teacher or coach or an older person, a peer, younger person or even a computer; and the Zone of Proximal Development (ZPD) is the cognitive gap or difference between the learner’s ability to perform a task with the help of another or through collaboration and the time the learner performs the task independently.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 3 (at this stage, interpreted by members of the working group only).

Table 3. Framework of skills related to social constructivism

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	√	√

Skills and competencies not covered: Whilst it is possible that all skills could be developed by considering this learning theory as a background, the roles of different social actors will clearly be important here. How teachers and significant others are involved, including the extent and their forms of interactions, could easily determine the feasibility of any of these forms of skill development.

Social learning

Source: This theory was introduced by Albert Bandura with the intention of clarifying how youngsters learn in social conditions by watching and afterwards mimicking the conduct of others. The theory postulates that individuals learn from each other, though observation, simulation, and demonstration. The theory has frequently been termed as a link between behaviourist and cognitive learning theories since it incorporates attention, memory, and motivation. Bandura believed that learning could not be exhaustively clarified through reinforcement, but the existence of others played a big role. The outcome of a learner's conduct was seen more as a result of watched behaviour than one they had adopted themselves. The results were gathered from a series of experiments where the youngsters watched as the grown-ups attacked Bobo Dolls. As the doll was hit, it fell down and bounced up again. When the youngsters were set free they imitated the forceful conduct of the grown-ups. He also observed that the youngsters were less ready to mimic the grown-ups when they saw them being rebuffed after the act.

Main features: Social learning has four elements: observational learning, reciprocal determinism, self-regulation, and self-efficacy. Observational learning involves observing a model doing something and then imitating the same. This can be achieved by observing someone, a description in a book or even a movie. For observation to take place successfully, it must be given attention, retention of what is observed and motivation of learners to apply what has been learnt. Reciprocal determinism describes the influence of the environment on the learners. It advocates for a positive learning environment if learning is to take place. It is more concerned with social interaction between the learner and others in the environment. The third element is self-regulation which involves setting of goals, discipline for actions and follow-through for learning to occur and performance to improve. Lastly, self-efficacy involves the belief the learner has in their capability to learn and perform. Hence, creation of an environment that foster confidence in the learner should be encouraged and cultivated.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 4 (at this stage, interpreted by members of the working group only).

Table 4. Framework of skills related to social learning

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	√	√

Skills and competencies not covered: Concerns with using this background learning theory are similar to those posed in discussing social constructivism. Although all of these skills could be developed, it is the order of these, and the importance of developing and subsequent reliance upon metacognitive skills such as self-regulation, that could affect the relationship of the development of these skills.

Constructionism

Source: This is an educational development theory developed by Seymour Papert of the Massachusetts Institute of Technology (MIT) and is based on Piaget's theory of knowledge. Piaget believed that knowledge is actively created, based on world experience, hence constructionism. He came up with several tasks and questions that revealed the different kinds of knowledge structure built by children at different ages. Evidence was gathered, based on how children of different ages constructed their knowledge, given the same task. Based on what Piaget had learnt from the children, Papert used it to rethink educational practice. Constructivism is a well-known theory of learning indicating that learners actively construct new knowledge by combining their experiences with what they already know. A similar sounding concept "constructionism", developed by Seymour Papert, takes constructivist theory a step further towards the points of action. Although the learning happens inside the learner's mind, this happens when the person is engaged in a personally meaningful activity outside of their mind. This can then make the learning interesting, real and shareable. Papert defined constructionism as: "From constructivist theories of psychology we take a view of learning as reconstruction rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that learning

is most effective when part of an activity the learner experiences as constructing a meaningful product” (Papert, 1986; Bruckman, 2006). Four aspects are essential to the design of constructionist learning environments. These are: learning through designing, personalising, sharing, and reflecting. Constructionism is grounded in the belief that the most effective learning experiences grow out of the active construction of all types of things (Papert, 1980). Constructionism strongly resonates with the current maker movement (Martinez & Stager, 2013). Constructionism as a theory symbolised a way of thinking about learning, the most general and effective being: building a model, reflecting on it, testing it, and sharing with others. DiSessa and Cobb (2004) argued that constructionism is “learning by designing” and falls into a category they called “frameworks for action”; however, they do not provide scientific ideas and structure for the design of learning environments. From that perspective, Noss and Clayson (2015) suggested a constructionist agenda and defined six characteristic: (1) modelling, (2) accessibility to the modelling process, (3) layering of principles and abstraction, (4) tapping into youth culture, (5) being represented in learner’s language, and (6) collaboration. Many educators and cognitive psychologists have applied constructivism to the development of learning environments.

Main features: Constructionism is built on the main idea that knowledge is not passively received either through the senses or by way of communication – it is actively built up by the person during the learning process. Another strong feature stressed by Glasersfeld (1988) and cited in various papers, is that the function of cognition is adaptive and serves the subject’s organisation of the experiential world, not the discovery of an objective ontological reality.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 5 (at this stage, interpreted by members of the working group only).

Table 5. Framework of skills related to constructionism

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	√	?

Skills and competencies not covered: Whilst it is feasible to consider that all skills could be developed using this background learning theory, the importance of development of social and emotional skills, and the need for these to be introduced at a young age, is stated more clearly in later definitions and discussions of constructionism. If there is a focus on interaction in learning activities with material resources, at the expense of interactions with peers and others when doing this, then this could hinder the important development of relevant and related social and emotional skills.

Situated learning

Source: This is an instructional approach developed by Jean Lave and Etienne Wenger in the early 1990s. The approach is based on the works of Dewey, Vygotsky and others who believed that learning takes place more if learners are actively involved in the learning process (Clancey, 1995). It is believed, however, that the idea of situated learning was first used by Brown, Collins and Duguid (1989) to develop a proposal for an instructional model that has inferences to classroom practice. This was as a result of the researchers observing successful learning situations. Examples of effective learning in any context or culture were identified and key features of the models analysed (Herrington & Oliver, 1995). Situated learning is a theory about the nature of human knowledge, claiming that knowledge is dynamically constructed as we conceive of what is happening to us, talk and move, but is concerned also with how individuals acquire professional skills. Our action can be situated in our role as a member of a community. Situated learning focuses on the relationship between learning and the social situation in which it occurs. The theory of situated learning claims that knowledge is not a thing or set of descriptions or collection of facts and rules.

Main features: According to Herrington and Oliver (1995), the principal theorists and critics of situated learning believe that learning environments that possess the following features elicit effective attainment of usable knowledge: authentic context, authentic activities, expert performances and the modelling of

processes, multiple roles and perspectives, collaborative construction of knowledge, coaching and scaffolding at critical times, reflection to enable abstractions to be formed, articulation to enable tacit knowledge to be made explicit, and lastly, integrated assessment of learning within the tasks. Situated learning declares that learning: (1) should be always integrated with the individual's identity and participation; (2) is constituting an evolving membership and capability to participate in different forms; and (3) is the means of reproduction and development of communities of practice.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 6 (at this stage, interpreted by members of the working group only).

Table 6. Framework of skills related to situated learning

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	√	√

Skills and competencies not covered: While situated learning would appear to be an ideal background learning theory to frame learning activities to develop all of these skills, it is difficult to see how this might be applied in practice for young people starting their involvement from, say, 5 years of age. Success would be likely to depend on having access to teachers and resources that would be competent and confident in terms of creating a situated learning environment.

Discovery learning

Source: Discovery learning established its base from theories developed by John Dewey (Dewey, 1997), Jean Piaget (Piaget, 1973), and Lev Vygotsky (Rice & Wilson, 1999) that described learning as active, process-based, and collaborative. It involves an instructional model and strategies that presents learners with more active and practical learning opportunities (Dewey, 1997; Piaget, 1973). According to Berding (2000), Dewey believed that children were naturally motivated to actively learn, and that education only served to make more learning possible. He believed that mental development was achieved through social interaction. Dewey saw children as participants in their learning rather than receivers of their learning. He established a Laboratory School at the University of Chicago where learners were encouraged to actively participate in collaborative learning activities such as students' building a playhouse to learn geometry and measurement principles. He believed that children should be active, participatory learners who collaborate with others to better understand meaningful situations. Piaget (1973), on the other hand, believed that understanding emanates from unearthing and that without it invention and creativity are eliminated and one is caught in only duplication. He also observed that children do not think with the same logic as grown-ups (Papert, 2001). He recognised that children were not "empty vessels" to be filled with knowledge, but active developers of it. Piaget saw children as active, participatory learners as they frequently generated and tested their understanding of the world. Lev Vygotsky stressed the influence of cultural and social interactions on mental development, mainly the interaction of children with other people (Rice & Wilson, 1999). Through the theoretical concept of the Zone of Proximal Development (ZPD), he emphasised that there is a difference in what a child can accomplish in isolation and what he or she can accomplish with assistance. Discovery learning is an inquiry-based, constructivist learning theory that takes place in problem-solving situations where the learner draws on his or her own past experience and existing knowledge to discover facts and relationships and new truths to be learned (Bruner, 1961). Models that are based upon discovery learning model include: guided discovery, problem-based learning, simulation-based learning, case-based learning, and incidental learning.

Main features: Bicknell-Holmes and Hoffman (2000) describe the three main features of discovery learning as exploring and problem solving (exploring and problem solving to create, integrate, and generalise knowledge); taking responsibility for learning (student driven, interest-based activities in which the student determines the sequence and frequency); and building new knowledge on existing knowledge (activities to encourage integration of new knowledge into the learner's existing knowledge

base). There are several features: (1) encourages active engagement; (2) promotes motivation; (3) promotes autonomy, responsibility, independence; (4) develops creativity and problem solving skills; and (5) tailors learning experiences. From Wikipedia discovery learning is described as:

“There are multiple essential components that are required for successful discovery-based learning which include the following: Teacher guidance where the emphasis is on building upon students’ reasoning and connecting to their experiences; Classroom culture where there is a shared sense of purpose between teacher and students, where open-mindedness and dialogue are encouraged; Students are encouraged to ask questions, inquire through exploration and collaborate with teacher and peers.”

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 7 (at this stage, interpreted by members of the working group only).

Table 7. Framework of skills related to discovery learning

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	?	?	?

Skills and competencies not covered: While this background learning theory could be used to frame the development of all skills, the approach to discovery learning would need to be carefully considered. Expecting discovery to result without appropriate and adequate support would be unlikely to enable an effective development of these skills; misunderstandings and misinterpretations could too easily develop and persist without sufficient monitoring and discussion.

Experiential learning

Source: The model of experiential learning was developed by Kolb (1984) and Rodgers (1969) and it was based on John Dewey’s concept of ‘Learning by Doing’. Rodgers emphasised the importance of experiential learning as knowledge application and not mental learning. He believed that experiential learning looked at the needs and wants of individuals and was connected to their change and growth. Kolb, on the other hand, came up with a four-stage cyclical process, namely: concrete experience, reflection, abstract conceptualisation and active experimentation.

Main features: Kolb (1984) described learning as a process of knowledge creation based on the alteration of experience and the understanding of experience together with its transformation. He further proposed six key features of experiential learning: being perceived best as a process, rather than outcomes; an unceasing process based on experience; need for conflict resolution between dialectically contrasting ways of adaptation to the world; an all-inclusive process of adaptation to the world; encompassing connections between the individual and the environment; and lastly involving creation of knowledge as a result of the relationship between social and personal knowledge. Experiential learning is the process of learning through experience, and is more specifically defined as “learning through reflection on doing” (Patrick, 2011, p.1003). A four-stage cyclical theory of learning, Kolb’s experiential learning theory is a holistic perspective that combines experience, perception, cognition, and behaviour. The four styles proposed are: (1) assimilators, who learn better when presented with sound logical theories to consider; (2) convergers, who learn better when provided with practical applications of concepts and theories; (3) accommodators, who learn better when provided with “hands-on” experiences; and (4) divergers, who learn better when allowed to observe and collect a wide range of information. Building upon earlier work by John Dewey and Kurt Levin, American educational theorist David A. Kolb believed “learning is the process whereby knowledge is created through the transformation of experience” (1984, p.38).

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support

the development of the skills in the framework are shown in Table 8 (at this stage, interpreted by members of the working group only).

Table 8. Framework of skills related to experiential learning

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	?	?

Skills and competencies not covered: While all skills could be developed using this background learning theory to frame learning activities, the basis of the learning theory and its focal elements were developed from adult investigations and approaches, rather than from exploration of developmental approaches at younger ages.

Problem-based learning

Source: Tse and Chan (2003) describe problem-based learning (PBL) as a learning approach where “the problem drives the learning”. The problem is given first and then the learners are presented with the opportunities to discover solutions to it (Salas, Segundo, Álvarez, Arellano & Pérez, 2014). The “learning is student-centered” (Tse & Chan, 2003) and the teacher acts as a facilitator whose role is to guide the “self-directed learners” (Forcael, Gonzalez, Orozco, Opazo, Suazo & Aranguiz, 2015; Stanford University Center for Teaching and Learning, 2001). Savery and Duffy (1995) assert that PBL first emerged in the field of medical education in the 1950s. This was as a result of increasing displeasure with the traditional medical education practice at McMaster University in Canada (Barrows, 1996). The PBL inventors critiqued traditional health science education due to its lecture approach, with more prominence put on memorisation of fragmented biomedical information, overlooking the importance of assisting learners cultivate the clinical problem-solving skills required for a lifetime of practice and learning (Barrows, 1996). Learners worked in small groups where they interacted with simulated patients with complex and meaningful medical problems. Using patient interviews, records, and selected laboratory results, they were able to identify learning issues and develop a diagnosis and treatment plan (Torp & Sage, 1998). The learning process was reinforced by teachers whose role was to facilitate discussion-based learning through questions, asking about and monitoring the problem-solving process (Hmelo, 1998). PBL is a constructionist method which allows students to learn about a subject by exposing them to multiple problems and asking them to construct their understanding of the subject through these problems. This kind of learning can be very effective; for example, in mathematics or computing classes, because students try to solve the problems in many different ways, stimulating their minds (Hmelo-Silver & Barrows, 2006). Many studies have reported the utilisation of PBL in teaching software engineering courses, especially in teaching Agile software development methods.

Main features: Popper (1994) believes that “Alles leben ist Problemlösen” (all life is problem solving); hence, life is full of learning opportunities. For PBL to take place, the following key features should be present: problem-focused (learning begins by addressing simulations of an authentic, ill-structured problem - what is to be learnt is built round a problem rather than creating a list of topics); student-centred (the teacher does not command the learning activities, but rather acts as a facilitator in the whole process); self-directed (students individually and collaboratively assume responsibility for generating learning issues and processes through self-assessment and peer assessment and access their own experiential knowledge and learning materials); self-reflective (learners monitor their understanding and learn to adjust strategies for learning); and facilitative (instructors are facilitators and not teachers). The following are some of the defining characteristics of PBL: (1) learning is driven by challenging, open-ended problems or tasks; (2) problems are context specific; (3) group work is common: learners work as self-directed, active investigators and problem-solvers in small collaborative groups; (4) a key problem is identified and a solution is agreed upon and implemented; (5) teachers adopt the role as facilitators of learning, guiding the learning process and promoting an environment of inquiry.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support

the development of the skills in the framework are shown in Table 9 (at this stage, interpreted by members of the working group only).

Table 9: Framework of skills related to problem-based learning

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	√	√	√

Skills and competencies not covered: This approach could enable the framing of development of all skills. Appropriateness to age would need to be considered, but, having said this, problem-based approaches are used in effective ways across the age span of compulsory and higher education.

Connectivism

Source: Connectivism is a digital age theory developed by George Siemens and Stephen Downes which came about as a result of the two criticising restrictions of behaviourism, cognitivism, and constructivism theories (Siemens & Downes, 2009). According to Siemens (2005), “Connectivism presents learning as a connection or network-forming process”. Siemens believes that learning today is too complex and we need to rely on a network of people and technology to store, access, and retrieve knowledge and motivate its use (Siemens, 2006). Learning is viewed as multi-faceted and particular tasks define which approach to learning is most appropriate to the learner (Siemens, 2003). Connectivism is a learning theory that explains how Internet technologies have created new opportunities for people to learn and share information across the World Wide Web and among themselves (Siemens, 2005).

Main features: According to Siemens (2004), key features of connectivism include: correct and current knowledge (the main purpose of all connectivism learning); decision-making (important to learning and is also a learning process); ability to establish associations among multiple fields, ideas, and concepts is essential; learning can exist in devices other than human beings (this includes networks, databases, communities); knowledge acquisition is a process of linking varied devices, knowledge bases, or accessing prevailing networks; knowledge acquisition relies on the varied thoughts and perspectives within a network; and being able to acquire more knowledge more vital than the already acquired knowledge. In addition to these, Siemens (n.d.) also affirms the following as key features of connectivism: the integration of thought and sentiments are vital to the creation of meaning; maintenance and nurturing network linkages in order to sustain continual learning; development and utilisation of knowledge is essential in knowledge acquisition; and lastly learning takes place in more than one way. A key feature of connectivism is that much learning can happen across peer networks that take place online. In connectivist learning, a teacher will guide students to information and answer key questions as needed, in order to support students’ learning and sharing on their own. Students are also encouraged to seek out information on their own online and express what they find. A connected community around this shared information often results. A lack of comparative literature reviews in connectivism papers complicates evaluating how connectivism relates to prior theories.

Match to the skills and competencies of a contemporary software developer: If a teacher used this learning theory as an underpinning base, learning activities that could be devised that would support the development of the skills in the framework are shown in Table 10 (at this stage, interpreted by members of the working group only).

Table 10: Framework of skills related to connectivism

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
All	?	?	?

Skills and competencies not covered: It is not clear that this learning theory would enable a framing of learning activities for use across the age range. Activities that would depend on uses and understanding of distributed resources and distributed knowledge might not easily support younger age groups.

Mapping of learning theory features for the framing of skills and competencies of a contemporary software developer

The mapping exercise shows that a number of background learning theories could be used to frame the development of learning activities in computing, computer science or informatics education. However, there are a number of critical concerns that are raised by this mapping exercise:

- If stages of cognitive development are used a frame for learning activity development, then computing, computer science or informatics learning will become decontextualised from a contemporary perspective. Certain skills will be emphasised earlier than others, and how these are then contextualised later will need to be a key concern.
- If social constructivist approaches are used as a background frame, then extents and forms of social intervention and social interaction need to be carefully considered, as they will influence outcomes and longer-term approaches to skills development and use.
- If social learning is used as a background frame, then the importance of developing and subsequently using metacognitive skills will need to be a focus for development from early ages.
- If constructionism is used as a background frame, then the presence and roles of social interaction will need to be as much a focus as material resource interaction, across ages, and in all forms of activity (with those concerned with technical skills just as much as those focusing on social and emotional skills).
- Situated learning could provide an ideal background frame, but then teachers and other supporters will need to be highly competent and confident, in creating and using a situated context for their learners.
- Discovery learning could form a background frame, but independent and unsupported discovery will not be likely to result in effective outcomes, considering the nature of the subject, and the misconceptions and misunderstandings that could result.
- Experiential learning could form a background frame, but its focus from investigating adult interactions means that it will need to be trialled and tested extensively enough across age ranges.
- Problem-based learning will appear to provide a useful background frame, and has been used across educational practice widely.
- Connectivism as a background frame is likely to be difficult to apply for younger age ranges if distributed resources and knowledge are involved in learning activities for those age groups.

Skills and competencies not covered by any learning theory

It is not possible to say that any of the skills could not be developed through learning activities framed by one or another learning theory. However, it is clear that background learning theories used to frame the development of learning activities can have a major effect, and could easily influence (positively or negatively) the forms of interaction and, hence, either enhance or limit developments of certain skills to lesser or greater extents. Table 11 shows, from the interpretations shown in previous tables, those skills that are highlighted as being most uncertain to be developed, considering how appropriate forms of learning activity might be created, taking cognitive constructivism, social constructivism, social learning, constructionism, situated learning, experiential learning, and problem-based learning theories as background frames into account.

Table 11: Framework of skills related to interpreted limitations of background learning theories

Elements or processes	Technical skills	Analytical skills	Social and emotional skills
Conceiving		??	???
Planning	?	??	???
Designing		??	???
Developing		??	???
Documenting		??	???
Debugging/testing	?	??	???
Deploying		??	???
Evaluating/improving	?	??	???
Maintaining	?	??	???

Rationale, proposed name and features of a new learning theory

Based on this analysis, whilst a completely new learning theory might not be shown as an absolute necessity at this time, it is highlighted at the same time that any background learning theory used as a frame needs to be considered adequately and appropriately in terms of how analytical and social and emotional skills particularly will be developed through appropriate learning activities. Similarly, how skills that might not be easy to develop at young ages are appropriately contextualised as early as possible, also will need careful consideration. Investigating stages of development for an updated learning theory approach, perhaps a learning theory that might be termed **social creative constructivism**, should be a focus of concern and action for future research.

References

- Barrows, H. S. (1996). Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning*, 68, 3-12.
- Berding, J. W. A. (2000). *John Dewey's participatory philosophy of education: Education, experience and curriculum* [Online]. Accessed from: <http://www.socsci.kun.nl/ped/whp/histeduc/misc/dewey01.html>.
- Berger, P. and Luckmann, T. (1991). *The social construction of reality*. London: Penguin Books.
- Bicknell-Holmes, T. and Hoffman, P. S. (2000). Elicit, engage, experience, explore: Discovery learning in library instruction. *Reference Services Review*, 28(4), 313-322.
- Bodrova, E. and Leong, D.J. (2012). Tools of the mind: Vygotskian approach to early childhood education. In: Rooparine, J. L. and Jones, J. (Eds.), *Approaches to Early Childhood Education* (6th ed.), 241-260.
- Brown J. S., Collins, J. and Duguid, P. (1989). Situated Cognition and the Culture of Learning. *Educational Researcher*, 18(1), 32-42.
- Bruckman, A. (2006). Learning in online communities. In: Saywer, K. (Ed.), *Cambridge handbook of the learning sciences*. New York, NY: Cambridge University Press, 461-472.
- Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review*.
- Chatti, M. A. (2010). The LaaN Theory. In: *Personalization in Technology Enhanced Learning: A Social Software Perspective*. Aachen, Germany: Shaker Verlag, 19-42. Accessible from: <http://mohamedaminechatti.blogspot.de/2013/01/the-laan-theory.html>
- Clancey, W.J. (1995). A tutorial on situated learning. In Self, J. (Ed.), *Proceedings of the International Conference on Computers and Education*. Charlottesville, VA: AACE, 49-70.
- David L. (2014). Problem-Based Learning (PBL). *Learning Theories*, July 23, 2014. Accessible from: <https://www.learning-theories.com/problem-based-learning-pbl.html>.

Department for Education (2013). National curriculum in England: computing programmes of study. Accessible from: <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>

Dewey, J. (1997). *Democracy and education*. New York, NY: Simon and Schuster. (Original work published 1916).

diSessa, A. A. and Cobb, P. (2004). Ontological innovation and the role of theory in design experiments. *Journal of the Learning Sciences*, 13(1), 77-103.

Duffy, T.M. et al. (Eds.) (1993). *Designing Environments for Constructive Learning*. Heidelberg, Germany: Springer-Verlag.

Edelman, G.M. (1992). *Bright Air, Brilliant Fire: On the Matter of the Mind*. New York, NY: Basic Books.

Forcael, E., González, V., Orozco, F., Opazo, A., Suazo, Á. and Aránguiz, P. (2015). Application of Problem-Based Learning to Teaching the Critical Path Method. *Engineering Education and Practice*, 141(3), 04014016. [http://dx.doi.org/10.1061/\(asce\)ei.1943-5541.0000236](http://dx.doi.org/10.1061/(asce)ei.1943-5541.0000236)

Gauvain, M. (2008). Vygotsky's sociocultural theory. *Encyclopedia of Infant and Early Childhood Development*, 3, 404-413

Glaserfeld, E. von (1988). The reluctance to change a way of thinking. *Irish Journal of Psychology*, 9(1), 83-90. Accessible from: <http://www.vonglasersfeld.com/111>

Gredler, M. (2008). Vygotsky's Cultural-Historical Theory of Development, In Sialkind, N.J. (Ed.), *Encyclopaedia of Educational Psychology*, 1, 1011-1014.

Herrington, J. and Oliver, R. (1995). *Critical characteristics of situated learning: Implications for the instructional design of multimedia*. In: ASCILITE 1995 Conference, University of Melbourne, Australia.

Hmelo, C. E. (1998). Problem-based learning: Effects on the early acquisition of cognitive skill in medicine. *Journal of the Learning Sciences*, 7, 173-208.

Hmelo-Silver, C. E. and Barrows, H. S. (2006). Goals and strategies of a problem-based learning facilitator. *Interdisciplinary Journal of Problem-based Learning*, 1, 21-39.

Kalpna, T. (2014). A Constructivist Perspective on Teaching and Learning: A Conceptual Framework. *International Research Journal of Social Sciences*, 3(1), 27-29.

Kirschner, P. A., Sweller, J. and Clark, R. E. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2), 75–86. doi:10.1207/s15326985ep4102_1

Kolb, D. A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Kop, R. and Hill, A. (2008). Connectivism: Learning theory of the future or vestige of the past? *The International Review of Research in Open and Distance Learning*, 9(3).

Lamb, S., Maire, Q. and Doecke, E. (2017). *Key Skills for the 21st Century: an evidence-based review*. Melbourne, VIC: Victoria University, Australia. Accessible from: <https://education.nsw.gov.au/our-priorities/innovate-for-the-future/education-for-a-changing-world/research-findings/future-frontiers-analytical-report-key-skills-for-the-21st-century/Key-Skills-for-the-21st-Century-Analytical-Report.pdf>

Lave, J. (1988). *Cognition in Practice*. Cambridge: Cambridge University Press.

Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.

Martinez, S.L. and Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom*. Constructing Modern Knowledge Press.

McKinley, J. (2015). Critical Argument and Writer Identity: Social Constructivism as a Theoretical Framework for EFL Academic Writing. *Critical Inquiry in Language Studies*, 12(3), 184-207.

- Miniflash (n.d.). *4 essential elements of social learning*. Accessible from: <https://www.mindflash.com/blog/4-essential-elements-of-social-learning>
- National Careers Service (2018). *Software developer: Programmer*. Accessible from: <https://nationalcareersservice.direct.gov.uk/job-profiles/software-developer#what-youll-do>
- Noss, R. and Clayson, J. (2015). Reconstructing Constructionism. *Constructivist Foundations*, 10(3), 285-288.
- Papert, S. (1986). *Constructionism: A new opportunity for elementary science education*. Massachusetts Institute of Technology, Media Laboratory, Epistemology and learning group: National Science Foundation.
- Papert, S. (2001). *Jean Piaget. Time* [Online]. Accessible from: <http://www.time.com/time/time100/scientist/profile/piaget.html>
- Papert, S. and Harel, I. (1991). *Constructionism*. New York, NY: Ablex Publishing Corporation.
- Passey, D. (2013). *Inclusive technology enhanced learning: Overcoming Cognitive, Physical, Emotional and Geographic Challenges*. New York, NY: Routledge.
- Patrick, F. (2011). *Handbook of Research on Improving Learning and Motivation*.
- Piaget, J. (1936). *Origins of intelligence in the child*. London: Routledge & Kegan Paul.
- Piaget, J. (1958). The growth of logical thinking from childhood to adolescence. *AMC*, 10, 12.
- Piaget, J. (1973). *To understand is to invent*. New York, NY: Grossman.
- Piaget, J. and Cook, M. T. (1952). *The origins of intelligence in children*. New York, NY: International University Press.
- Popper, K. (1994). *Alles leben ist problemlösen*. Munich, Germany: Piper Verlag.
- Rice, M. L. and Wilson, E. K. (1999). How technology aids constructivism in the social studies classroom. *Social Studies*, 90(1), 28-33.
- Rogers, C.R. (1969). *Freedom to Learn: a View of What Education Might Become*. Columbus, OH: Charles E. Merrill.
- Salas, J., Segundo, J., Álvarez, C., Arellano, J. and Pérez, A. (2014). Evaluation of the Use of Two Teaching Techniques in Engineering. *International Journal of Engineering Pedagogy*, 4(3), 4. <http://dx.doi.org/10.3991/ijep.v4i3.3287>
- Savery, J. R. and Duffy, T. M. (1995). Problem based learning: An instructional model and its constructivist framework. *Educational Technology*, 35(5), 31-38.
- Siemens, G. (2003). *Learning Ecology, Communities, and Networks: Extending the Classroom*. Accessible from: http://www.elearnspace.org/Articles/learning_communities.htm
- Siemens, G. (2004). *Connectivism: A Learning Theory for the Digital Age*. Accessible from: <http://www.elearnspace.org/Articles/connectivism.htm>
- Siemens, G. (2005). Connectivism: A learning theory for the digital age. *International Journal of Instructional Technology and Distance Learning*, 2(1), 3-10.
- Siemens, G. (2005). *Connectivism: Learning as Network Creation*. e-Learning Space.org website. Accessible from: <http://www.elearnspace.org/Articles/networks.htm>
- Siemens, G. (2006). *Connectivism: Learning theory or pastime of the selfamused?* Elearnspace blog. Accessible from: http://www.elearnspace.org/Articles/connectivism_selfamused.htm
- Siemens, G. (n.d.). *About: description of connectivism*. Accessible from: <http://www.connectivism.ca/about.html>
- Siemens, G. and Downs, S. (2009). *elearnspace*. Accessible from: <http://www.elearnspace.org/blog/>

SkillScan (2012). *Chart of Skill Categories, Skill Sets and Sample Career Options*. Accessible from: <https://www.skillskan.com/sites/default/files/chart-of-skill-sets.pdf>

Stanford University Center for Teaching and Learning. (2001). Problem-Based Learning. *Speaking Of Teaching*, 11(1), 1-8. Accessible from: [http://web.stanford.edu/dept/CTL/ Newsletter/](http://web.stanford.edu/dept/CTL/Newsletter/)

Torp, L. and Sage, S. (1998). *Problems as possibilities: Problem-based learning for K-12 education*. Alexandria, VA: ASCD.

Tse, W. and Chan, W. (2003). Application of Problem-Based Learning in an Engineering Course. *International Journal of Engineering Education*, 19(5), 747-753. Accessible from: <https://www.ijee.ie/articles/Vol19-5/IJEE1440.pdf>

Vincini, P. (2003). The nature of situated learning. *Innovations in Learning*, 1-4.

Vossoughi, S., Escudé, M., Kong, F. and Hooper P. (2016). Tinkering, Learning and Equity in the After-School Setting.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Wynn, E. (1991). Taking Practice Seriously. In J. Greenbaum and M. Kyng (Eds.), *Design at Work: Cooperative design of computer systems*. 45-64.

WG3: Creating and Looking at Art with Logo Eyes

Evgenia Sendova, *jenny.sendova@gmail.com*

Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences, Bulgaria

Christos Chytas, *christos.chytas@uni-oldenburg.de*

Computing Education Research Group, University of Oldenburg, Germany

Katarzyna Olędzka, *katarzyna.oledzka@oeiizk.waw.pl*

Computer Assisted Education and Information Technology Centre in Warsaw, Poland

Ralf Romeike, *ralf.romeike@fau.de*

University of Erlangen-Nuremberg, Germany

Wolfgang Slany, *wolfgang.slany@tugraz.at*

Catrobat Team, Graz University of Technology, Austria

Abstract

Computation is an integral component in nearly every dimension of our lives and *provides new tools for self-expression even in areas that one would not dare to enter without the access to computers*. In 1967, Seymour Papert, Wally Feurzeig and Cynthia Solomon designed Logo to allow children and youth to explore programming and mathematics concepts, while providing exciting opportunities for creativity and experimentation. Computation provides an appropriate framework for dealing with notions of *how to* and *what if* rather than with notions of *what is*, which already has an impact on the way we teach fine arts, music, or literature. For Papert and his fellow constructionists the programming was not only a technical skill for industry but also a mode of creative and personal expression. Today, the absence of computation in mainstream art and science education is still apparent but the appearance of Logo and new modern tools that inherit the ideas of the Logo educational philosophy (known as *constructionism* today) promises to change that. Drawing on examples from experiences with IT teachers and children in participatory design courses, we discuss *how to harness the Logo culture of making things happen, making things work, so as to open the eyes of young people to the beauty around them*.

Keywords

art; creativity; coding; logo; visual programming; maths; stitching; embroidery

Prologue (in which the Turtle makes a point of the computer art)

Achilles and the Turtle (Fig. 1), characters we know not only from the famous Zenon paradox but also from the essay of Luis Carol *What the Tortoise said to Achilles* [1] and the book *Gödel, Escher, Bach* by Douglas Hofstadter [2], in which the author promotes the idea that in the history of human civilization the division between science and art is in fact superficial.



Figure 1. Achilles and the Turtle discussing the modern and the artificial art (cartoonist Yovko Kolarov)

Turtle: About 60 years ago I attended an exhibition where there were only black paintings looking practically identical. The artist would address all the questions with a mysterious smile: Looking at art is

not as simple as it looks... *But then would continue with a serious tone:* Art is too serious to be taken seriously [3].

Achilles: *I know what you are talking about – in a recent exhibition I attended (Fig. 2) the largest paintings in the modern art wing appeared to be solid black; but with sustained looking, shapes started to emerge. I wonder what art critics would say if such paintings were generated by a computer.*



Figure 2. An exhibition of Mark Rothko in MFA, Boston (2018).

Turtle: *True! But the attempts to model abstract art do not intend to challenge the art critics. Maybe it is for the first time today that the computer provides people with environments where they could experimentally verify ideas, formulated by eminent representatives of the abstract art such as Kandinsky and Mondrian. You might have heard that Mozart has composed music by combining randomly ready-made 2-bar musical fragments. Randomness can also be seen in art, e.g. Hans Arp has thrown color pieces of paper by leaving to the destiny to combine them. It is known that the only rule the representatives of Dadaism followed was randomness.*

Achilles: *You mean that the abstractionists do not accept any rules?* **Turtle:** *On the contrary. In his book On the Spiritual in Art the founder of the abstract art, Kandinsky, writes: We are approaching the time when a conscious, reasoned system of composition will be possible, when the painter will be proud to be able to explain his works in constructional terms.*

Achilles: *Sounds like someone who would have found like-minded people among the constructionists...*

Turtle: *In the field of the so called Artificial Art there are programs which can generate paintings on a specific topic. An interesting example is Aaron, developed by Harold Cohen, (Fig. 3), which generates figures resembling people dancing or playing with a ball in an environment looking like a forest, rocky hills or beach. Due to the random choice of specific variables Aaron always generates different paintings, obeying a specific style.*



Figure 3. Detail from an untitled AARON drawing, 1980 (left); AARON image created at the Computer Museum, Boston, MA, 1995 (right) [4].

Achilles: *I would rather think that it is more natural for an intelligent computer to draw a self-portrait, i.e. a computer which draws a portrait of itself, which draws a portrait of itself, etc.*

Turtle: *I know your passion about recursion. But you will see in the works of our authors and their students that the computer could be used not only to play the role of an amateur-abstractionist, but also as an environment for artistic experiments. In such an environment (Logo or Logo-like) one could comment on what makes the abstract art in fact art and not just random collections of figures spread on a canvas. Besides, using a language-based computer environment it is easier to give working definitions of abstract notions, such as harmony of colors, balance, tension, calmness, and verify experimentally the validity of these definitions. Besides, computation provides new tools for self-expression even in areas that one would not dare to enter without the access to computers.*

Achilles: *They say that a picture is worth more than 1000 words. (Even if it is a computer work). Thus – FORWARD to the paintings!*

Turtle: *Although I am just a little ahead of you, I don't think you have chances of reaching me (and my relatives for that matter) before infinity. My recently born cousin, the Logo turtle, is already a well-known artist. Let us see what the authors of this article have to say about how they and their students look at art with Logo eyes...*

The Logo turtle as an eye-opener and as a tool for enriching the artists' palette

Art or a Logo exercise?

The first two pictures in Fig. 4 are the simplest version of abstract art which could be modeled by means of the basic primitives of the Logo turtle graphics. Thus, they could be included as an exercise not only to implement students' skills in controlling the turtle but also to open their eyes for the art of Max Bill and his love of geometry [5].

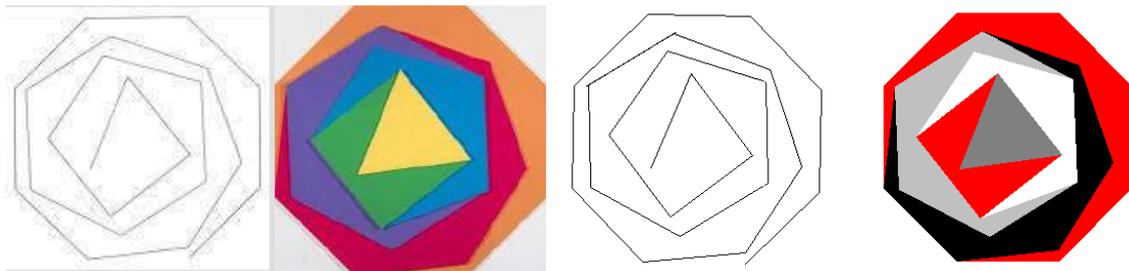


Figure 4. Max Bill (lithographs from 1938) and their Logo variations.

An episode from a PD course for IT teachers

During a PD course for teachers in informatics at Sofia University a shy young woman handed to the lecturer a piece of an old calendar with a painting by Vasarely (Fig. 1, leftmost) and ask her if it could be modeled by means of the Logo turtle [6].

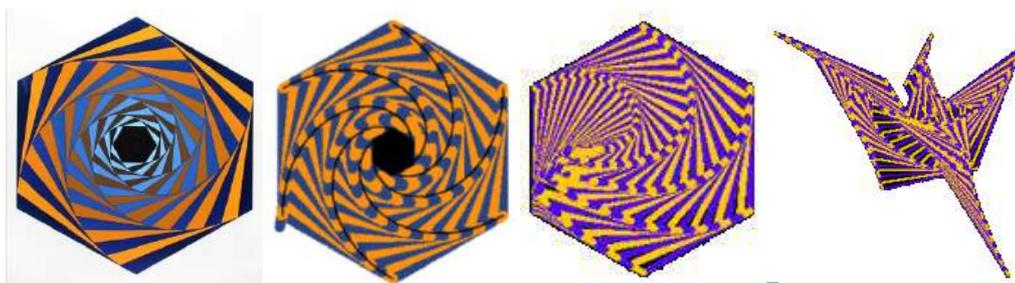


Figure 5. Vasarely's Los Angeles and Logo variations on it.

Of course, there are different ways to create computer models of *Los Angeles*. After a short discussion they figured out that the *Four-bug problem* could be used (modified for 6 turtles in this case) by connecting the positions of the turtles at regular intervals of time alternating the color of the trace. From there on variants by modifying the parameters in the initial setting of the problem (e.g. the strength of the attraction, the number and the positions of the turtles) were created. A sufficient number of experiments produced realistic images (Fig. 5, rightmost). And variations on a theme are in fact *the crux of creativity*. Furthermore, *one can come to an understanding* of such fundamental cognitive processes as *pattern perception, extrapolation and generalization* **only** by modeling them in the *most carefully designed and restricted microdomain* [7].

Episodes from pre-service course for math and CS teachers

Ideas of identifying what is essential in a specific painting, in the author's style, or even – of a period in art, could be found in the wonderful book of James Clayson [8]. Good practices in the context of visual modeling with students of different age and cultural background demonstrate the applicability of Clayson's ideas. An inspirational synergy among mathematics, informatics and art has been achieved within the pre-service and in-service teachers in mathematics and informatics at Sofia University. The first stage of the modeling process was to identify fundamental elements in paintings characterized by their vibrant colors and sharply patterned geometric forms (e.g. de Still group, Kandinsky, Sonia Delaunay, Escher) and then to create stylized versions of those by Logo procedures. The next stage was to understand the *balance* (Mondrian) and the *forces-tensions living in the painting* (Kandinsky). The students created *working definitions* of balance according to different criteria (e.g. determining the weight of an element by its size, complexity, color) and tested them in a number of iterations, concentrating on the phenomenon to be modeled. The students realized that the abstract paintings contain a harmony whose laws could be revealed and their hypotheses about those laws could be materialised in computational form to be verified experimentally. Here are some examples.

Modeling circles a la Kandinsky



Figure 6. *Deepened Impulse* (left); *Circles in a Circle* (right).

Various ideas of modeling the circles include using sequences of nuances to create the illusion of 3D objects, to create a *balance* among their locations, to animate them (make them pulsate or move according to different rules).

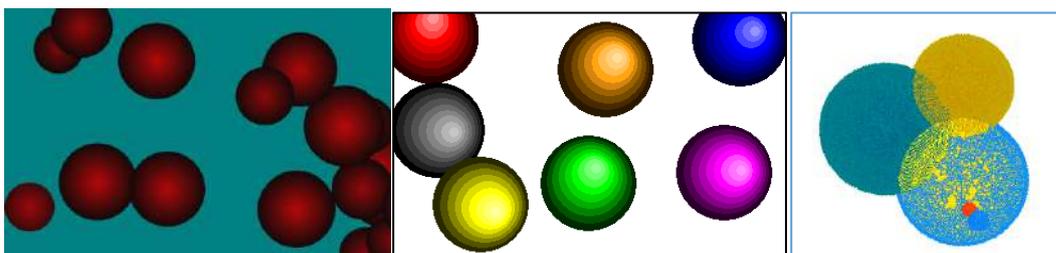


Figure 7. *Creating visual effects by means of Comenius Logo.*

As seen from these pictures the work on specific elements of a painting being modeled requires experiments of various ideas and techniques both from informatics and artistic point of view.

Variations on Vega by Vasarely

For modeling Vega (Fig. 8, leftmost) we used Lissajous curves describing a complex harmonic motion. The graphs are the traces of a turtle whose Cartesian coordinates are obtained by the positions of two other turtles moving along specific intervals on the coordinate axes of rectangular coordinate system.

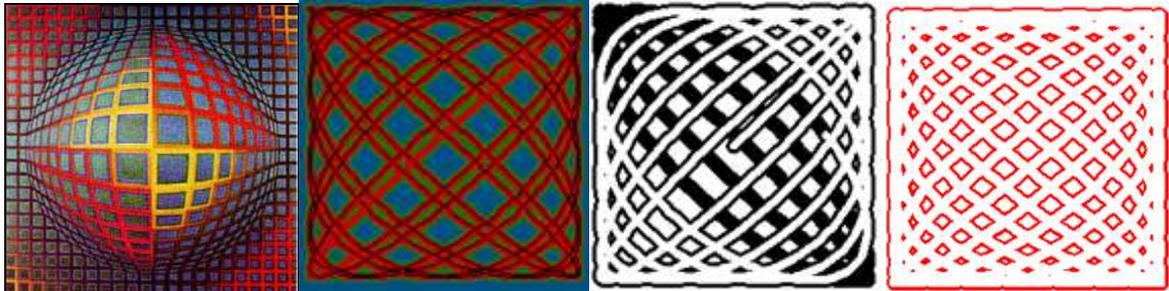


Figure 8. Vasarely Vega (1957) and Comenius Logo variations.

Such curves could be described also by a single turtle by means of trigonometric functions. The reason of using three turtles (apart from using a more modest mathematical apparatus) is to generalize the motion of two of the turtles by driving them along polygon lines while the position of the third is always in the middle between the first two (Fig. 9).

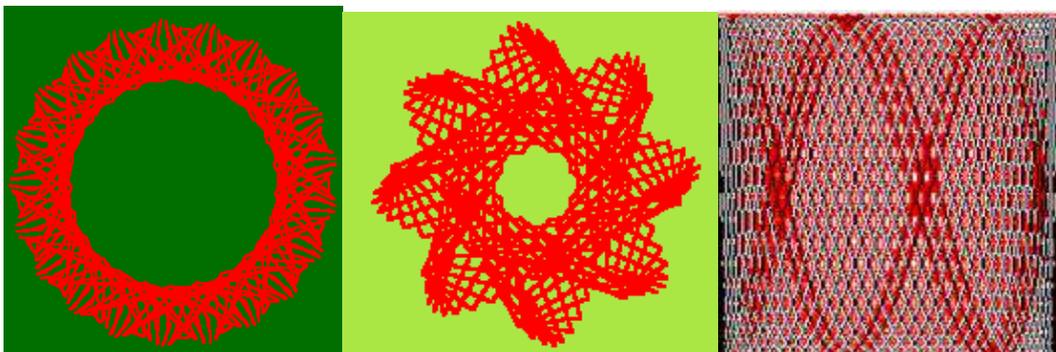


Figure 9. Exploring further the effect of modifying the Logo models of Vega.

In our sessions we encourage students and teachers to not reduce their visual models to imitations only. Instead, they are encouraged to use their own creativity and experiment with their own design ideas.

This process contains (to use the phrase of Hofstadter) *unpredictable predictability*. In other works we can expect the appearance of new stylistic attributes depending on the rank in which we vary the random variables. After leaving the frames of the strict imitation we could be inspired by new combination of forms and colors and get new insight leading to a new formalization – a process supported by the enormous potential of the computer to make various combinations in short time and by our human potential to interpret new abstract structures [9].

Recent programs in artificial art that can tweak photos to mimic the style of famous painters are already widely available. There are even apps that do this, such as DeepArt [10]. But even more ambitious is a new system producing images in unconventional styles from scratch [11].

Episodes from our work with children

Looking with Logo eyes around us

Geometric shapes and regular patterns can be found practically everywhere, although they are often unnoticed by an untrained eye. For example concepts like the *golden ratio*, *Fibonacci sequence*, *symmetry* can be found in nature (from the number and configuration of petals in flowers to the spiral patterns of shells). Such concepts have intrigued the interest of artists and scientists for centuries and computation provides the means to express such concepts faster and more efficient than ever. Logo is a language suitable for dealing with phenomena which we could better understand by modeling them ... What is valuable in programs in a microworld tuned to a particular domain (art in this case) is that they could be used as glass-boxes. During online courses for primary school children [12] we start from simple drawings to move to more complicated ones. By solving several tasks students build their own knowledge. They combine finding recurrent elements with the ability to describe pictures by Logo procedures. Moreover, children improve their understating of geometric shapes.

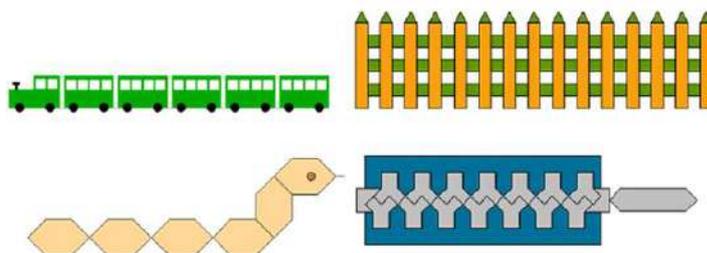


Figure 10. Tasks for finding repetition from miniLogia competition.

They need to integrate art, mathematics and computational thinking. Students learn how to efficiently use the most important procedures of turtle graphics, apply iteration and recurrence and break down a problem into sub-problems. Motives are made by simple and complicated structures, there are rhythms in different dimensions, sometimes balance or even the tension. Mathematical knowledge is needed to solve the problems: square and its diagonal, proportions, regular polygons like triangle, square, hexagon, angles: right angle, straight angle, angle of 360° , partition into equal parts.

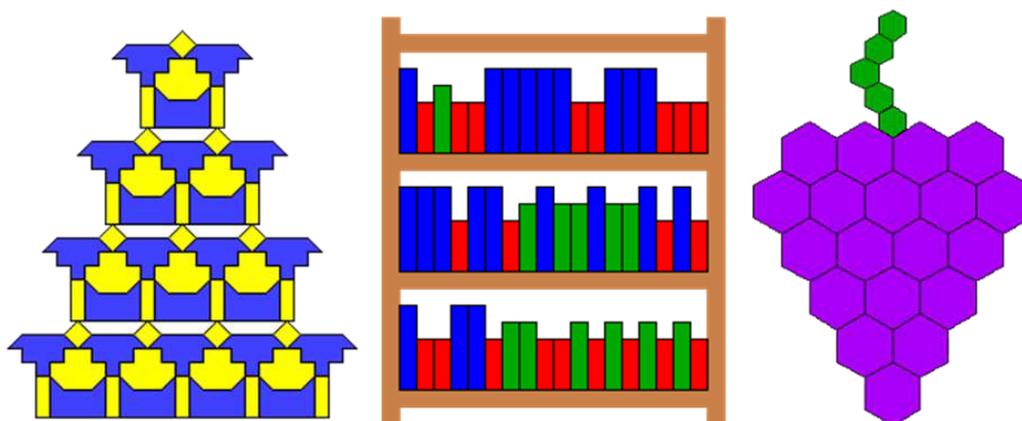


Figure 11. The more complicated tasks from miniLogia competition.

The older the students are, the more advanced pictures they can draw. During the miniLogia competition for primary school children [13] or the Logia competition for *gymnasium* students, [14] participants solve different problems inspired by everyday life (Fig. 10-11).

Logo eyes help us understand better the world

Using different geometric shapes, we can build some pictures that are tricky. Sometimes our perception of objects is inadequate to their physical properties. Optical illusion can be explained from physiological point of view, but also can be examined by drawing.

The line segment forming the shaft of the arrow with two tails is perceived to be longer than that forming the shaft of the arrow with two heads. When one compares the code for a turtle, we can realize that these two segments are equal length (line 3 and 13).

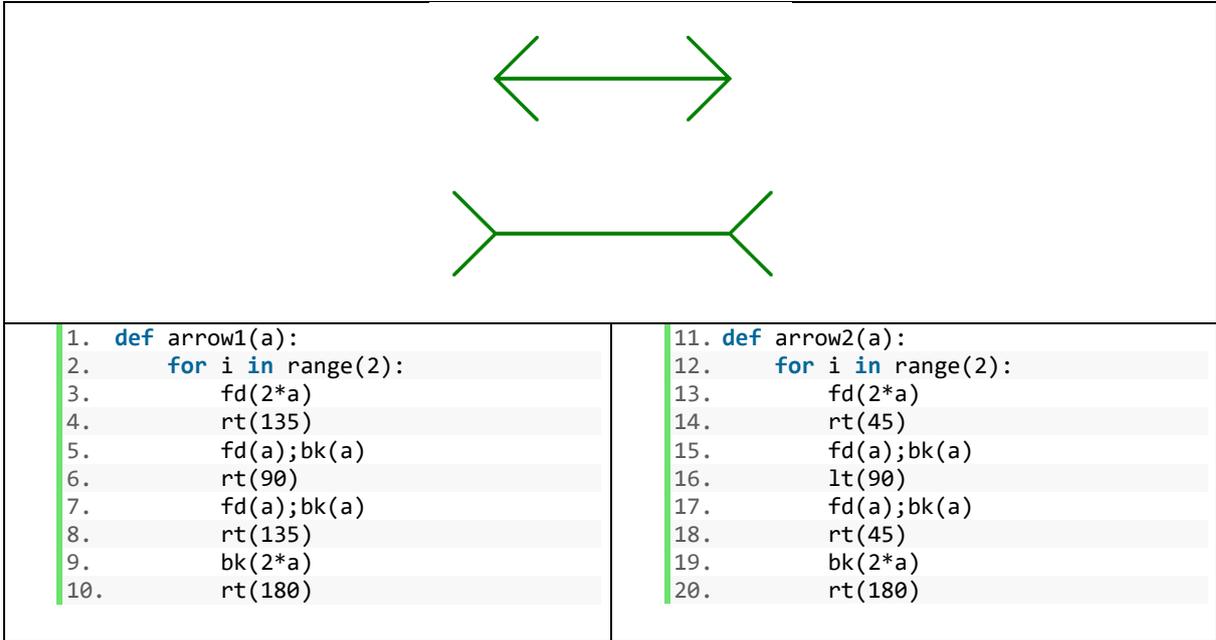


Figure 12. Two arrows and the code in Python with module turtle.

In another example two circles of identical size are placed near to each other and one is surrounded by small circles while the other- by a big one. The first central circle appears to be bigger than the second one. While comparing the code in Python: the central circles are identical (line 4 and 11), but the difference effect is due to the difference in the external circles (lines 6-8 and 13-15).

```

1. def jump(a):
2.     pu();fd(a);pd()
3. def big_circles(d):
4.     dot(d)
5.     for i in range(6):
6.         jump(d)
7.         dot(d/2)
8.         jump(-d)
9.         rt(60)
10. def small_circles(d):
11.     dot(d)
12.     for i in range(6):
13.         jump(4/2*d)
14.         dot(3/2*d)
15.         jump(-4/2*d)
16.         rt(60)
                
```

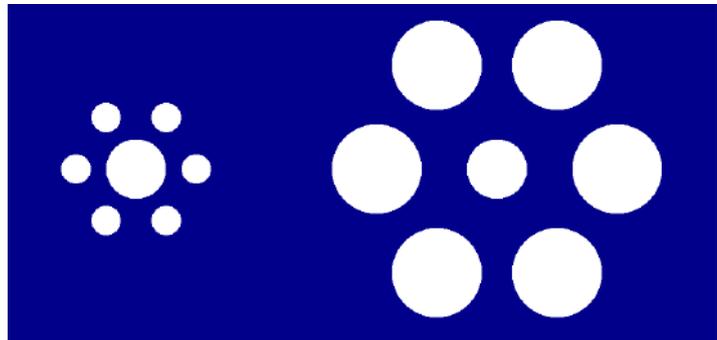


Figure 13. The Ebbinghaus illusion.

In the third example, there is a grid built by squares. On both pictures the squares are identical. While on the left they are forming straight columns, on the right – they are shifted. This creates the illusion that the horizontal lines are not parallel.

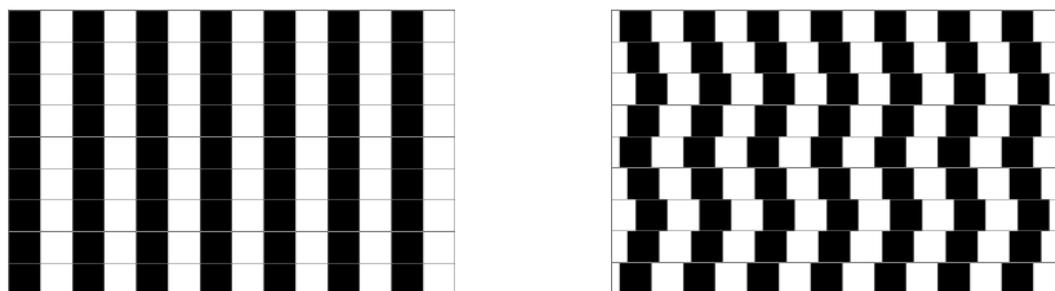


Figure 14. Grid built by squares

Playing with optical illusions made by a turtle, can lead to important discoveries. It helps the students to understand how human brain can be deceived and how to cope with situations when the perception is different from the reality. There are artists (e.g. Maurits Cornelis Escher) who use it for entertainment and fascination. For children, it is a field to meet challenges and be stimulated to develop their own creativity.

Beyond Logo

In the early 80s, when personal computers started infiltrating mainstream education, Papert published *Mindstorms* [15], a book on the potential of computers in children's and youth's education. In his inspiring book, Papert introduces the idea of a *Mathland* as a place where learning mathematics would be the equivalent of people learning their native language as when they were children. Logo (with its educational philosophy known as *constructionism* today) consists of both an environment and a tool to promote such types of learning. It allows children and youth to participate actively in creating and understanding more complex concepts of mathematics and computation in more creative ways than the ones provided in mainstream education in the past. Furthermore, innovations in educational technology which inherit the ideas behind Logo provide accessible tools that can support the creation of sophisticated projects through algorithmic or generative design and digital fabrication. With turtle graphics or block-based commands children and youth can use powerful computational tools to create projects ranging from small programs to physical artefacts (e.g. 3D printed models). The creation of physical artefacts with computational means can open new ways for creating and looking at art through tangible interactions, where the artistic creations become "objects-to-think-with" [15]. The term "*begreifbarkeit*" as coined by Katterfeldt et al. in [16] describes an element in a situation where the learners can grasp something physically (grab something tangible) but also grasp something as understanding it (in our case computational models for art). A good example of "graspable" art in the service of mathematics and computing education is given in the work of [17]. In a school setting, the students explored computational and mathematics concepts under the lens of 3D printing as a medium for creativity. They used turtle graphics in Beetle blocks [18] to create artistic artefacts that worked as "objects-to-think-with". However, even though the results of the course were promising, the combination of mathematics and programming concepts was a great challenge for the learners. Mathematics and computing science as subjects are considered difficult and not creative by many students. Therefore, we need to provide the right conditions and challenges of computational design considering the diverse experience, knowledge and interests of students. The intersection of mathematics, computational design and art might be an efficient way to highlight the beauty of science through the creation of personally meaningful artefacts.

Catrobat and TurtleStich

Catrobat is a free open source non-profit project that allows users to create and publish their own games and interactive animations directly from their smartphones, without the need for any PC, laptop, Chromebook, or tablet. Initiated in 2010, Catrobat as of July 2018 has more than 700,000 users from 180 countries, is available in 50+ languages, and has been developed so far by almost 1,000 volunteers from around the world. Catrobat's apps and services have been immensely influenced by MIT's Scratch project, and Catrobat is considered to be Scratch's little sister project for smartphones. Indeed,

Catrobat's apps allow to import Scratch projects, thus giving access to more than 37 million projects on phones as of July 2018. Catrobat's goal is to provide computational thinking skills for all worldwide teens, especially teens from less developed areas where other computational devices such as PCs are almost non-existent. Catrobat is a visual programming language like Scratch or Snap! and, like them, includes powerful Turtle graphics pioneered by Logo. Catrobat also allows to take advantage of the many sensors and high-resolution multi-touch screens of modern smartphones, allow to project one's smartphone's screen to arbitrarily large screens, some of which being multi-touch enabled with a back-channel to the phone's screen, thus allowing to serve as large interactive boards for games or large-scale art installations. Catrobat's apps also allow to seamlessly interact wirelessly with a plethora of external devices such as Arduino and Raspberry Pi boards that are frequently used by artists in interactive installations. Using Catrobat, creating the software for these interactive art installations is so straightforward that regular visitors to an art exhibition can be empowered to not only control these installations but also change or even create completely new objects of art in a few minutes without any previous knowledge of coding, on their own smartphones.

Below find a few examples from the many interactive animations that have been created using Catrobat. From left the right, "Psychedelic"⁸⁰ allows to tap the screen anywhere to place arbitrary colored and sized off-centered turning discs; "Japanese Rock Garden" allows to use a virtual rake with one's finger to create patterns in the pebbles as known from traditional Japanese rock gardens such as in the world-famous Ryōan-ji Zen Temple in Kyoto, where the creation of these patterns is used in traditional meditation by Buddhist monks; "Ten Fingers Painting" which allows to draw with all ten fingers at the same time with discs that constantly change their colors while one paints – all these interactive animations have an almost hypnotic impact on the users and in particular are endlessly fascinating for small children.

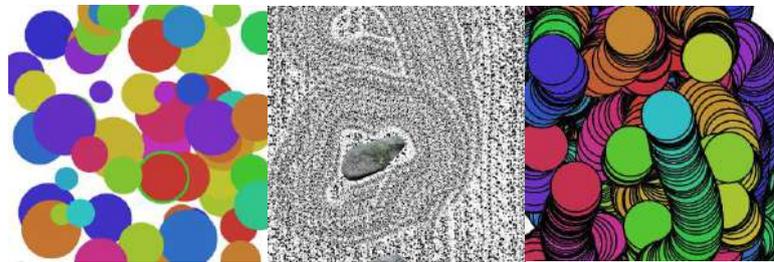


Figure 15. Interactive animations with Catrobat.

In a newly funded multi-year project, the Catrobat team has started to implement a TurtleStitch extension inspired by the TurtleStitch.org project based on Snap!. Catrobat's TurtleStitch extension allows to transfer the patterns users create on their phones to a programmable embroidery machine. Using the concepts of Logo's turtle graphics, users can create arbitrary patterns, including interactively generated ones using live user input. These patterns can then be stitched on clothes, bags, etc. Programmable stitching machines are increasingly becoming available in Fablabs, and recently have also become available in selected shops from Ikea and Muji. The prices of these machines range from 600 Euro for the smallest devices for private uses, up to hundreds of thousands of Euro for very large industrial machines. A pattern created with a Catrobat program called *Dream Catcher* together with parts of the corresponding Catrobat code on the left is shown in Fig. 16.

⁸⁰ All mentioned Catrobat projects can be found by searching for their names in the "Explore" section of Catrobat's coding apps, which can be downloaded from, e.g., Google Play via <https://catrob.at/gp> for Android smartphones (as of July 2018, the Catrobat app for iPhones is still a closed beta on iTunes).

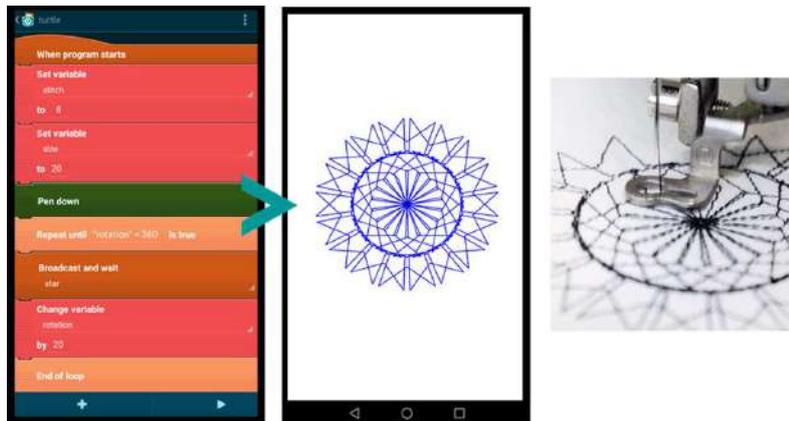


Figure 16. The "Dream Catcher" pattern.

On the right there is a photo of a stitching machine in the process of applying the pattern to a T-shirt (the photo is used with kind permission from Andrea Mayr-Stalder from the TurtleStitch.org project, since Catrobat's stitching extension is still under development).

Logo-like constructions from everyday life objects



Figure 17. Adam Hillman (Witenry) is truly one of the most talented artists on Instagram (<https://www.instagram.com/witenry/>)



Figure 18. Kasey Lund – physics professor, using a ruler (left);
A 3D bugs variation by George Karageorgiev, software developer

Conclusion

Thus, the study of the structure of knowledge and creative processes from procedural point of view has emerged as opposed to a more declarative point of view typical for the traditional education. Computation provides an appropriate framework for dealing with notions of *how to* and *what if* rather than with notions of *what is*, which already has an impact on the way we teach art, fine arts, music or literature. Both fine art and informatics have a specialised vocabulary of notions and use a specific syntax. The abstract art speaks rather of things *to be seen*. Similarly, Logo is a language suitable for dealing with phenomena which we could better understand by modeling them ...

What is valuable in programs in a microworld tuned to a particular domain (art in this case) is that they could be used as glass-boxes. In addition to provide an accessible enough formalised description of the process of generation, one can play with them long enough to check their flexibility. It is in this sense that we talk about *Logo as editor of ideas* (the editing process including corrections, modifications, refinement and enrichment of the program).

And now, it is time to give the floor again to Achilles and the Turtle:

Epilogue (in which Achilles and the Turtle reflect: what is the point?)



Achilles: *So, we are not arguing anymore if the computer can think but rather if what it creates with our help is in fact art...*

Turtle: *What the computer generates could not be called art in the usual meaning of it, but when we say this, we should be clear why not... What has been formalised in terms of Logo procedures is in fact a materialised conjecture about patterns noticed and reflected in a model of specific art works. When experimenting with such a model we could add or reject some rules based on the results and thus achieve a certain balance between the expected and unexpected.*

Achilles: *Would you please remind me what is a model in the context of creativity?*

Turtle: *The notion of model used here is in harmony with the definition of Marvin Minsky: To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A.*

Achilles: *Entering the shoes of B, I can accept that programs generating works of fine art are models (A*) – while playing with them I started noticing things that I had not seen before, while looking at the original paintings (A). Not to mention that while at school I hated the grammar, the music theory and the critical analysis of poetry or fine art. I thought that I could enjoy the art without being an artist...*

Turtle: *The works presented by the authors of this article illustrate the goal behind the computer generated art. Starting with relatively simple models, one can tackle much more complex ones based on what one has learned. Thus we do not only learn to model but model to learn (to paraphrase my spiritual father, Seymour Papert)...*

Achilles: *Can we look at the computer microworlds in a Logo sense as a means for exploring and editing ideas?*

Turtle: *Not only – the abstract sciences and the art alike work with imagination. Thus all the computer environments providing means for exchanging and sharing ideas are enriching these fields.*

Achilles: *Interesting. Some poets think that the computers are drying the imagination of children. And you are saying that computers help us seeing and even sharing our imagination.*

Turtle: *If we paraphrase Bernard Shaw: some look at abstract paintings and ask “why?”. Others generate abstract art by computer and ask “why not?”.*

Achilles: Another interesting thing in the context of visual modelling is that the shortcomings in a model are not fatal. On the one hand, we could easily figure them out provided that we sufficiently carry out many experiments. On the other hand they could provide us some interesting artistic ideas. By the way, I remember somebody saying that the debugging is the most powerful educational idea of 20th century! But going back to the point of programming in the field of art, I believe that the goal is not to replace the artists but rather to give more people the chance of understanding what could be formalised...

Turtle: And to enter in spheres which they otherwise would touch only as passive consumers. The enigma of the creativity will be left to the professional artists; they would start from a higher level, the one requiring an insight... It is in fact the creativity which is the common characteristics of the science and art.

Achilles: But do the computer works have the chance of being taken seriously?

Turtle: The main reason for computer-generated art to be treated as normal is that the culture of 20th century broadened the spectrum of what is considered real art.

Achilles: So, you mean that a robot could not pretend to be a man, but if it is present at a Masquerade ball it would be taken for a man who pretends to be a robot...

Turtle: Yes, similarly some computer-generated works could pass for real art, especially if we don't know that their author is computer. This is the well-known *Elisa effect*.

Achilles: Does the opposite also hold – to look for shortcomings of art works just because we think they are generated by a computer? I believe that people are prejudiced in both directions.

Turtle: **The real challenge from the point of view of AI is to create** a model, which not only does things but also knows that it does, i.e. it is aware of having its own style which could be developed. This, by the way, holds true for creativity in general – why do we consider an art work to be great is not trivial. But **even if** we would model art works by means of computer just to distinguish between the great art and the imitation, we would make a point...

References

- [1] Carroll, L. (2018). What the tortoise said to Achilles. In *Thinking about Logic* (pp. 3-7). Routledge.
- [2] Hofstadter, D. R. (1980). *Gödel, Escher, Bach* (pp. 137-138). New York: Vintage Books.
- [3] Barcio, P. (2018, July 26). How AD Reinhardt taught us to look at art, from <http://www.ideelart.com/module/csblog/post/269-1-ad-reinhardt.html>.
- [4] Cohen H. and AARON (2018, July 26). A 40-Year Collaboration, from <http://www.computerhistory.org/atcm/harold-cohen-and-aaron-a-40-year-collaboration/>
- [5] Fleischmann G. (2018, July 26). Max Bill and his love for geometry, from <https://www.mitpressjournals.org/doi/pdf/10.1162/07479360152383787>.
- [6] Sendova, E. (2001). *Modelling creative processes in abstract art and music*. In G. Futschek (Ed.) EUROLOGO 2001 A Turtle Odyssey, Proceedings of the 8th European Logo Conference 21-25 August, Linz, Austria
- [7] Hofstadter, D. R. (1981). Metamagical themas. *Scientific American*, 244(1), 22-16.
- [8] Clayson, J. (1988). *Visual modeling with Logo*. MIT press.
- [9] Sendova*, E., & Grkovska, S. (2005). Visual modeling as a motivation for studying mathematics and art. *Educational Media International*, 42(2), 173-180.
- [10] DeepArt, a novel artistic painting tool. (2018, July 26). From <https://deepart.io/>.
- [11] Elgammal, A., Liu, B., Elhoseiny, M., & Mazzone, M. (2017). CAN: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*.

- [12] Konkursy - OEliZK. (2018, July 26). From <http://konkursy.oeiizk.edu.pl>.
- [13] Minilogia competition. (2018, July 26). From <http://minilogia.oeiizk.waw.pl>.
- [14] Logia competition for gymnasium students. (2018, July 26). From <http://logia.oeiizk.waw.pl>.
- [15] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- [16] Katterfeldt, E. S., Dittert, N., & Schelhowe, H. (2015). Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction*, 5, 3-10.
- [17] Kastl, P., Krisch, O., & Romeike, R. (2017, November). 3D Printing as Medium for Motivation and Creativity in Computer Science Lessons. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 27-36). Springer, Cham.
- [18] Beetle Blocks - Visual code for 3D design. (2018, July 26). From <http://beetleblocks.com/>.

WG4: The Constructive Strategies in Teaching Humanities with Films⁸¹

Jūratė Baranova, *jurabara@gmail.com*

Vilnius University Institute of Educational Sciences, Lithuania

Lilija Duoblienė, *lilija.duobliene@gmail.com*

Vilnius University Institute of Educational Sciences, Lithuania

Luc Anckaert, *luc.anckaert@kuleuven.be*

KULeuven and Vives, Department of Paramedical Sciences and Biotechnology, Belgium

Wilfried Baumann, *baumann@ocg.at*

Austrian Computer Society, Wien, Austria

Abstract

Our contribution investigates the question whether is it possible to apply constructionist methods of education in teaching humanities with fictional films. From a more sceptical viewpoint one could argue, that this is not possible for several reasons. The article starts the discussion, suggests some arguments for justification of positive answer and reveals several different possible experiences of including constructionism in teaching humanities with fictional films. The theoretical basis for these approaches are the pragmatic pedagogy of John Dewey, the Deleuzean theory of cinema and Paul Ricoeur's theory of personal identity. The article deals with the several different possible experiences of including constructive strategies teaching humanities with films.

Keywords

Constructionism; humanities; teaching films; Deleuze; film theory; personal identity

1. Introduction: The Challenge of the Sceptic

Let's start to analyse the problem from the sceptical challenge. One can say that teaching social issues with fictional movies is not in line with constructionism for several reasons. First of all, the sceptic says: An educational process implies there is a consensus on what has to be taught and especially in this case what are the commonly agreed on moral values. While this seems straightforward for many questions this will surely be controversial for others. Who decides those cases? The sceptic does not trust in teacher's or professor's moral sense. Secondly, the film industry is just another industry, trying to make money and under the control of people that follow their own personal interests and not the interest of society as a whole. For any educational process we should ask if it is beneficial for society as a whole. There is a thin line between education (which should benefit society as a whole) and manipulation (which usually only benefits the manipulator).

Thirdly, Constructionist learning is when learners construct mental models to understand the world around them and learning can happen most effectively when people are active in making tangible objects in the real world. But there is hardly anything more passive, according to the sceptic, than watching a fictional movie. It tries to involve us emotionally, to incite feelings like anger, guilt, pity, pride, love, fear, empathy, curiosity, etc. The fictional movies merely give the audience the feeling to be involved at best, which is not in the sense of the active involvement required by constructionism.

Fourthly, the sceptic supposes the best what one can expect from the fictional movie is some sort of the information. Information that is presented visually is often easier to understand and more convincing than textual information. The effect of visually acquired information is in some cases longer lasting than other information because visual information can sometimes be recalled easier than other information.

⁸¹ The half of the article is based on the investigation included into the project "Philosophical Sources and Problems of Multimodal Education" financed by the Research Council of Lithuania (No.S- MIP-17-37)

Written text, like a novel, can just as well try to evoke powerful images but with a movie a much larger group of people can be addressed. Many people lack the reading competence, the imagination or the patience to read a novel. In a movie it is easily possible to add subtle, implicit, additional information that is registered and processed subconsciously at least partly. This somewhat subliminal content passes the critical barrier every person is equipped with, much more easily because it is difficult to scrutinize information whose acquisition someone is not aware of. This additions could be specific visual clues, for example like subtle properties of the clothes the hero or the villain is wearing, it could be the sound track or it could be generic visual effects like a fish eye view, a tint, a soft focus or a blurred view. As a consequence from above, some people cannot question what is presented to them visually they believe everything they see in a film is real. And this visual access to the information has nothing to do with the student-centred, discovery learning.

Firstly, despite of these doubts the sceptic supposes that some movies, albeit usually not the ones dealing with social issues, have large followings (Star Trek, Rocky Horror Picture Show) that attend fan screenings where members from the audience dress up in costumes and re-enact scenes from the movie. Although this is an act of imitation rather than a learning process, it hints that active involvement is possible and even desired by parts of the audience. How could such an active involvement look like? There could be physical symbols for representing characters, important objects or locations from the movie, satisfying the haptic element of Constructionism. Participants could discuss and re-enact alternative endings, additional scenes, sequels, turns of the stories that they didn't like, ethical implications etc. There could be a prepared list of tasks to perform or questions the group has to work on. Members of the group could change the story add new characters or change their role. The results could be added to a database and the group could be able to compare their results to results from other groups.

Also, participants could be encouraged to create a work of art inspired by the movie they watched. This could be a drawing, a painting, a poem... This work of art can but not necessarily has to reference any elements from the movie. This work of art could express different feelings that the viewer experienced. But this work could also contradict, challenge or ridicule the movie.

2. Theoretical Basis: Addressing the Scepticism

The response to the challenge from the followers of using fictional movies in constructionist learning would be following. First of all: to the challenge that educational process implies there is a consensus on what has to be taught and especially in this case what are the commonly agreed on moral values we reply: Distrust for the teachers ability to discern by herself/himself what the moral values are is the feature of totalitarian education, where all the rich variety of ethical life is reduced to several simplified concepts. We consider that the educator's diploma suppose the required ability in the incommensurable alternatives of real life take the courage for the moral decisions, also in addition to it we are following Immanuel Kant's insight that rational agent is able to understand the essence of categorical imperative for the reason of his rationality (even in the case she/he is Moon dweller).

Secondly, we do agree with the sceptic that film industry is something different from the process of education. It has its own pitfalls, which were reflected by the film philosopher Gilles Deleuze, when he wrote: "Cinema is dying, then, from its quantitative mediocrity. But there is a still more important reason: the mass-art, the treatment of masses, which should not have been separable from the accession of the masses to the status of true subject, has degenerated into state propaganda and manipulation, into a kind of fascism which brought together Hitler and Hollywood, Hollywood and Hitler" (Deleuze 1989: 159). But on the other side cinema is not the only one of the spheres of culture with the sin of collaboration with fascism. Yes, Leni Riefenstahl, the talented film director can be blamed for her sympathy with National Socialism But the fact that such famous writers as Ezra Pound or T.S.Elliot or Louis-Ferdinand Celine also expressed antisemitic views does not stop from using literature in the process of education. The same case is with the fictional films or the other branches of art. The kitsch in the painting or unsuccessful piece of the music do not exclude the power of talented works. The fact that some sort of film production is mediocratic, manipulative or express doubtful values does make the films as the real pieces of art less valuable for the purposes of education. How to find the distinction? The suggestion of Deleuze would be very simple: just concentrate an attention to the good films. In his

philosophy of cinema he had found and discussed on about four hundred 'good movies'. Of course, in this case the educator should have some sort not only moral rationality, but also the developed aesthetic taste to discern between mediocratic and good movies. In difference from the sceptic we treat fictional movies first of all as the works of art. So we see deep need for using arts in the teacher's education.

Thirdly, we oppose the sceptics' opinion, that the films influence only the emotions of their spectators. It would be very limited aesthetic theory. As we treat them as the pieces of art, we consider their influence in regards to Kant understands of the nature of aesthetic taste, it means as the disinterested game of imagination and intellectual abilities. Fourthly, so in contrast to the sceptic's view, we do not make emphasis on the process of the gaining information when using fictions films in humanitarian education, but consider them as the challenge for creative mental learning and provoking critical abilities of the students. We are following Deleuze's insight, that watching the good movies first of all is encounter, which inspires a thought. But not the thought stemming from the sources of everyday experience, but the special type of the thought, in Deleuze words "the identity of thought with choice as determination of the indeterminable" (Deleuze 1989: 171). The spectator of serious movies faces the problem: how to withstand something which is unbearable and unthinkable in reality. So firstly, we do agree that it is possible and useful to use methods of creating the different endings of films stories, the same as it would be possible to play with the written texts. It can create much fun and amusement for the students as well as for the educator. But the question remains about the value and purpose of such recreated endings. Do they have the value in itself as the source for the entertainment? We do not trust the young people to take the life seriously and construct by the use of their imagination and mental skill their own reading of serious cinema as a valuable piece of art? Modern cinema is only indirect representation of reality. But in evoking the special kind of thinking, it has the power to restore our belief in the world, says Deleuze (see Deleuze 1989: 166).

3. Methodology

The research is provided using and combining theoretical analysis and three case studies. The thought experiment is used in justifying the possibility of applying the constructivist methods in teaching humanities with films. The analytical review of critical literature is used to clarify the theoretical premises of research. The qualitative experiment is used investigating the possible involvement of students. The phenomenological method is used reflecting the analysis of the data. The comparative method is used to investigate the distinction between associational and differentiatinal type of thinking. The hermeneutical method is used link the Biblical texts with possible interpretations of the film.

4. Teaching with films: associated and differentiated access to thought's construction

Looking on the experience of teaching with films in different countries, we can find more than few strong traditions. We would like to analyze at least two of them. One, which is oriented toward teaching social problems, is elaborated by William B. Russell in his book *Teaching social issues with films* (2009). The other one, which is elaborated by Jasson Walin in *A Deleuzean approach to curriculum. Essay on a pedagogical life* (2010) and Jan Jagodzinski *Visual Art and Education in an Era of Designer Capitalism* (2010), also their publications in the book *Deleuze and Guattari, Politics and Education. For a People-yet-to-Come* (2014), suggests new way of teaching with films for critical and in the same time creative purposes. Both traditions are quite new, both are oriented toward teaching and learning to think in a special constructional way and the second one, following Deleuze, operates not with association, but with differentiation.

The two schools mentioned above demonstrate that different approaches to apply films for teaching and learning can follow distinctively different methods. They also show what understanding of the world any method constructs and what results we can expect. The first tradition of teaching with films, represented by Russell teaches to understand, to recognize and to use associations, whereas the second one, represented by Wallin and Jagodzinski, teaches to criticize, to experiment, to invent and to create. Construction of concepts, thoughts and meanings - in other words - ways of thinking are very important

for both of them. Learning by construction of knowledge forms the basis of constructivism (Phillips, 1995, von Glaserfeld, 1995, Piaget, 1955, Vygotsky, 1997), and constructionism (Papert, 1980; Berger, Luckmann, 1967). Both traditions mentioned above are sometimes referred to with either of those terms by scholars, depending on approach and arguments. We suggest the way which allows going beyond constructionism.

Some of scholars, investigating constructionism and constructivism, and separating *cognitive constructivism* from the *metaphysical constructivism* think, that “metaphysical issues are irrelevant to the pedagogical enterprise except when explicit philosophical issues arise”. (Grandy, 1998, 415). This is the state which is applied by Grandy to science teaching. We are thinking about social and humanities having in mind studies on pedagogical or teacher’s training courses, teaching as well as learning at school. We think that metaphysics plays an important role for pedagogy and when teaching critical thinking and agree with the idea of Rosi Braidotti about the purpose of humanities, expressed with these words: “We know by now that the field is richly endowed with an archive of multiple possibilities which equip it with the methodological and theoretical resources to set up original and necessary debates with the sciences and technologies and to meet other grand challenges of today” (Braidotti, 2013, 15). So in which sense humanities and teaching to think constructively is important during the classes for film analysis and why it is beyond constructionism?

In the books *Teaching social issues with films* Russell suggests methods, schemes, templates of survey, describing and exemplifying the tools very precisely. His main focus is on teaching social problems, such as poverty, drugs, asocial/risk families, bullying, addiction, depression, aids marginalization of cultures, care about animals and others sensitive topics (30 social issues). These problems are suggested by Russell to analyze because of related to American social life, culture and pedagogy, but not only; it is common for many countries. The theoretician formulates concrete questions to work with film material: questions for gathering information, analysis, interpretation and creation. Students can recreate the end of film or even synopsis, to present their fabula (how they understand film with their personal narration). They are inspired to discuss hot problems, to evaluate the actions of heroes and position of film director. The teachers are instructed with the list of films, proper according to student’s age and must to avoid the scenes of violence and sex.

What does Russell use for theoretical standpoint and methodology? It is mostly Driscoll and Engle teaching of critical thinking, decision making and reflective thinking (Russell, 2009). That is also close to the Deweyan tradition of problem solving, trying to find answer for the question one has. Dewey it calls interest. Interest is the main vehicle. So films give great possibility to compare what is already in students experience and to think following moving image. Dewey in *Democracy and Education* says: “Any activity with an aim implies a distinction between an earlier incomplete phase and later completing phase; it implies also intermediate steps. To have an interest is to take things as entering into such a continuously developing situation, instead of taking them in isolation”. (Dewey, 1997, 137) and adds quite pragmatic note, that thinking “is the intentional endeavour to discover specific connections between something which we do and the consequences which result, so that the two become continuous” (ibid, 145). Theory of teaching with films, based on this standpoint, suggests construction of world view and especially understanding of social field through critical and reflective thinking and decision making. It deals with connection of various elements and involvement of new elements, which appears in the process of learning, also reconstruction of presented ideas based on students own understanding, values and experience. Russell states, that such tool for film analysis “increase students interest in the material being studied, thus allowing it to become more meaningful and relevant to the student. Furthermore, authentic classroom activities help teachers achieve instructional goals such as retention, understanding, reasoning, and critical thinking”. (Russell, 2009, 2).

Russell presents a list of films, mostly well-known and popular (Schindler’s list, 1993, *Trainspotting*, 1996, *The Terminal*, 2004, *Scarface*, 1983, *American Girl*, 2002, *The Virgin Suicides*, 1999), which are awarded in international films’ festivals. They are very good and undoubtedly fit for analysis of social issues, giving a lot of material for the interpretations. On the other hand, some films, when teaching to perceive situations associatively and to make decisions in accordance, give cliché instead of creativity.

Wallin and Jagodzinski suggest bit different kind of films, probably not always good for teaching teenagers, much more fitting for curriculum studies at the university level. Anyway, they expect to bring to the schools the new way of thinking. Wallin suggests film of Jim Jarmush *Ghost Dog: the way of Samurai* (1999), also Quentin Tarantino *Kill Bill* (2003) and Todd Haynes *I'm not there* (2007). He demonstrates how to work with the problem of time and atemporal person, how to link different heterogeneous lines of actions, how newly to treat the absent of arguments for the agreement, and the problem of multiply identity. In that way he expects to protect students' thinking from cliché, stereotypes, and to escape technics of repression during pedagogical activities. Teaching with films according to him is step forward from banking education, criticized by Paul Freire. Wallin explores Deleuzian and Guattarian concepts and cinema theory about time-image, which interrupt into movement-image and in that way brakes dialectical understanding of film actions, instead - open space for imagination and unexpected combinations of elements which fulfill the cracks, gaps, ruptures in films, which appear by specific montage. Montage of such films is differing from classical montage, because is oriented toward presentation of intervals rather than connections of shots.(Milerius, 2013). It can be called as *montrage*⁸². Here is a lot of space for linkage of different heterogenous elements and planes, because the main vehicle in film teaching is not the interest, but desire and affect. How they differ? If the interest is oriented toward concrete results, desire much more works for involvement into the process of creation through the affect (Deleuze, 1995). That means active participation in creational process, which is not personal, it means being a part of assemblage. So the construction is not personal and even not social, much more mechanical, depended on combination of organic and artificial, real and imaginary, social and natural. It is construction not of forms, but of forces. It is not about identity and individual, but about individuation and becoming. As Deleuze states: "Cinema always narrates what the image's movements and times make it narrate. If the motion's governed by a sensory-motor scheme, if it shows a character reacting to a situation, then you get a story. If, on the other hand, the sensory-motor scheme breaks down to leave disoriented and discordant movements, then you get other patterns, becomings rather than stories".(Deleuze, 1995, 59). Breaking, crossing and displacing appear as main tools as well as cracks and ruptures. Stemming from Jan-Luc Godard films Deleuze states: "This is not an operation of association, but of differentiation, as mathematicians say, or of disappearance, as physicists say: given one potential, another one has to be chosen, not any whatever, but in such a way that a difference of potential is established between the two, which will be productive of a third or of something new" (Deleuze, 1989, 179,180). Such a method Deleuze calls "Between": between two visual or sound images, between two affectations, between sound and visual image et ctr. Wallin also uses Godard way of montage for the interpretation of *I'm not there*, and especially Godard words: "It is not where you take things from- it's where you take them to" (Wallin, 2010, 194).

Working in the same tradition Jagodzinski analyzes films such as Sergei Eisenstein's *Battleship Potemkin*, (1925), Joaquino Fernandez, Colino Gunno *Indoctrination* (2011) or Lana and Lilly Wachowskis *The Matrix* (1999), when first one demonstrates dialectical move and spectator involvement into action and third one – erases border between reality and hyper reality. Using these and other films Jagodzinski pay attention onto the message of ideology, which is perfectly demonstrated in the film of Eisenstein and from the other side – possibility of another construction of world view, which is much more complicated and much more rich, integrating what is imaginary, virtual and only possible, as it is in the film of Wachowskis. In both cases students are inspired to understand the construction of their worlds, though access is different in both films cases: to construct one's vision according to ideology and concrete expectations (*Battleship Potemkin*) and to show how one can be constructed in modern, much more complicated reality, through the erasing borders between real and imaginary, natural and technological, human and not human (*The Matrix*). Jagodzinski criticizes construction of people consciousness, especially of students, and following Deleuze outlines new way of presenting image, which does not fit for narrow formation of thinking as well as marketization and selling of image for the masses. Moreover new way of films montage, based on differentiation and intervals, helps to destroy usual way of thinking, which is easy going for manipulations, dominant policy and dominant pedagogy.

⁸² According to Nerijus Milerius (2013) it stems from the word *montrer* (fr. to show)

The other kind of films, which he chooses for the analysis in the classes – is of performansist Bill Viola, who is famous for his video installations such as *The Greetings* (1995), *Five Angels for the Millenium* (2001), *The Raft* (2004). His installations allow appearing in different realities. Viola works with different consistencies, especially water and absolutely different speed of movement in it. That is also related to different time perception, quite close to filming in style of slow mood. Jagodzinski's examples of video projects and fiction films allow understanding world as infinitive; more than one usually see and hear. This method helps to open thinking for imperceptible and what is only possible. It is not the matter of how film or performances directors express their view, rather how student are involved in film as machine, working through affects, percepts and concepts. So the student during the humanities lessons participating actively and creatively creates the thought and is created by thoughts. She or he becomes one of the elements, integrated in the assemblage as creational machine. Not because of ideological construction, but because of creational event, this happens during the classes. Jagodzinski, who follows Deleuze, thinks that such a films allows to be much more closer to virtual world, full of surprises, new combinations, and in that way to expend teaching and learning possibilities. The aim of visual studies according to Jagodzinski is to investigate paradoxes of "lived" life. "The power and *force* of the image *in an expanded sense* (be it in performance, film, television installation, and so on) reside in its affect or intensity in parallel with its contents. This means that semantically or semiotically ordered levels of analysis—representation as such—are no longer adequate for the task. A turn to philosophies of the unconscious that address the paradox of these two levels—the semantic and the affective—as they work and twist with each other in different contexts becomes a necessity for VCAE's⁸³ advancement" (Jagodzinski, 2016, 104).

The author of this sub chapter experimented with many films, using different methods, working not with school students, but with university students of pedagogical studies. The main result working with them is their huge interest in new paradigm, new way of thinking, constructing their inner and extrinsic world not with very concrete separate elements of clear shape and content, but with elements of absolutely different level, plane, from the different assemblages. They work on combining heterogeneous elements and consequently inventing the world in the process of *becoming*. They watch and interpret but also create practically during the course of *Visual studies and education*. Students already created their own multimodal projects, trying to find proper images for their ideas, to combine them, to add any existed or to create sound track. They presented wonderful examples of mixing elements of different levels or fields and producing very unique audio-visual constructs, for example: mixing videos from funeral of President Kennedy and concerts of *The Beatles*. In this combination the visual images, expressing different crowd's emotions are absolutely mischievous. Additionally it was complicated with the sound track, which was created separately and was not diegetic - not coincided with visual image. The feeling of reality was mixed with fantasy of creator and spectator. The perception and understanding of separate events were blundered and in that way created enigma of the film. The other students' film used the image of the legs of school children, their move under the tables, and special sound track. This combination provoked to think the idea of film in many directions. The question is – why educational programme students, who are ready to go to work in schools choose to experiment with sound and image in unexpected way looking for new combinations in contrary to the creating projects in traditional way? The same tendency is evident during the analysis of films in the classes, such as Peter Weir *Dead poet society* (1989), Eric Toledano *Intouchables* (2011), Hal Hartley *Unbelievable truth* (1989) and Wim Wenders *Wrong move* (1972), then huge attention was shown for the *Wrong move*, which mostly expresses new way of thinking; thinking of infinity, experimenting, inventing and thinking in different directions according to the unpredictable vectors of nomad. The result of teaching with films is not students' thinking according to given new constructs of film, but thinking side by side with the invention of constructs, experimentation and creativity. Such a methodology of teaching thinking is not as much humanistic, as it is post-humanistic, oriented toward link of all fields- human and technologies, natural and artificial, actual and virtual, and it is beyond constructionism. Through the differentiation, cracks, ruptures, inbetweenness, involving also new elements, it deals with broad scale of elements in the same time is the part of much bigger creational and constructional process, which is more than human. Coming back to the question about empathy, feelings, which are expressed in the films: will we skip

⁸³ VCAE- Visual, cultural and art education

them? New teaching thinking does not stress emotional field. It combines perceptions, sensations and thoughts. All are important, though thoughts are axis for the construction beyond already legitimated constructionism.

5. Constructivism in Cinema: What Films are Suitable for Humanitarian Education?

Matthew Kearney and David F. Treagust demonstrated how in teaching Physics the educational research on constructionism can be united with the development and use of a multimedia computer program. Using interactive digital video clips they presented sixteen real world demonstrations to Physics students in order to elicit their pre-instructional conceptions of Force and Motion and encourage discussion about these views. (Matthew Kearney, David F. Treagust 2001). We ask alternative question: can the educational research on constructionism and multimedia be useful in teaching not only Physics, but also and Metaphysics, having in mind not only Philosophy, but also the Humanities in the broader sense. Is multimedia able to elicit students pre-instructional conceptions of Human reality and encourage discussion even about education itself? Can it be considered as the Copernicus turn in Humanitarian education?

Our preliminary hypothesis is that some cases of multimedia really are able to meet this particular challenge. We restrict this particular our research to fiction movies.

The history of fiction movies reveals the origin of the term 'constructionism' and its practice. It originated in Russia at the beginning of the 20th century in the sphere of revolutionary architecture and design. It was also used and reflected by Russian film director Sergej Eisenstein in the process of creation of the revolutionary movies. Eisenstein discovered not only the possibility of the dialectical editing, but also reflected how to construct the fiction movie in order to impress the public and to impose the ideas the creator intends to impose. Every piece of art, says Eisenstein, has an educational purpose (Eisenstein). This educational purpose in film is reached by the dialectical construction of the film as organic whole, which leads to the pathos experienced by the spectators when the film reaches its climax. Eisenstein realized the artistic essence of the cinematic image as "*producing a shock to thought, communicating vibrations to the cortex touching the nervous and cerebral system directly*" (Deleuze 1989: 151). In his book *Problems of Film Director* Eisenstein step by step reveals the inner laboratory of the constructing his the most popular movies, which imposed the belief of the inevitability of the Socialist October Revolution in masses. Was this revolution in reality so inevitable? Or maybe just the result of the successful conspiracy of the small group of Bolsheviks and Eisenstein with his films persuaded the public of its historical meaning? After watching the Eisenstein's film *Battleship Potemkin* Joseph Goebbels, the Nazi propaganda minister, reflected; "anyone who had no firm political conviction could become a Bolshevik after seeing the film.' The most celebrated scene in *Battleship Potemkin* is the massacre on the Odessa Steps. The sequence is built from separate shots combined in a very dynamic, rhythmic way. Eisenstein edited the film to produce the greatest emotional response, so that the viewer would feel sympathy for the rebellious sailors of the Potemkin and hatred for their overlords. The case with the Eisenstein's constructivist approach used for the aims of destructive and dangerous propaganda reveals the limitations and possible pitfalls of the constructivist approach. If the truth about the reality is unattainable and we rely as on knowledge on our own world view constructed by ourselves, why not to construct our own truth and by the means of personal genius not to impose on the rest of population? Where are the limits indicating the film director have to stop? As a matter of fact Eisenstein in the history of film making as usual is praised for his genius findings in technique of film constructing and the destructive impact of his movies to the history of his country as usual is skipped with silence and tolerance (see Baranova 2017).

The opposition to Eisenstein's constructivist theory of editing was anti-constructivist approach to film editing suggested by the Russian film director Andrei Tarkovsky. Tarkovsky in his book *Sculpting in Time* opposing Eisenstein conception of montage wrote: "I feel that Eisenstein prevents the audience from letting their feelings be influenced by their own reaction to what they see. When in *October* he juxtaposes a balalaika with Kerensky, his method has become his aim, in the way that Valery meant. The construction of the image becomes an end in itself, and the author proceeds to make a total

onslaught on the audience, imposing upon them his own attitude to what is happening” (Tarkovsky 1987: 118). Tarkovsky rejects constructivist approach trying in films to catch the glimpses of reality itself. The signs of reality coincide with the signs of time. Reality reveals itself through flowing time. “Time in cinema becomes the basis of bases, like sound in music, colour in painting”, writes Deleuze reflecting Tarkovsky’s idea (Deleuze 1989: 288). Montage, says Deleuze, as following Tarkovsky, can be only indirect representation of time. But how to reach the direct image of time? How to approach the reality itself? To our research the following question is also very important: has an educator moral right to use films in teaching process in the manner of the constructionism practiced by Eisenstein (organic model of cinema education, using Deleuze’s terms), or should an educator encourage the students to be influenced by glimpses of flowing reality he is able to discern in *hors-champ* (beyond field) of the film (crystalline model of cinema education)? Can the films be used as ideological materials in education or just as the source of more profound questions about human reality? If one chooses for the answer the second alternative the questions still remains: how it is possible to create the movie without ideological input? Do such type of a film have any educational meaning at all? What kind of film should be used in humanitarian education?

Deleuze would had answered: good films. Not depending on genre or topic. Good films, according to Deleuze, are those which are able to restore the link between man and the world which is recently broken: “Only belief in the world can reconnect man to what he sees and hears”, concludes Deleuze (Deleuze 1989: 166). Deleuze does not state that the films will open what reality is in itself. He does not join the position of intuitive realism. And in this respect he is closer to constructionism than to realism. But emphasizing the need for belief in reality he becomes rather close to William James’ fideism. This statement can be declared as the one of the main educational aims encouraging to include films in the curriculums of the humanitarian education. “Something in the world forces us to think. This something is an object not of recognition but of fundamental encounter”, writes Deleuze in *Difference and Repetition* (Deleuze 139). Watching films one encounters the glimpses of possibilities from the different specter of perspectives one is not able to experience in everyday life. We will add: the selection of films for humanitarian education needs to meet the requirement of hidden secret directly not expressed in the image. The education process should presuppose the hidden encounter. As one of the first year philosophy student wrote in essay about impression of watching Krzysztof Kieslowski film *Three Colors: White Trois couleurs: Blanc*: “Every minute of watching film and the feeling afterwards I experienced strange feeling: I can define it as puzzlement, at the same time as the silent admiration, but also the inability to comprehend what is going on” (K.K.). The learning experiment with the film was provided in order to discuss the concepts of optimism and pessimism, the will to belief, the will to die and the will to live, stemming from William James pragmatism in the course of the philosophical ethics. The encounter with the film as event leads twenty students who participated in this educational experiment to the different conclusions. Some of them concluded that the moving stimulus for the main character Karol radically change his life was just the fact of his temperamental optimism, some of them – the obsession with love for his lost wife, some of them – the need for revenge, one of the students refused to suggest his own explanation saying it would be too oversimplified. All the views were expressed during the seminar in the form as a conference. The educator had no purpose to present one final and generalized point of view pretending to the ‘truthful interpretation’. She only expressed her own point of view open to criticism as well.

Kearney and Treagust in their already mentioned experiments with using digital multimedia in promoting a student’s conceptual development in the domain of Physics discerned four methodical steps: a. articulation and/or justification of the student’s own ideas; b. reflection on the viability of other students’ ideas; c. critical reflection on the student’s own ideas; d. construction and/or negotiation of new ideas” They also concluded that this “program provides students with an opportunity to engage in ‘science talk’ ... and a means of developing science discourse skills (exploration, justification, negotiation, challenge etc.) (Kearney and Treagust 2001: 69).

The first two steps can be noticed in the experiment with the movie *White* provided in the course of philosophical ethics. Are students able to learn from these different points of view in this open discussion and to modify their primary insights, as is presupposed in the pedagogy of social constructionism? Is there a need for this modification mainly in humanitarian education? Or the educational outcome from

this teaching experience is the encounter with the inevitability of living of pluralistic social universe? The educator also supposes that something in the process of pedagogical experiment should remain as the secret also for the educator. On the other hand analysis of the essays reveal that the students are able to change their opinions even during the process of the reflection in writing and keeping inner monologue with themselves. Student A.L. in essay starts the reflection of movie from sceptical tones. He had read in advance and before that from all the trilogy of Kieslowski *Three colours* this particular film *White* is the weakest at all. After the reflection in written form he ends with rather different conclusion: "It is very difficult to summarize such a subtle and in many aspects ambiguous (from a point of view of and moral posture) film *White* by Kieslowsky trying to reduce it to one or the other stimulus. It would be an idiotic attempt. Kieslowsky is not sorcerer who pronounces how the things should be and how it is necessary to behave. But the geniality of the film and certainly of Kieslowsky reveals itself when the situations – hypothetic or realistic open the plan for the question: what is *this*, *which* forces us to act and not to give up. And the complaints expressed in the beginning of essay do not seem any more so justified (because of the unclear end and all other things). The meaningful is the opening up the field for discussion" (A.L.). "It was very interesting for me to learn that it is possible to consider Karol's action as the revenge or obsession with love. It did not come to my mind when I watched the movie", reflected one of the students K.B., who relied on the alternative of temperamental optimism.

So the films suitable for humanitarian education are the films which resist one straightforward interpretation and create the field for possible multidimensional social constructionism in learning. Deleuze in his philosophy of cinema discerned two kinds of constructionism in film making: the idea of a vertical construction going right to the edge in both directions suggested by Eisenstein – and already discussed; "The whole was thus being continually made, in cinema, by internalizing the images and externalizing itself in the images, following a double attraction. This was the process of an always open totalization, which defined montage or the power of thought" (Deleuze 1989: 193). On the other side Deleuze mentions the constructionism of French director Jean Luc Godard who's aim was not to construct the whole, but who supposed that 'the whole is the outside'. The point of constructionism of Godard is quite different. In the first place, says Deleuze, the question is no longer that of the association or attraction of images. "What counts is on the contrary the *interstice* between images, between two images: a spacing which means that each image is plucked from the void and falls back into it" (Deleuze 1989: 193) From the point of view, of virtual constructionism, which tries to reach the whole Godard experiments with the disconnected images which bring together Golda Meir and Hitler in *Ici et ailleurs*, Deleuze acknowledges, would be intolerable. We are also not ready to suggest the reading of Godard films for bachelor students. But for the master students who took the course of Deleuze studies seminar the experience of reading of Gardard's films can be productive. Godards experiments with the Hitler created the scandal. Deleuze is on the side of Godard: "But this is perhaps proof that we are not yet ready for a true 'reading' of the visual image. For, in Godard's method, it is not a question of association. Given one image, another image has to be chosen which will induce an interstice *between* the two. This is not an operation of association, but of differentiation..." (Deleuze 1989: 193).

6. Film as an imagination for creating identity

Already Aristotle pointed out the mimetic character of the arts that lead to a catharsis. The *catharsis* - hardly described by Aristotle in *Poetics* - can be understood as the therapeutic concept and insight into one's own emotional relationship to reality and the harmonization of emotion and reflection. The *catharsis* is the result of the mimetic-narrative structure of the identity. In narrative mimetics, the *idem* is challenged to follow the long path of emotional identifications that can lead to a cathartic insight into the narrative constructed identity as *Mimèsis II*. The film *Va, vis et deviens* (Radu Mihaileanu) illustrates this process. Within educational processes he forms an invitation and challenge to purify the own mimetic processes by placing them in a broader metaphorical context. The film is the imaginative-metaphorical enactment of the identification process

The impressive film *Va, vis et deviens* paints the intriguing fates of an Ethiopian child who gets lost in the misery of an African refugee camp in Sudan. As an adopted orphan, Solomon grew up in Israel. He studied medicine in Paris as an adult and married the girl with whom he had been in love for a long time

in Jerusalem. Eventually the main character returns to Sudan as a *Médecin sans frontières* and finds his old and shriveled mother in an emotional scene.

At the heart of the movie there is a kind of Talmudic discussion between the white Jewish boy Michael and the black adopted son Solomon. The question of Adam's skin color is at the heart of the discussion. Like in a miniature, the central challenge is meticulously worked out.

The imposed question is: "What is color of Adam's skin?" Solomon had asked the rabbi to participate in the discussion in order to prove his Jewish identity. The white Michael is the first to be given the floor and makes the following statement, skillfully supported by references to the Torah:

Michael:

Adam was created after the image of God. And the beautifully chosen color was white. We were all this, in the beginning.

After the flood Noah and his sons left the ark. Noah cursed the descent of his son Cham, whose grandson Canaan said "Cursed is Canaan: slave to his brothers he will be" (Genesis 9:25).

Kush, the eldest son of Cham, is the heir of another curse. Certain descendants of Cham are said to have black skin.

This is how it happened: Kush became black and from him were born the *Kushim*, the black people of Africa. The descendants of Cham have become slaves and blacks.

Michael's argumentation is built up in the three steps of the classical mythical narrative of the golden origin, the decay and the restauration. First of all it is said that man was originally created after the image of God. The objective reference is linked to a subjective interpretation from which the conclusion is drawn that everyone originally belonged to the white race. The white skin is linked to the created original. Then social inequality is justified by the flood and by the curse of one human being by another. The curse as a sanction leads to the slave status. Finally, the racial differences are also explained by a curse. Social inequality is linked to racial inequality.

Salomon initially wants to repeat the starting point of his opponent, as it fits within the genre of the proposed discussion:

Salomon:

Are Adam and God of the same color, white?

... Michael has said. ...No.

After the hesitation and interruption of the discourse, Salomon looks at the Qes Amhra, the Jewish black religious leader who had fled Sudan with him. Solomon had come to know the man as the figure who protested against the second-class position of the Falasha Jews by invoking their recognized Jewish identity as legitimate descendants of the Biblical king Salomon and the African queen Sheba. The Qes helped him write letters to his mother in Sudan, thus forming the symbolic link between Solomon and his lost homeland. When Salomon wants to follow on from Michael's argumentation, the Qes nods no. Solomon remembers his words immediately preceding the discussion. When writing to his mother, his letter was rejected as a mere repetition of pre-existing models and formulas:

No, this is not good. Interpret, Schlomo, do not repeat like a parrot. Interpret. Insert Schlomo into the text. Let us take this up again.

Salomon interrupts his discourse, which threatened to result in a mere repetition of the existing arguments. The duplicate can never have the qualities of the original. Pure repetition of actions and speech leads to decay. It lacks the imagination of *Mimèsis II* that links the present with the past and the future. Ricoeur understands human identity in a non-substantialist way. Human identity is a fragile phenomenon. It rests on a twofold dialectic. The identity can be understood as sameness (*idem*) and selfhood (*ipse*). Within this dialectic, identity is formed by a threefold *mimèsis*. Sameness expresses the permanence in time, the structure of my identity, or *what* I'm. Selfhood expresses the changing identity of *who* I am. Within this dialectic, there is an apodictic element. As sameness, identity expresses the permanence in life, the constant element of identity. But this apodictic identity is not adequate. The

discovery of the meaning of one's own identity is a life-long adventure. Moments of joy and fear, life and death, self-loss, psychological traumas, erotic experiences and hospitality form the concrete content of the person I am ultimately. This construction of identity is happening in a narrative way. Understanding oneself takes place in the exploration of the meanings of the ever-changing *ipse*. By means of narrative identifications, the subject constructs the diachronic self-constancy of the experienced identity within the continuity of the identity (*ipse*). Therefore, identity is not a fact that can be unfolded, but the long detour in which the identity gets a concrete meaning throughout the story of life. The alterity is multifaceted here. This narrative aspect has a triple mimetic structure. The second, central mimesis (*Mimèsis II*) interweaves the activity of storytelling with acting in its temporal character. It is the textual configuration (emplotment) that links with the past (*Mimèsis I*) as prefiguration and with the future (*Mimèsis III*) as refiguration. This has an imaginative structure. In the narrative mimetic process, the identity construction takes place. The relationship to the triple absence of time is constitutive. A metaphorical texture is woven through the *mimèsis*, in which the *idem* acquires meaning as an *ipse*.

Salomon makes a radically different choice and takes a new start. The act of speaking is congruent with an alternative vision of creation. The contradiction between the two visions of creation summarizes the stake of the film in a very concentrated way. Unlike Michael, Solomon understands creation as a continuous beginning and a call to responsibility. Alluding to a Midrash passage on Psalm 90:3 where the word and the Torah are marked as preceding the creation of the world ("In the beginning, two thousand years before the heaven and the earth, seven things were created, the Thora, written with black fire and white fire") and alluding to the opening of the Gospel of John ("In the beginning was the word"), Solomon takes a new beginning.

At first, there was the language, the word. God created the earth and life by giving breath to the word. God has believed in Man. In everyone.

Creation begins anew in every human being and God believes in every human being. In every human being there is a new beginning. In this vision, creation is not a one-off event in which every new person is a repetition of the original, broken condition of existence that is marked by evil. As Franz Rosenzweig elaborates in detail, creation is the continual re-creation of new forms of life, the abundance of being born over and over again. Nothing precedes this creation. This means that no determining or limiting factor - or determining logic - precedes creation that would restrict free creativity. Every creation is a radically new beginning. In this context philosophers and theologians speak of a creation from nothing or a *creatio ex nihilo*. We find this thought very richly expressed in several texts by Jean-Luc Nancy. In his book *Being Singular Plural*, this French philosopher reflects on human existence as absolutely irreducible and singular, but also as fundamentally committed to our fellow human beings:

Right away, then, there is the repetition of the touches of meaning, which meaning demands. This incommensurable, absolutely heterogeneous repetition opens up an irreducible strangeness of each one of these touches to the other. The other origin is incomparable or inassimilable, not because it is simply "other", but because it is an origin and touch of meaning. Or rather, the alterity of the other is its originary contiguity with the "proper" origin. You are absolutely strange because the world begins *its turn with you* [le monde commence à son tour à toi].

The repetition referred to here is not the faithful copy, which is an imitation of the previous one. However, there is a repetition of constantly recurring sources of meaning that are mutually irreducible. Every human being has its own individuality because the world in every human being miraculously begins anew. Through this new beginning every human being is connected with his own origin. Nothing precedes this origin. In other words, being human is not conditioned by the past, the body, the colour of the skin, the gender, etc. Nancy explicitly links this thought with the *creatio ex nihilo*: "Not only is the *nihil* nothing prior but there is also no longer a 'nothing' that pre-exists creation; it is the act of appearing [*surgissement*], it is the very origin – insofar as this is understood only as what is designated by the verb to originate."

The statement about the ever renewed creation is of course not a cosmological statement. It concerns the human condition and the ethical responsibility of the human being. Creation takes place by

interpreting and giving meaning to life and existence over and over again. In order to bring this about from his own life and name, Solomon invokes the text:

He gave us the word so that we could all give it a personal breath, miraculous, different, deep, human. By interpreting the word. As for Adam, his name is derived from 'Adamah', which means 'earth' in Hebrew. God created Adam with the earth of clay and with water. He breathed his breath and breathed into the miraculous, as he did with the word. That's how Adam was born. Adam has the color of clay: red. Like the Indians. Red in Hebrew is: 'Adom'. You see, Adam is neither white nor black; he is red.

As is well known, Hebrew is written in the square-letters, which consists only of consonants. The first word of the Torah looks non-vocalized as follows: ב ר א ש י ת. In the first centuries of our era, in order to avoid any misunderstandings when reading, the Masorets added the vowels in the form of small dots and signs. In his Hebrew grammar, Spinoza reiterates the Jewish-mystical intuition that vowels are the soul of the letters. Without vowels, the consonants are a body without a soul. The consonants become readable when they are animated by the vowels. This inspiration is found in reading. Solomon speaks of man from the same image. Being created is a spiritual event. As a creature, the individual human being is not determined by the burden of the Flood with the racial differentiation, but every human being is a double being: well-founded and rooted in the earth, but also spiritually brought to life. Human existence arises in the creative act: every time physicality is animated and breathed life into in all its diversity. In other words, each person is his or her own origin, a *creatio ex nihilo*. This implies that the colour of the skin does not determine identity. And if there were any doubt about that: if the first person already has a colour - which is actually an irrelevant question - then it is red.

After the liberating perspective from creation, Solomon interprets the discussion from his own breath, his own personal process. Eroticism and an offspring have an important role here.

But, does he feel good, alone, red, in this new world?

Then God thought of Eve. But Adam did not understand what God wanted, what he asked him to do. What was he supposed to be and do down here?

Salomon meets Sarah, a white Jewish woman with curly, red hair. She is the daughter of a politically and religiously highly conservative father. By choosing Solomon, she, like the wife of Biblical Abraham, has to break all ties with her family. As a white Jewish 'princess' - which is the translation 'Sarah' - she wants to create a new generation by marrying the dark king Solomon. For Salomon, however, the intimacy of the sexual relationship means the physical revealing of his identity as a non-Jew. After all, he had not been circumcised. What is his responsibility here?

The question of identity is elaborated narratively in the film in the face of tests:

And what about all these tests?

The tests are represented by moments in which the main character looks at the moon with a dreamy gaze. The moon depicts the mother who stayed behind in Sudan: the absent origin of his identity. Salomon looks at the moon five times. Five times this points to an important turning point. The arrival in Israel sealed his fate as a refugee in order to grow up in a foreign country far away from his mother. He speaks to the moon and he reverses the stone that his mother had given him as a fetish. Then there is a conflict in the school where he is forced to eat and where he claims not to be guilty. The school symbolizes the structural initiation into culture. In the conversation with his mother, he tells that he wants to keep his own identity and that he refuses to adapt to the new culture. He takes off his shoes in a very symbolic way and walks barefoot to the South. A third time Salomon looks at the moon after he has read the opening sentence about creation in the Hebrew Bible. After the acquaintance with the Torah he asks himself the question what it means to be Jewish: to become white, to speak Yiddish or as the well insane Mrs. Silbermann who survived the camps, constantly saying 'Oï, Oï Mein Gott'. After asking this question, he was excluded from school and walked barefoot to the South again. A fourth time he looks at the moon when his ultimate proof of being a Jew - winning the Talmud discussion - is not convincing. Previously he had become a *bar mitza*, fell in love with Sarah and was sent away by her father. After the Talmud discussion, he went to his adoptive parents and told them not to be their son. He leaves for a *Kibbutz* and tells Sarah that he has a new girlfriend, Mandela. However, the resounding name of the African liberation hero is assigned to a bovine animal intended for slaughter. A fifth time

after seeing images of the famine in Africa, Solomon wants to return to Sudan. De Qes prevents him from doing so and tells his own life secret: he has lost wife and children and says he is condemned to live. This condemnation consists of an ethical responsibility: to help those who have survived. Ethical responsibility is not a free choice of human beings, it is a condemnation. After this, the Qes shows the only object he has left from his 'homeland': an old, damaged Thora scroll. In the Talmud we read a strange story about the Torah (*Shabbat*, 88a):

Rav Abdimi bar Chama bar Chassa said: this teaches us that the Holy-Blessed-Be-He has bent the mountain over them in the form of an inverted tub and he said to them: If you accept the Torah, the better; if not, it will be your tomb.

We remember that after the exodus from Egypt, the people were wandering through the desert for forty years. At one point Moses received the Law from God on Mount Sinai. Rav Abdimi bar Chama bar Chassa is said to have said that God, by the gift of the Law to Moses on Mount Sinai, had placed the mountain like a tub above the people. When the people "listen and do" the Torah, life comes into being. Otherwise the mountain remains a private tomb. Doing the Torah is an ethical task the human being cannot escape. But the guidelines of the Torah (Rosenzweig translates the word as *Weisung*) are also the only thing man is given: no country, no people, no religion, nothing. As in the story of the Second Testament about Lazarus and Abraham the Qes says to Solomon: you have the Torah, that's enough. There is no more as an orientation in life, you have to do it with that: *va-t-en!*

The symbol of the moon is in tension with the Jewish identity: land, education, race, language, the Shoah, a community ideal (*Kibbutz*), the Torah... The tension can be summed up in the topic of the lost homeland. What is Salomon's homeland? The Greek *éthos*' means dwelling place. The material, political, cultural, domestic identity of a concrete existence connected in an established people or the ethical homeland of the Torah? During the struggle with his secret, Solomon did not find a house or 'home' in Israel, but became ill from nostalgia for his country of origin. He wants to go south barefoot.

In the film, the moon symbolizes homesickness for the homeland: a concrete, historical reality or an ethical homeland? In order to develop this, we start from a description of dwelling, inspired by Hannah Arendt and Emmanuel Levinas. Identity is never an abstract given, but arises in the course of a lifetime. One is not a human being, but becomes a human being. The course of life is always linked to a place of dwelling, a home. A small phenomenology of dwelling shows that the identity is supported by a threefold attachment. This connectedness is always supported by a separation or an openness to the other. The time of living is characterized by three events: natality, eroticism and fertility. These events form the connection between the past, the present and the future – the emplotment of *Mimésis II*.

Natality or being born shows the passivity of human identity: no one is the origin of his own life. Natality points to some irreducible dimensions of human existence. Man is the result of a relational event of which he is not the source. This means that man is fundamentally intersubjective and is carried by a past that is unobtainable. The physicality of the birth refers to a transcendental origin. In addition, man has been thrown into the world and physically by birth. The fact of the *In-der-Welt-Sein* - which, unlike Heidegger, is understood from the perspective of natality and not death - shows the human openness to life in the world. The closed intimacy of natality is through physicality the possibility of life in the world.

In eroticism man gives himself over to a rhythm of time that is not controllable. Life is physically shared with the other person. This interaction is not the sum of a mutual initiative, but the origin of a new experience. As in creation, a new sentence springs from the eroticism. This takes place at the crossroads of identity and connection. In eroticism one finds oneself through the connection with the other. This event creates a new meaning and future. In eroticism there is an ambiguous interplay of simultaneity and dissimilarity, presence and 'coming'. The eroticism as a concrete time of 'living together' indicates that man is not an original and pure identity (a monadic without windows or doors), but grows in relationships with very concrete others.

Through parenthood, mankind experiences a new future in a child. The term future can be understood in two ways. The French '*futur*' refers to the future as the realization of a project or a plan that man sets out for himself. In this vision, every future moment is the extension of the present or the same. It is the time in which man makes 'projects' that are realized through the development of strategies and manipulations: future reality is calculated and manipulated. The French '*avenir*', on the other hand,

points to this future that literally 'arrives' at man. It is the unexpected and the unplanned that can take place in a lifetime. This future rises above all project thinking. It is the time of the difference in which the other or the strange 'arrives' at man. Parenthood is the flesh-covered experience of the future: in the child a time is opened that is irreducible for any systematic thinking.

The threefold structure of natality, eroticism and parenthood shows the dwelling place of mankind. Man is carried by a triple absence. The origin cannot be repeated (*Mimèsis* I: the stakes of the discussion about the meaning of creation); the erotic connection puts identity at risk and forms her (*Mimèsis* II: Solomon's refusal to reveal the secret of his non-Jew in the eyes of Sarah); fertility is a concrete and physical experience of a different future (*Mimèsis* III: the birth of the son coincides with the retrieval of the mother in Sudan).

The process of constructing identity can respond to this absence in a double way. Either the absent can be recovered: the natality or the origin is then understood as the blood identity (the one-time creation of the white man as an image of God); the erotic becomes profit and selfishness (the temptation scene through the whore and the beating through her pimps); parenthood becomes the insurance of the future (the vision of Sarah's father and Salomon's adoptive father). The double figure of Solomon, however, is the rebuttal of this: his origin is always a lost origin and he goes in search of lost time. The flawless erotic relationship with Sarah is impossible because Solomon conceals the secret of his non-Jewish identity. Parenting is not the establishment of his Jewish identity, but the crisis of homesickness.

The identity is characterized by the tension between *idem* and *ipse*. Is the identity anchored in a substantial and original presence? This is Michael's thesis. When we approach the identity construction from the outlined characteristics of dwelling, it can be seen as a search for the *ipse*. The place of dwelling of this *ipse* is marked by difference, which makes an ethical identity possible. Is identity an ethical identity? Solomon's tests are the question of his responsibility. What am I doing here on earth? The difficulties in exploring the relationship with Sarah show the problems of identity and solidarity.

In education, constructing identity is a search between the *idem* and the *ipse*. Mimetic narrativity can be the royal way in this quest. The inscription of one's own life story in a mimetic plot - for example, provided by a film - leads to the question what is the place of identity: bound to a solid origin or challenged by the fluidity of an increasingly absence. The answer to this question requires mimetic imagination that leads to a *catharsis*.

8. Summary as Conclusion

The authors discovered and suggested the three possible constructive strategies in the process of teaching with films: 1. Creation of the students their own multimodal projects, trying to find proper images for their ideas, 2. Watching films alongside with reflection of some philosophical concepts, afterwards writing essays and presenting them to the group in the discussion as the possibility to encounter the glimpses of reality from the different perspectives. 3. Using film as a challenge to purify student's own mimetic processes by placing them in a broader metaphorical context. The film in this case is the imaginative-metaphorical construction of the personal identification process. The authors discuss the three strategies as parallel, not opposing or excluding each other. All three strategies lead to the constructing students world view not with very concrete separate elements of clear shape and content, but with elements of absolutely different level, plane, from the different assemblages and enlarge the capacity of their critical and creative thinking. Also these approaches develop their social capacities – the ability of the understanding and communication with the different other.

References

- Arendt, Hannah (1958) *The Human Condition*. Chicago: The University of Chicago Press.
- Aristote (1980) *La Poétique*. Dupont-Roc, R. & Lallot, J. Editors. Paris: Seuil.
- Auerbach, Erich (1946) *Mimesis. Dargestellte Wirklichkeit in der abendländischen Literatur*. Bern: Francke.

- Berger, P. L., Luckmann T. (1967), *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. Anchor.
- Braidotti, R. (2013). Posthuman Humanities. *European Educational Research Journal*, Vol. 12 (1). Sage. pp.1-19. <http://dx.doi.org/10.2304/eeerj.2013.12.1.1>
- Deleuze, Gilles (1986) *Cinema 1: The Movement-Image*. Trans. Hugh Tomlinson and Barbara Habberjam. Minneapolis: University of Minnesota Press.
- Deleuze, Gilles (1989) *Cinema 2: The Time-Image*. Trans. Hugh Tomlinson and Robert Galeta. Minneapolis: University of Minnesota Press.
- Deleuze, Gilles (1994) *Difference and Repetition*. Trans. Paul Patton. London and New York: Continuum.
- Deleuze, G.; Guattari, F. (1987). *A Thousand Plateaus: Capitalism and Schizophrenia*. Trans. B. Massumi. Minneapolis, London: University of Minnesota Press.
- Deleuze, G.; Guattari, F. (1994) *What is Philosophy?* Trans.H. Tomlinson and G. Burchell, New York: Columbia University Press.
- Deleuze, G. (1995). *Negotiations, 1972-1990*. New York: Columbia University Press. Dewey, J. (1997). *Democracy and education: an introduction to the philosophy of education*. New York: The Free Press.
- Eisenstein, Sergei (2004) *Problems of Film Direction*. Honolulu, Hawaii: University Press of the Pacific.
- IJsseling, Samuel (1997) *Mimesis. On Appearing and Being*. Kapen: Kok Pharos.
- Grandy, R. E. (1998) Constructivism and Objectivity: Disentangling Metaphysics from Pedagogy in M. R. Matthews (ed.), *Constructivism in Science Education*. Dordrecht: Kluwer Academy Publishers, pp. 113-123.
- Glaserfeld, E. von. *Radical Constructivism: a way of knowing and learning*. London: Falmer press, 1995.
- Jagodzinski, J. (2010). *Visual Art and Education in an Era of Designer Capitalism. Deconstructing the Oral Eye*. New York: Palgrave Macmillan.
- Jagodzinski, J.(2014).On Cinema as Micropolitical Pedagogy: Is there an Elephant in the Classroom? *In Deleuze and Guattari, Politics and Education. For a People-yet-to-Come*. (Eds.) M. Carlin, J.Wallin. New York, London: Bloomsbury, pp. 1-15.
- Levinas, Emmanuel (1979) *Totality and Infinity. An Essay on Exteriority*. A. Lingis, Translator. Nijhoff: The Hague.
- Levinas, Emmanuel (1987) *Time and the Other*. R.A. Cohen, Translator. Duquesne: Duquesne University Press.
- Kearney, Matthew, Treagust David F. (2001) Constructivism as a Referent in the Design and Development of a Computer Program Using Interactive Digital Video to Enhance Learning in Physics, *Australian Journal of Educational Technology*, 2001, 17(1), 64-79.
- Milerius, N. (2013) *Montažas ir intervalas kine in Milerius [Montage and interval in film]*, N., Žukauskaitė, A., Baranova, J; Sabolius, K.; Brašiškis, L. (2013). *Kinas ir filosofija [Film and Philosophy]*. Vilnius: Vilniaus universiteto leidykla.
- Nancy, Jean-Luc (2000) *Being Singular Plural*. R. D. Richardson, Translator. Stanford, California: Stanford University Press.
- Nussbaum, Martha (1994) *The Therapy of Desire. Theory and Practice in Hellenistic Ethics*. Princeton: Princeton University Press.
- Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Idea*, Harvester Press.
- Piaget, J. (1955) *The Child's Construction of Reality*. London: Routledge and Kegan Paul.

- Phillips, D.C. (1995) The Good, the Bad, and the Ugly: The many faces of Constructivism in Education Researcher 24 (7), pp. 5-12.
- Ricoeur, Paul (1969) *Le conflit des interpretations. Essais d'herméneutique*. Paris: Seuil.
- Ricoeur, Paul (1975) *La métaphore vive*. Paris: Seuil.
- Ricoeur, Paul (1983-1984-1985) *Temps et récit*. Paris: Seuil.
- Ricoeur, Paul (1990) *Soi-même comme un autre*. Paris: Seuil.
- Russell, B. W. (2009) *Teaching social issues with Films*. Information age Publishing Inc.
- Vygotsky L.S. *Education Psychology*. Boca Raton, Florida: St.Lucie Press, 1997.
- Wallin, J. J. (2010). *A Deleuzean approach to curriculum. Essay on a pedagogical life*. New York: Palgrave Macmillan.
- Wallin, J. (2014) Education needs to Get a Grip on Life. In *Deleuze and Guattari, Politics and Education. For a People-yet-to-Come*. (Eds.) M. Carlin, J.Wallin. New York, London: Bloomsbury.pp.117-141.

WG5: Constructionism in the Classroom: Creative Learning Activities on Computational Thinking

Michael Weigend, mw@creative.informatics.de

Holzcamp Gesamtschule Witten, University of Münster, Germany

Zsuzsa Pluhár, pluharzs@caesar.elte.hu

Eötvös Loránd University, Faculty of Informatics, Budapest, Hungary

Anita Juškevičienė, anita.juskeviciene@mii.vu.lt

Vilnius University Institute of Data Science and Digital Technologies, Lithuania

Jiří Vaníček, vanicek@pf.jcu.cz

University of South Bohemia, Faculty of Education, Czech Republic

Kazunari Ito, kaz@si.aoyama.ac.jp

Aoyama Gakuin University, Japan

Igor Pesek, igor.pesek@um.si

University of Maribor, Faculty of Natural Sciences and Mathematics, Slovenia

Abstract

An essential assumption of Constructionism is the idea that students learn best when they construct artefacts that they consider to be relevant (Papert). Computational thinking (Wing) is made up of thought processes, such as abstraction, decomposition, algorithmic thinking, evaluation and generalization. This contribution discusses learning activities without computer (“unplugged activities”) related to computational thinking, which challenge creativity. In more detail, these arrangements have these common properties:

- Creativity. The students create a product that can be shown around later. This may be a physical artefact or a performance, which could be documented (photo, video). The concrete outcome is very individual and may be surprising in contrast to analytical tasks with just one correct solution.
- “Unplugged”. The activity is experience-based. It (ideally) demands all the senses and challenges the whole person. The students do not use a computer (“unplugged”) and do not develop a program but may use Lego blocks, pencil and paper or other material found in their environment.
- Time. The activity can be performed without preparation ad hoc in one lesson in 5 to 40 minutes, in contrast to projects that are carefully planned and require weeks of work. The focus is on design ideas.

Based on Csikszentmihalyi’s system model and Margarete Boden’s psychological model, this contribution discusses the possibilities of being creative in the classroom. Extending Csikszentmihalyi’s approach, the classroom situation is seen as a “local” creative system. Specifics of the domain “computational thinking” are discussed.

This contribution suggests to distinguish between four types of creative unplugged activities:

Type 1: Create an algorithm solving a given problem and present it.

Type 2: Create to a given algorithm or informatics concept a new situation, in which this algorithm or concept can be used as well and present it.

Type 3: Create an algorithm with certain structural properties (like loops, recursion, functions calls etc.) and present it

Type 4: Create a visualisation for a given algorithm or concept of computer science and present it.

More than 300 computer science educators (school and university) from different countries were asked about this classification scheme, about their experience with creative unplugged activities and the

relevance and educational potential of the different types. Qualitative and quantitative analysis of the answers suggest that the classification scheme is quite acceptable for the community of CS educators. Creative unplugged activities are not often used in CS education but are still considered to be relevant.

1. Introduction

An essential assumption of Constructionism is the idea that students learn best when they construct artefacts that they consider to be relevant (Papert, 2000). Additionally, the process of creating has educational value in itself, since creativity – the ability to generate new ideas – is an educational goal of high importance in a modern society (Resnick). Computational thinking (Wing) is made up of thought processes, such as abstraction, decomposition, algorithmic thinking, evaluation and generalization. Creativity in computer science lessons often refers to creating digital artefacts with a computer (programming, visual design). Many students love computer science because they see opportunities to be creative using computers (Romeike & Knobelsdorf, 2008). However, computer science (CS) education in schools is more than just programming. To develop computational thinking and get a deep understanding of CS concepts a *variety* of different learning activities might be useful. This contribution discusses activities *without* computer (“unplugged activities”) related to computational thinking and challenging creativity. In more detail, these arrangements have three common properties:

- Creativity. The students create a product that can be shown around later. This may be a physical artefact or a performance, which could be documented (photo, video). The concrete outcome is very individual and may be surprising in contrast to analytical tasks with just one correct solution.
- “Unplugged”. The activity is experience-based. It (ideally) demands all the senses and challenges the whole person. The students do not use a computer (“unplugged”) and do not develop a program but may use Lego blocks, pencil and paper or other material found in their environment.
- Time. The activity can be performed without preparation ad hoc in one lesson in 5 to 40 minutes, in contrast to for example software development projects that are carefully planned and require weeks of work. The focus is on design ideas.

This article is structured in following way. First, we introduce a classification scheme for creative unplugged learning activities. Then we discuss theoretical approaches to creativity and computational thinking. After that we present our international survey on creative unplugged activities in CS education.

Creative Unplugged Learning Activities

A creative task challenges creativity. In the context of Computational Thinking, creative skills may be used for finding and representing a problem (that can be solved by an algorithm), an algorithm (solving a problem) or a representation (visualisation) of an algorithm. An unplugged activity on computer science is an activity without computer. There are wonderful unplugged activities on computational thinking (Bell, McKenzie, Witten, Fellows, & Adams, 2015). But some of them are not creative. They are often like puzzles with one correct solution. On the other hand, creative tasks have *many* correct solutions and they always lead to a product that can be presented to an audience. We distinguish between four types of creative unplugged tasks related to computational thinking:

Type 1: Create an algorithm solving a given problem and present it without a computer. Here, the challenge is to find an algorithm with appropriate commands and data representations.

Example: Write a description how to walk from the entrance of this building to a certain room that can be understood by a person, who has never been in this building before.

Example: Transmit a sequence of zeros and ones on the school yard from A to B via intermediate stations using body language.

Type 2: Create to a given algorithm or informatics concept a new situation, in which this algorithm or concept can be used as well and present it. In this case, the creative challenge to develop a story, a problem from a different context, that can be solved with the algorithm. The student must understand the given algorithm to be able to check whether the story is appropriate.

Example: Describe a situation in which binary search is useful. Present the situation in a play using things from this room.

Example: Find situation, which can be modelled by a list of lists. Draw a picture illustrating this situation.

Type 3: Create an algorithm with certain structural properties (like loops, recursion, functions calls etc.) and present it in some way. Here, the creative challenge is both to find a problem and an algorithm. This type of task is solved by teachers that create illustrating examples for programming concepts.

Example: Here is the definition of a function with one parameter in Python style:

```
def make_a_ball(thing):  
    open your hand  
    put the thing into your hand.  
    while not thing has the shape of a ball:  
        move your fingers  
    return the thing
```

```
# main program  
make_a_ball(paper sheet)
```

Create an algorithm in Python style and write it down. The program text should contain the definition of a function with two parameters and a function call. Use things from this room as parameters.

Type 4: Create a visualisation for a given algorithm or concept of computer science and present it. Here, the creative challenge is to find a representation that the audience can understand. The students can use different materials and expressive means: drawing, role play with dialogs and props, pantomime, Lego etc. This kind of creative tasks solve designers who develop illustrations for a textbook.

Example: There is a collection of cards with informatics concepts. Each individual student or each team draws a card and visualizes the concept through a mime.

2. Creativity and Constructionism

The Creative System

Mihaly Csikszentmihalyi (Csikszentmihalyi, 2013) points out that »Creativity is a central source of meaning in our lives« (p.8). People feel excited, joyful and »fully living«, when they are creative.

A person is creative, when she or he produces an idea or artefact that is new. Thus, creative persons change our culture by adding new elements. According to Csikszentmihalyi's system model, creativity is not an individual ability but a system consisting of three parts (p.27 ff):

- A domain, that is a system of symbols and rules (like mathematics, music, computer science)
- A field that includes all persons that act as gatekeepers to the domain. They decide, whether an idea or product is accepted as new part of the domain.
- An individual person, that is creative and creates new elements within the domain

The level of creativity is not only depended on a person, but different domains offer different opportunities to be creative. Domains with clear structure, high centrality within the culture and good accessibility help to develop creativity.

How can a field (persons) affect creativity?

- A field can be reactive or proactive. Proactive fields stimulate novelty. In school life, science fairs and science slams promote novelty.
- Narrow or broad filter when selecting novelty
- A field can be connected well to the rest of the social system and channel support into its own domain.

Psychology of Creativity

Margarete A. Boden (Boden, 2007) distinguishes historical creativity (H-creativity) and psychological creativity (P-creativity). H-creativity takes place, when people come up with ideas that are new and have never been shared in the history of mankind. P-creativity happens, when a person comes up with an idea that is subjectively new to this person, but has been shared by someone else before.

According to Boden there are three types of P-creativity: combinational, exploratory, and transformational creativity.

Combinational creativity involves the generation of unfamiliar (statistical unusual) combinations of familiar ideas. A typical example is a visual collage, made of found photos. Visualisations of CS concepts (using body language, Lego, images) include combinational creativity. However, the CS concepts might be less familiar than the elements that are used for visualisation. It also might be that the result of the process is an artefact of value and includes many design ideas.

“In exploratory creativity, the existing stylistic rules or conventions are used to generate novel structures (ideas), whose possibility may or may not have been realised before the exploration took place.” (Boden, 2007). In CS, exploratory creativity may happen, when students experiment and explore a programming language and discover new programming techniques that way.

Transformational creativity is the creation of ideas people had previously thought impossible and which they still find counter intuitive. An example from the history of art is starting painting abstract pictures, where previously the paintings had all been representational. In transformational creativity the new idea is so radical that it is difficult to understand for the social environment.

Creative unplugged activities in the classroom clearly focus on combinational creativity. It is not forbidden that very bright high school students develop exploratory or even transformational creativity, but such events are beyond the scope of general education.

In contrast to Csikszentmihalyi's system theory, Boden's concept of P-creativity is subjectivist and ignores the social aspect of novelty completely. An idea is new, just if the subject has never thought about it and never seen it. There are two implications: P-creativity might be an illusion and P-creativity is dependent on how experienced a person is.

P-creativity might be an illusion, because people are not completely aware of their experience. Jenny might have seen a solution to a problem and forgotten this later. But traces of the knowledge are still in the sub consciousness. When she is in a similar situation later and the solution pops in her mind, she just thinks it was *her* idea, but in fact she had taken it from someone else. Someone who is a novice in a domain (like children playing with Lego for the first time) probably discovers more frequently subjectively new things than an experienced person.

Gerd E. Schäfer (Schäfer, 2001) points out that young children are “necessarily creative” because almost everything in their environment is new to them. They reconstruct the world in their mind by observing and experimenting and discover (or create) new knowledge all the time. Each day is full of surprises.

Creativity in the Classroom

Local Domains and Local Fields

Csikszentmihalyi's theory is based on interviews with exceptionally creative individuals from science, art and business. They know domains and fields very well. Creative scientists, working in research labs and universities, are familiar with relevant literature and attend international conferences. Obviously, the perspective of high school students is rather reduced. However, the classroom situation can be considered as a micro world representing the society in a small scale. The parts of domains the students are familiar with are defined by the school curriculum, textbooks and learning media in the WWW. We call this environment a “local domain”. All materials that are relevant in school life are monitored by universities and government administration, which are connected to international educational standards in some way. Therefore, the local domain of computer science at an individual school reflects the global domain of computer science. There is one fundamental difference between the local and the global

domain related to persistency. The local domain might change rather quickly, whereas the global domain is rather stable and expands relatively slowly.

When there are bright students and a supporting teacher in a CS class, in one year the local domain might expand quickly. Students do unusual projects and share them in some way with the local community. Examples of such sharing are posters in the hall of the science department or artefacts in a school exhibition. When these students or the teacher leave the school, this additional knowledge might get lost. The next age cohort of students might be less interested in CS and the local CS domain shrinks again.

In addition to the global field of CS, there is a local field of CS at each high school teaching this subject. The local field mainly consists of CS teachers. They decide whether a student's work is "creative" in the meaning of "something surprising" or "unexpected". Many teachers that have filled our questionnaire have stated that "creativity" is an important criterion for quality.

Also, students may be members of the field. They are experts and have specific knowledge about the (local) domain of CS and they act as gatekeepers. There are several filtering systems and gatekeeping activities in school life that involve the students:

Programming teams must decide on design issues. "Which idea is new and should be implemented in this project?"

Scratch projects which are published in a studio may be commented by classmates. A comment might include a statement on "how creative" a solution is.

The local field may be influenced by parents and students that are not in CS classes. They give a feedback on work, shown in private, published in the school or presented in public exhibitions and science slams. They are not part of the field because they are not familiar with the domain and because they are not directly part of a filtering system.

The Domain Problem of Creative Computer Science

When creating a program, the programmer is not creative in the local domain of computer science unless she or he develops a new programming technique, which is not mentioned in the textbook or curriculum. However, she or he might have found this »new« technique in the internet. This would also be a creative act related to the local domain, if this novelty is shared and accepted by the local field. Although the idea is not new within the global domain, it is a local novelty and the local domain is extended by this idea.

A software project might be an act of creativity in a different (local) domain if the project implements a new idea in this domain. Example:

Students have learned how to write interactive programs according to the simple structure »input-process data – output«. The creative task is: "Develop an interactive program that can be used to calculate the concentration of methane in a room or building. "

A version of this task which can be solved without a computer would be: "Develop an algorithm that can be used to calculate the concentration of methane in a room or building and present this algorithm in some way."

In this case the creative challenge is a) to find an explicit task within the context »concentration, gases, methane« and b) to find a solution for this task.

a) The context is from the domain »high school chemistry«. The students have to solve problems within this domain, when designing the task which can be solved by an algorithm:

- Which are typical rooms that might be contaminated by methane (e.g. school lab with Bunsen burners)? Which situation is interesting and relevant (e.g. flames of Bunsen burners are extinguished and gas is emitted)?
- What kind of data are accessible and might be used for input (number of Bunsen burners, amount of gas that is emitted from a Bunsen burner per second, Volume of the room)?

b) In order to create the algorithm, they have to find out how to calculate the concentration of methane from the input data. This includes designing automatized dialogs using proper language and units.

This example illustrates that the application of computational thinking (in this case algorithmic thinking) may lead to creativity in other domains.

Unplugged Creative Tasks on Computational Thinking - Building Bridges Between Domains

Creative tasks without a computer might be creative (in Csikszentmihalyi's system model), if the symbol system of the local domain is extended by fresh ideas how to represent a CS concept.

When students visualize a recursive algorithm in a role play they do not create a new algorithm. The novelty is new metaphorical representations for calling a function, passing parameters, processing data and so on. The task "create a role play" encourages to construct connections between domains. This goes well if the students are experienced in role plays. Building connections between concepts from different domains (from a familiar source domain to a more unfamiliar target domain) is called metaphoric thinking. New metaphors for CS concepts can be considered as an extension of the local domain of CS, since affects the way how to understand these concepts and how to use them. Consider an example from a different domain: The number line, visualising the set of real numbers, is part of mathematics.

New examples for algorithmic idea can be considered as an enrichment of the domain of CS. Each new example extends the already known area of application of the algorithm. This is also part of CS knowledge. Abstraction is part of computational thinking.

Skills and Attributes of Creative Persons

According to Csikszentmihalyi, being creative is like being involved in a traffic accident. One may have a disposition to be creative but every act of creativity depends on several factors. One cannot say that a person starts a creative process. Creative people interviewed by Csikszentmihalyi said they were just lucky. They were at the right time at the right place.

Csikszentmihalyi mentions attributes and skills that increase the chance of a person to be creative:

- Creative people must internalize the creative system: know the domain very well.
- They are curious and are interested in a domain.
- Creative persons are playful and have discipline (both is important), they need endurance and self-confidence to finish a project.
- Creative people can alternate between fantasy and imagination and a sense of reality.

The Enjoyment of Being Creative

Csikszentmihalyi points out that it is joyful to be creative. The optimal experience is called flow, the feeling, when things are "going well as an almost automatic, effortless, yet highly focused state of consciousness" (Csikszentmihalyi, 2013). The joy of being creative – the joy of novelty – is determined by the human genes. Novelty is important for the development of a culture. It increases the fitness to adapt to changing environments. Creative unplugged tasks on computational thinking are an educational method to motivate students to learn and reflect CS concepts.

Creative persons interviewed by Mihaly Csikszentmihalyi reported that situations that are experienced as enjoyable (flow) contain these elements:

- There are clear goals in every step of the creation process. In flow we know exactly what to do.
- There is immediate feedback, which makes it easy to distinguish between good and bad ideas.
- There is a balance between challenge and skills.
- Action and awareness are merged. (I am thinking about what I am doing.)
- Distractions are excluded.
- There is no worry of failure, the subject is too involved to be concerned
- The self-consciousness disappears.
- The sense of time becomes distorted.

- The activity becomes autotelic. It is joyful per se.

Teachers designing creative tasks should do everything to keep the joy of the creation situation.

How to Encourage Creativity?

An important element of CS classroom education is designed activities. They consist of a task and some material which is used to solve the task. “Unplugged” activities use tangible material like blocks or any material that is ad hoc available, pencil and paper or just own body, but not computers. The charm of “unplugged” activities is often the contrast between the material and the CS concepts that are elaborated.

In research literature and educational programs one can find much advice how to encourage and foster creativity. In this section we try to relate this advice to creative unplugged activities, considering

- activity design, including the task and the used material,
- scaffolding during the performance of the activity
- presentation of the results including rules for feedback

Activity Design

Boden mentions different ways to encourage creativity depending on the type of creativity which is intended to be developed. Combinational creativity can be encouraged by enlarging the variety of concepts in a person’s mind. The more concepts I know the more “unusual combinations” I can find. Additionally, a person can practice finding new associations between concepts and learn to judge the value and novelty of ideas. (This is internalizing the field in Csikszentmihalyi’s system theory.)

In task design, a simple method to enlarge the repertoire of concepts is to let the students work in teams consisting of persons with different interests and knowledge. When people collaborate in diverse groups, new and surprising ideas may evolve easier.

Secondly, the task should be surprising and raise curiosity and interest. Csikszentmihalyi suggests to cultivate curiosity and interest. If unusual results are expected, the task should be unusual too.

Performance

Csikszentmihalyi suggests to cultivate flow in everyday life. People should wake up with a specific goal they look forward to. Everything we do we should do well and joyful. An atmosphere of strength and joy should be maintained during the performance of a creative activity. The learning environment should make it easy to focus and prevent distractions. An empty school yard might be a good environment for some creative activities. Classrooms should be designed to support the creative process, like the laboratory in Dalton education.

Scaffolding during the process should support and encourage the following operations (Csikszentmihalyi, 2013):

- clarify, analyse and re-define the problem or question to uncover new ways of looking at it,
- try to find connections between seemingly unrelated subject matter,
- challenge established wisdom by asking: how would I improve this?
- recognise alternative possibilities,
- look at things from different perspectives.

Presentation

Boden (Boden, 2007) points out that students should not be discouraged by dismissing new and surprising ideas as mistakes. Creative persons need self-confidence. The logical consequence is that encouraging creative task should always lead to a success. There is no right and wrong. Imagine the task “Create a role play visualising the execution of a recursive function.” After some time, the team has to come up with a role play and perform it. This role play must be a success, even if there is hardly a visible connection to recursion at first glimpse. The teacher must support a certain culture of reception, which is friendly and appreciating. The audience must appreciate that watching all the little details of

the play and reflecting and discussing their relation to the CS concept of recursion helps to get a deeper understanding.

During the presentation the field gets active. This is important for the creators since they must internalize the field and must learn to distinguish between good and bad ideas, new and old ideas. The presentation is a good opportunity to discuss novelty and values. Some ideas, developed in creative unplugged tasks might get documented in some way and extend the local domain of high school CS. The challenge for the school is to install opportunities for this documentation: science fairs, showcases, posters, wikis etc.

3. Computational Thinking and Creativity

In this section the view on creativity from computational thinking (CT) practitioners point of view is presented. First the concept of CT is briefly analysed. Next the creativity concept in CT domain is presented.

In 2006 Wing (Wing, 2006) presented the concept of CT: “ ... involves solving problems, designing systems, and understanding human behaviour by drawing on the concepts fundamental to computer science”. Based on this it can be concluded that CT involves three key components: algorithms, abstraction, and automation. However, the rice of this concept can be related to Seymour Papert by introducing it in the context of suggesting an alternative, computationally-based mathematics education (Papert, 1996). Hence the researchers are very interested in CT approach and its application in education. However, it is still at an early stage of maturity (Lockwood, Mooney, 2017) and the steady definition of CT is not provided (Voogt et al., 2015). In order to explore CT approach (Voogt et al., 2015) provided the discussion about the definition and core concepts of CT only in the Computer Science domain (excluding other domains). They focus on finding similarities and relationships in the discussions about CT. Similarly, the analysis made in (Kalelioglu et al, 2016) provided the word cloud of CT definitions used in analysed papers. The generated ‘Wordle’ by Kalelioglu and others was based on the definitions provided by analysed researchers and not included some core concepts of CT.

CT definition challenge was analysed also in others domains, such as mathematics and science by (Weintrop et al., 2016). They reviewed the literature on CT and proposed a definition of CT in the form of a taxonomy consisting of four main categories: data practices, modelling and simulation practices, computational problem solving practices, and systems thinking practices. Additionally, the set of ten CT skills were proposed. This set was developed based on literature review on CT with a focus on applications to mathematics and science.

Some researchers focus on CT skills in order to explain CT (Curzon et al, 2014; Atmatzidou & Demetriadis, 2016) or very similar named concepts (Catlin & Woollard, 2014), such us abstraction, decomposition, generalization.

Others interpret CT based on its elements. Such us (Grover & Pea, 2013) identified nine elements: Abstractions and pattern generalizations (including models and simulations); Systematic processing of information; Symbol systems and representations; Algorithmic notions of flow of control; Structured problem decomposition (modularizing); Iterative, recursive, and parallel thinking; Conditional logic; Efficiency and performance constraints; Debugging and systematic error detection. Additionally, others argue that CT is an activity, often associated with, but not limited to, problem solving (CSTA & ISTE, 2011; Beecher, 2017, p. 8; Haseski et al., 2018).

All these concepts (skills, elements) are widely used by researchers and educators with the aim to teach computational skills across the different curriculum. In the book by Williams (Williams, 2017) ISTE developed concepts are used in K-5 curriculum. Author argues that CT can be incorporated into any subject. Also, she gave the case studies with Bee-Bots, Code.org, Scratch and ARIS and presented how CT concepts can be applied to these tools in detail.

CT can be seen as a fundamental skill for everyone, not just for computer scientists. It is applicable in either a computerised or unplugged problem-solving process. CT has the potential for application in a wide range of disciplines as the creative learning arrangements. Two main strategies are used for CT skills development: unplugged activities (activities that

involve logic games, cards, puzzles, strings or physical movements to get in touch with computer science concepts such as algorithms, data transmission or data representation) and computerized activities (such as, programming in arrow-based visual environments, programming in block-based visual environments, using textual programming languages, programming that is connected with the physical world) (Moreno-León et al, 2018).

In this paper CT is defined as the skills of being able to develop creative solutions for the problem with an algorithmic approach by handling a problem by the individuals that could establish healthy communication in a cooperative environment (Korkmaz et al., 2017).

CT is a set of skills that help to set up a problem in such a way that a computer can help you solve it (Krauss & Prottsman, 2017, p.47). However, computers are not necessarily. Solutions could be represented as computational steps and algorithms (Aho, 2012). Main CT skills are as follows:

- Abstraction is the process of creating something simple from something complicated, by leaving out the irrelevant details, finding the relevant patterns, and separating ideas from tangible details (Atmatzidou & Demetriadis, 2016);
- Algorithmic thinking is the ability to understand, execute, evaluate, and create computational procedures (Lamanga, 2015);
- Decomposition is the process of breaking down problems into smaller parts that may be more easily solved (Atmatzidou & Demetriadis, 2016);
- Generalisation is transferring a problem-solving process to a wide variety of problems (Atmatzidou & Demetriadis, 2016);
- Evaluation is the ability systematically (through criteria and heuristics) make substantiated value judgements (Catlin & Woollard, 2014).

Moreover, researchers agree that CT is overlapping with many aspects of 21st century skills such as creativity, critical thinking, and problem-solving (Lye, Koh, 2014). According to Korkmaz et al. (2017), CT is the extension of the problem solving skills of a person and the development of the creativity and critical thinking skills of the people by re-focusing. Thus creativity plays an important role in CT approach (Korkmaz, 2017). CT mainly focuses on the creation of algorithms by using problem solving skills. Moreover, most of CT parts involve creativity, such as algorithmic design, generalisation and pattern matching (e.g., needs big creative leaps to see the links between apparently different situations), evaluation (e.g. needs creativity in coming up with logical arguments or ways to explore situations) (Curzon, McOwan, 2017, p. 209). Creativity in abstraction occur, e. g. when students invent a new representation for recursion - say in a role play. Additionally, Curzon and McOwan (2017) argue that most creative ideas come from groups not from individuals. Moreover, CT foster creativity in the classroom due it allowances for student to move from technologies' consumers to developers in order to benefit society, also, creativity can be augmented by CT (Mishra et al., 2013). The College Board (2017) developed the CT framework for a Computer Science Principles course for high schools in the USA. According to it computing is a creative discipline and student are engaged in creative aspects of computing by designing and developing computational artefacts, also, by applying computing techniques to solve problems (p. 9), such as, by employing non-traditional, non-prescribed techniques, the use of novel combinations of artefacts, tools and techniques, as well as, exploration of personal curiosities (p.11). Students could be developing artefacts for creative expressions and those can reflect personal ideas or interests (p. 12). In such way, computing enables students to use creative development processes.

Many creative unplugged activities examples are presented by Curzon and others (2014). Let's take, for example, such a type of creative unplugged activity that aims to create an algorithm for solving a given task and present it without a computer. The task is to help person with locked-in syndrome (total paralysis due to a stroke) to communicate. The solution could be the communication by blinking: one person says the letters of the alphabet, and the other blinks when they get to the letter they wish to communicate. Thus the creative algorithmic thinking is demonstrated when the audience are asked to think of improvements, the details of this algorithm that need further thought to make it work. As well as, generalisation needs the creativity from audience for suggesting things like predictive texting and

frequency analysis as improvements. Likewise, evaluation needs creativity to discuss an algorithm's functionality, its performance and its usability.

Creativity has its place in many areas, such as art, economy, psychology and science. In schools it is associated with art, music, and writing classes (DeSchryver, Yadav, 2015). And it is differently defined in various fields. Moreover, such a complex concept is difficult to measure, although many approaches to the measurement of creativity are developed (Jackson et al., 2012). Creativity in CS field usually aims at producing new mechanisms, both hardware and software, that can provide solutions to practical problems. However, creativity involves the same kinds of cognitive processes that generate answers in computing as well as in the natural sciences (Saunders & Thagard, 2005). Moreover, in CT, creativity could be seen as an ability by applying imagination to develop a physical object or some mental or emotional construct (Korkmaz et al., 2017) that is judged to be novel and also to be appropriate, useful, or valuable by a suitably knowledgeable social group (DeSchryver, Yadav, 2015).

4. Questionnaire

In order to get the opinions from the computer science educators at all levels of educational system about the creative unplugged activities in the classrooms and also their opinion about the categories model of creative unplugged activities, an online survey was conducted. First the questionnaire in English language was prepared with questions about the four categories of the creative unplugged tasks and questions about the use of such type of tasks in their daily practice in the classrooms. Then the questionnaire was translated to the national languages and distributed through different national channels to the computer science educators. In the table 1 is the distribution of the educators based on their country of origin and total number of answers which is 360.

Table 1. Country of origin of the respondents

Country	Number of respondents
Czech republic	133
Lithuania	8
Slovenia	25
Germany	153
Japan	14
Other countries	27
Together	360

Types of Creative Unplugged Activities in Computer Science

The questionnaire first presented definitions and examples of four different types of creative unplugged activities and asked the respondents how clear the definitions were. The responses were on scale from unclear (1) to clear (5). In table 2 descriptive statistics are presented. It can be observed, that for educators' types 1 and 3 are clearer and more understandable than type 2 and 4. It seems that educators feel more confident with activities that ask for specific structural features than with more general concepts. One of the reasons might be that activities and tasks that fit in type 3 category are more frequently used in CS teaching.

Table 2. Clarity of the definitions of the four types of creative unplugged activities

Type of unplugged creative learning activity	Degree of clarity	
	Mean	Std. Deviation
<i>Type 1: Create an algorithm</i> Invent an algorithm that solves a given task and present it without a computer.	3.78	1.16
<i>Type 2: Find an example</i> For a computer science algorithm or concept, find a new situation in which to apply this algorithm or concept.	3.24	1.27
<i>Type 3: Find an example algorithm (more open form)</i> Find an algorithm that has certain given structural features (e.g., loops, recursion, function calls) and represent it in some way.	3.77	1.12
<i>Type 4: Create a visualization</i> Invent a visualization for an algorithm or concept of computer science.	3.33	1.25

Next we asked them if they know about creative unplugged activities that do not fit in presented model. 30% of the teachers claimed that they know other activities that do not fit. We asked them if they can provide an example or comment.

107 from 360 respondents (more than $\frac{1}{4}$) claimed that they know other creative informatic activities without computer out of our four categories. 38 of them described one or more activities, some respondents gave general comments.

Only about one third from these 38 free text responses can be considered as examples for creative unplugged activities according to the given definition. Most of the presented activities were unplugged but not creative. They were typical problem-solving activities with only one possible solution (e.g. ordering data, comparison of algorithms). Some activities were not from CS (e.g. painting). In some cases, it was impossible to decide whether this activity is creative or not, for example from the response "puzzles – Sudoku" one cannot not deduce whether the respondent meant solving sudoku or creating a new one.

Some of the responses described concrete creative activities which can be related to one of the four types e.g. "Representing the TCP/IP protocol stack through role play using envelopes as packets" which is of type 4 (create a visualization).

However, a few answers from the 38 responses were valid creative unplugged activities that do not fit to our model as it is. One teacher suggested this task: "Create a protocol for bidirectional communication via one wire". Since a protocol is not an algorithm, this activity is not of type 1 (create an algorithm). Activities like this would be covered by the model, if we slightly extend the scope of type 1, like: "Create an algorithm *or part of an algorithm...*" Part of an algorithm could be a protocol, a data structure, a class model etc.

Experiences with Creative Unplugged Activities in Computer Science

Next we asked the respondents about their experiences with creative tasks. On the question "How often have you spent the last 12 months doing creative tasks without a computer in computer science education or a computer science course?", 50% of the respondents said they used them once or twice, 27% used it more than twice and 23% said they never done them in the classroom. If we look at the results of the next question about the reaction in the classroom where nearly 60% answered that the reaction is positive and combine them with the result from this question where 50% of educators use unplugged creative tasks use them only once or twice in one year, the situation is rather disappointing.

Maybe one reason lies in the fact, that there aren't that many available prepared creative unplugged activities that educators could use or educators don't have time to use them in classrooms.

On the question "If you have engaged in creative tasks, what has been the reaction of the students?" we got the following results presented in Table 3. Just 10% of all answers is negative or with no reaction, which is good. Fact that almost 60% educators noticed positive reaction must be used to further push unplugged activities in the classrooms.

Table 3. Student's reaction to creative unplugged activities

Student's reaction	Percentage
Positive	13.6%
Rather positive	45.8%
neutral	12.7%
Rather negative	6.6%
negative	0.5%
no reaction	3.6%
didn't answer	16.9%

Next, we asked educators, if they could tell us why and how they used creative unplugged tasks. Respondents could check multiple options. The results are presented in table 4. Educators are using creative unplugged tasks for the activation or as suggestion to think about CS concepts. Note that more than one third of educators use unplugged activities also as entertainment element in the classrooms.

Table 4. Educators' intentions for using creative unplugged activities

Intention	Percentage
As an activating entry into a new topic.	51%
As a suggestion to think about certain abstract concepts.	47%
To promote creativity.	26%
As training, e.g. to practice the correct use of technical terms.	25%
As a diagnostic tool to identify a prior understanding of a topic or misconceptions.	16%
As an entertainment element to relax a course.	38%

The last question in this section was "Creative solutions differ from examples previously discussed in class. They are novel and original. To what extent do you assess the aspect of creativity in the work of

your students?". The responses are in Table 5 and show that almost all educators in some way assess creativity of their students' work. Nevertheless, more than 50% don't assess creativity as official criteria. One of the reasons lie in the fact that educators must comply to national rules for assessment (e.g. in Slovenia it is very complicated to assess creativity) and educators sometimes avoid the risks with just not assess creativity. Other reason lies in the fact that is already hard to describe and define creativity in CS and therefore educators don't feel confident, that they know how to assess creativity of their students.

Table 5. Relevance of assessing creativity

Assessment of Creativity	Percentage
I have practically never assessed the creativity of my students' work.	8%
I have occasionally evaluated the creativity of a solution, but creativity has not been an official quality criterion for my assessment of the work of students.	48%
Creativity is one of my official quality criteria for evaluating the work of students. However, creativity has only a subordinate rank over other criteria (such as correctness of the solution).	35%
Creativity is one of my most important official quality criteria for evaluating the work of students. Among my evaluation criteria, creativity has a similar importance to that of correctness.	9%

Educational Potential of Creative Tasks

In this section we asked the teachers about their opinion on the educational potential of creative tasks. More precisely, we asked them about specific potential and which type of creative tasks can be used instead. More answers could be selected at each question. Results for different purposes are in Table 6. As a replacement for programming tasks most educators think that activities that fit into Type 1 (Create an algorithm) are most useful and appropriate. The reason for such results lies in the fact, that Type 1 activities are in conceptual way very similar to the programming tasks, only allowed tools are different. For other three type it's not that easy to see, how such activities can replace programming tasks. It's interesting to notice, that educators don't feel that Type 3 activities can replace programming tasks, although such activities develop understanding of some concept similar as programming tasks do. Most of educators think that all types of creative tasks, except Type 3 tasks, are suitable for use in the classrooms as enrichment. Analysing the answers more in depth, we see, that almost 90% of respondents that checked Type 3 also checked tasks of Type 1 and Type 2. Educators think that tasks of Type 2 are most appropriate to use them as encouragement, although other types of activities are also used in classrooms. We can conclude that educators think that finding a new situation encourages students to think about CS and also helps students to use their CS knowledge in different areas of their life. Most educators think that tasks of Type 1 and 2 help students develop such skills that can be used in other scientific fields. The reason lies in the fact, that educators think that first two types help to develop computational thinking which can be later applied in various fields.

Table 6. Educational potential of creative unplugged activities

Educational Potential	Type 1 <i>Create an algorithm</i>	Type 2 <i>Find an example</i>	Type 3 <i>Example algorithm</i>	Type 4 <i>Visualisation</i>
A replacement for programming tasks providing a comparable learning experience in less time.	72%	50%	44%	46%
An enrichment to a lesson or lecture to make the content more relevant and attractive to the participants.	68%	58%	39%	62%
As a encouragement for students to think about computer science concepts.	61%	67%	54%	60%
As a help to develop transferable skills.	62%	71%	49%	56%

Professional Experience in Informatics Education

In last section we asked them about their background in Computer science education. Results are in Table 7; more than answer was possible. We can observe that our respondents were from all levels of education and some of them had experience with authoring a book or being involved in research activities. One quarter of respondents were also teacher trainers. We can conclude that such spread of experiences gives this survey reliability and important view in how educators think and view creative unplugged tasks in CS education.

Table 7. Respondents' professional experience

Experience	Percentage
Teaching computer science-related content at a primary school	28 %
Teaching computer science related content at a secondary school	88 %
Teaching computer science related content at a university	14 %
Authoring or co-authoring a textbook in the field of computer science	12 %
Research in the field of computer science	13 %
Teacher training in the field of computer science	27 %

5. Discussion

The survey has shown that most teachers use creative unplugged activities in CS classes, but not very frequently. One of the reasons we can conclude from the responses is that computer science educators consider unplugged creative tasks as not very relevant, since respondents suggested that they usually

use them in the initial parts of a lessons, as introduction to some topic or as initial phase of problem solving tasks. They also suggested that these activities could be used for group work.

From curricula point of view, teachers saw application of these tasks as an introduction to understanding how computer works, in media education and painting art activities and in creating criteria by ordering data. This might indicate that some teachers connect creativity more to using ICT than to computer science.

The questionnaire explained four types of creative unplugged task and gave examples. For many teachers this might be the first contact with this area and they might have problems to understand this short introduction. This could explain some strange answers.

Teachers gave some attitudes to using creative informatics non-computer tasks. Some attitudes were against leaving programming and coding, some of them expressed conviction that they could replace programming in some parts of curricula (without specification) and some of them apologize respondents for not doing enough for it (because of lack of time in compulsory education that leads teachers to teach this as extracurricular activities, e.g children clubs).

A possible reason that teachers do not use creative unplugged tasks in CS education so much might be that focussing on computational thinking is not established so well in CS education. Many CS classes are mainly programming classes or teach how to use ICT as a tool.

How to assess creative work seems to be unclear for many teachers, which might partly be caused by the fact that the term “creativity” doesn’t have one single definition. On other hand assessing creativity can be very subjective (if the teacher dominates the field) and teachers don't feel confident enough to assess what is creative.

6. Conclusion

We need teachers who are able to support creativity. Designing and conducting creative unplugged activities should be part of teacher training. The question is, how to do this. We need concepts for workshops inspiring teachers and future teachers.

Students can only be creative in domains they are familiar with. (For example, visualizing a CS concept by mime works, if the students are experienced with mime.) Therefore, teachers creating new creative unplugged arrangements should involve their students.

The four types of creative unplugged activities that we have suggested, are just a start. In future research we can use the open text answers of the questionnaire to improve and extend the classification scheme.

References

- Bell T., et al. (2015): CS Unplugged, Computer Science Without a Computer, https://classic.csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf
- Futschek, G, Moschitz, J.: Developing algorithmic thinking by inventing and playing algorithms. Proceedings of the Constructionism 2010, 1/10 (2010)
- Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf
- Weigend, M. (2017) Smartwalk: computer science on the schoolyard. IFIP World Conference on Computers in Education. Springer, Cham.
- Wing, J. (2006). Computational Thinking. Communications of the ACM, 49(3), 33-35.
- Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). Computers in Human Behavior, 72, 558-569.
- Csikszentmihalyi, M. (1996). Creativity. Flow and the Psychology of Discovery and Invention

- Cropley, A. J. (1997). Fostering creativity in the classroom: General principles. In M. A. Runco (Ed.), *The creativity research handbook* (Vol. One, pp. 83e114). Cresskill, New Jersey: Hampton Press.
- CAS computational thinking - A Guide for teachers (2015). Guidance for both primary and secondary school teachers aimed at developing a shared understanding of computational thinking (Available at: <https://community.computingschool.org.uk/resources/2324/single>).
- Krauss, J. & Prottsman, K. (2017). *Computational Thinking and Coding for Every Student. The Teacher's Getting-Started Guide*. Corwin Press Inc.
- Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661-670.
- Lamagna, E. A. (2015). Algorithmic thinking unplugged. *Journal of Computing Sciences in Colleges*, 30(6), 45-52.
- Catlin, D., & Woollard, J. (2014, July). Educational robots and computational thinking. In *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education* (pp. 144-151).
- Papert, S. (2000) What's the big idea? Toward a pedagogy of idea power. *IBM SYSTEMS JOURNAL*, Vol. 39, pp. 720-729.
- Boden, M. (2007) *How Creativity Works*, published by Creativity East Midlands for the Creativity: Innovation and Industry conference, 6th December 2007.
- Romeike, R., Knobelsdorf M. (2008) Creativity as a Pathway to Computer Science. *ITiCSE'08*, June 30-July 2, 2008, Madrid, Spain, pp. 286-290
- Schäfer, G. E. (2001): Prozesse frühkindlicher Bildung [educational development in early childhood], Cologne, Germany. URL: https://www.hf.uni-koeln.de/data/eso/File/Schaefer/Prozesse_Fruehkindlicher_Bildung_Duplex.pdf
- Saunders, D., & Thagard, P. (2005). Creativity in computer science. *Creativity across domains: Faces of the muse*, 153-167.
- Jackson, L. A., Witt, E. A., Games, A. I., Fitzgerald, H. E., Von Eye, A., & Zhao, Y. (2012). Information technology use and creativity: Findings from the Children and Technology Project. *Computers in human behavior*, 28(2), 370-376.
- Mishra, P., Yadav, A., & Deep-Play Research Group. (2013). Rethinking technology & creativity in the 21st century. *TechTrends*, 57(3), 10-14.
- Papert, S., (1996) An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1), 95-123.
- Lockwood, J., & Mooney, A. (2017) *Computational Thinking in Education: Where does it fit? A systematic literary review*. arXiv preprint arXiv:1703.07659.
- Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715-728.
- Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, 4(3), 583.
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.
- Curzon, P., McOwan, P. W. (2017) *The Power of Computational Thinking: Games, Magic and Puzzles to Help You Become a Computational Thinker*. World Scientific.

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014) Introducing teachers to computational thinking using unplugged storytelling. In ACM Proceedings of the 9th workshop in primary and secondary computing education, 89-92.

Atmatzidou, S., & Demetriadis, S. (2016) Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661-670.

Grover, S., & Pea, R. (2013) Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.

ISTE, CSTA. (2011) Computational Thinking in K–12 Education leadership toolkit.

Beecher, K. (2017) *Computational Thinking: A Beginner's Guide to Problem-Solving and Programming*.

Haseski, H. I., Ilic, U., & Tugtekin, U. (2018) Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. *International Education Studies*, 11(4), 29.

Williams, H., (2017). *No Fear Coding: Computational Thinking Across the Curriculum (CODEK5)*, ISTE.

Moreno-León, J., Román-González, M., & Robles, G. (2018, April). On computational thinking as a universal skill: A review of the latest research on this ability. In *Global Engineering Education Conference (EDUCON), 2018 IEEE* (pp. 1684-1689). IEEE.

Lamagna, E. A. (2015). Algorithmic thinking unplugged. *Journal of Computing Sciences in Colleges*, 30(6), 45-52.

Lye, S. Y., & Koh, J. H. L. (2014) Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, 41, 51-61.

DeSchryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. *Journal of Technology and Teacher Education*, 23(3), 411-431.

College Board (2017) *AP Computer Science Principles. Course and Exam Description*. College Board, NY.

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, 55, 832–835.

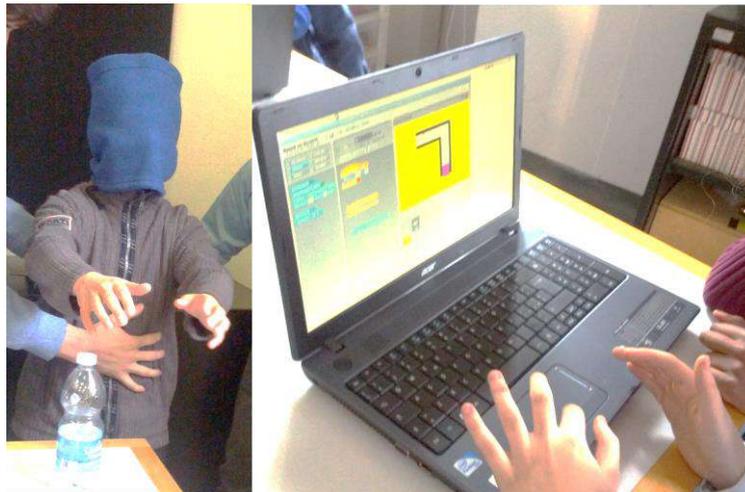
WG6: Learning to Program in a Constructionist Way

Mattia Monga, *mattia.monga@unimi.it*
Università degli Studi di Milano, Milan, Italy

Michael Lodi, *michael.lodi@unibo.it*
Alma Mater Studiorum - Università di Bologna & INRIA Focus, Italy

Dario Malchiodi, *dario.malchiodi@unimi.it*
Anna Morpurgo, *anna.morpurgo@unimi.it*
Università degli Studi di Milano, Milan, Italy

Bernadette Spieler, *bernadette.spieler@ist.tugraz.at*
Technische Universität Graz, Austria



Abstract

Although programming is often seen as a key element of constructionist approaches, the research on learning to program through a constructionist strategy is somewhat limited, mostly focusing on how to bring the abstract and formal nature of programming languages into “concrete” or even tangible objects, graspable even by children with limited abstraction power. However, in order to enable constructionism in programming several challenges must be addressed. One of the crucial difficulties for novice programmers is to understand the complex relationship between the program itself (the text of the code) and the actions that take place when the program is run by the interpreter. A good command of the notional machine is a necessary condition to build programming skills, as is recognizing how a relatively low number of abstract patterns can be applied to a potentially infinite spectrum of specific conditions. Programming languages and environments can either help or distract novices, thus the choice is not neutral and their characteristics should be analyzed carefully to foster a good learning context. The mastery of the notional machine, however, is just the beginning of the game: to develop a real competence one must be able to think about problems in a way suitable to automatic elaboration; to devise, analyse, and compare solutions, being able to adapt them to unexpected hurdles and needs. Moreover, it is important to learn to work productively in a team, in an “organized” way: agile methods seem based on common philosophical grounds with constructionism.

1. Introduction

Educators, generals, dieticians, psychologists, and parents program. Armies, students, and some societies are programmed. [Alan Perlis]

Constructionism [Papert and Harel 1991] is a strategy of education which has its roots in Piaget's constructivist theory of learning as an active process, in which people actively *construct* knowledge from their personal experience of the world. In general, students do not just receive pre-built ideas from teachers: they have to make them up by engaging themselves with problems, projects, and other people (instructors, but also peers). Papert's constructionism indeed emphasizes the importance of having personally-meaningful goals and "*public artifacts*" (not necessarily concrete ones: either "a sand castle on the beach or a theory of the universe" [Papert and Harel 1991]) that can be shared and discussed with others interested in the same (learning) enterprise [Resnick 1996]. This is sometimes summarized with the four P-words: Projects, Peers, Passion, Play and this motto indeed inspired successful educational initiatives such as the Scratch programming language [Resnick 2014].

1.1 Constructionism and programming

However, while programming is often seen as a key element of constructionist approaches (starting from Papert's LOGO, a programming language designed to enable the learning of geometry), the research on learning to program through a constructionist strategy is somewhat limited, mostly focusing on how to bring the abstract and formal nature of programming languages into "concrete" or even tangible objects, graspable even by children with limited abstraction power [Resnick et al. 2009; Horn and Jacob 2007; Hauswirth, Adamoli, and Azadmanesh 2017]. Notwithstanding this, constructionist ideas are floating around mainstream programming practice and they are even codified in some software engineering approaches: agile methods like eXtreme Programming [Beck and Andres 2004], for example, suggest several techniques that can be easily connected to the constructionist word of advice about discussing, sharing, and productively collaborating to successfully build knowledge together [Resnick 1996]; moreover the incremental and iterative process of creative thinking and learning [Resnick 2007] fits well with the agile preference to "responding to change over following a plan" [Beck et al. 2001].

The iterative process described by Resnick in [Resnick 2007] originated by observing how kindergarteners learn, and is now called "creative learning spiral" (Figure 1), that describes MIT's view on how to learn creatively [Resnick 2017]. When you learn by creating something (e.g. a computer program) you **imagine** what you want to do, **create** a project based on this idea, **play** with your creation, **share** your idea and your creation with others, **reflect** on the experience and feedback received from others, and all this leads you to **imagine** new ideas, new functionalities, new improvements for your project, or new projects. The process is iterated many times.



Figure 1: Creative learning spiral (Resnick 2017)

This spiral describes an iterative process, highly overlapping with iterative software development cycle (see 5.1).

2. What does it mean to learn programming?

The basic assumption/premise behind programming — *i.e.*, producing a precise description of how to carry out a task or to solve a problem — is that an *interpreter*, different from the producer of the description, can understand it and effectively carry out the task as described. There are thus two distinct but tightly tied aspects in programming:

1. the program itself (the text of the code),
2. the actions that take place when the program is run by the interpreter.

We thus need to know the interpreter in order to program, in particular we need to know:

- the set of basic actions it is able to perform,
- a language it is able to understand, with rules on how to compose basic actions,
- the relation between *syntax* and *semantics*, that is what actions it will perform given a description, and, conversely, how to describe a given sequence of actions so that it will perform them.

The first aspect, that is the program *source code*, is explicit, visible. The second one instead, that is the actions that take place when the program is run, is somewhat implicit, hidden in the execution time world, and not so immediate to grasp for novices. Moreover, this aspect is sometimes underestimated by both teachers and learners: teachers, as experts, give it for granted; learners tend to construct personal intuitive, not necessarily coherent, ideas of what will happen.

This dichotomy of programming — its static visible code and its implicit dynamics — emerges as a critical issue when learning to program, as shown by studies from different perspectives. To cite a few [Sorva 2013]:

- Phenomenography studies show how novice programmers tend to perceive programming as no more than the production of code, missing to relating instructions in the program to what happens when the program is executed.
- Studies on programming misconceptions point out how most of programming misconceptions have to do with aspects that are not readily visible in the code but are related to the execution time, both in term of what will happen and of what will not unless explicitly specified in the code.
- Threshold concept theory identifies program dynamics as a candidate threshold concept in programming as it has many of the features that characterise threshold concepts; among others: it is a troublesome barrier to student understanding, it transforms how the student perceives the subject, it marks a boundary between programmers and end users.

To help novice programmers take into account also the dynamic side of programming, the concept of *notional machine* has been proposed. A notional machine is a characterisation of the computer in its role as executor of programs in a particular language (or set of languages, or even a subset of a language) for didactic purposes. It thus gives a convenient description of the association syntax-semantics. The following learning outcomes should therefore be considered when teaching to program:

- the development by students of a perception of programming that does not reduce to production of code, but includes relating instructions to what will happen when the program is executed, and eventually comes to include producing applications for use and seeing it as a way to solve problems;
- the development of a mental model of a notional machine that allows them to make the association (static) syntax - (dynamic) semantics and to trace program execution correctly and coherently.

In particular this latter outcome goal will include the development of the following skills:

- given a program (typically one's own) and an observed behaviour:
 - identify when debugging is needed,
 - identify where a bug has occurred,
 - be able to correct the code;
- given a program and its specifications, be able to test it;
- understand that there can be multiple correct ways to program a solution.

If this is a crucial point in learning to write executable descriptions, however, programming is indeed a multifaceted competence, and the knowledge to construct and the skills to develop span over several dimensions, besides predicting concrete semantics of abstract descriptions given via programming languages:

- understanding general properties of automatic interpreters able to manipulate digital information;
- thinking about problems in a way suitable to automatic elaboration;
- devising, analysing, comparing solutions;
- adapting solutions to emerging hurdles and needs;
- organizing team work and productively eliciting, organizing, and sharing the abstract knowledge related to a software project.

2.1 Constructionism and learning to program

In some sense programming is intrinsically constructionist as it always involves the production of an artifact that can be shown and shared. Of course when teaching programming, this aspect can be stressed or attenuated.

Failure rates and dropout percentages in traditional programming courses and the urge to introduce programming early in school curricula have fostered new approaches to teaching programming, where this aspect has gained importance. Indeed the following points are given particular consideration:

- *motivation*: programming tasks should be engaging to keep pupils' motivation high;
- *syntax*: novices should be introduced first to the logical aspects of programming and only at a later stage to the syntax;
- *a constructivist approach*: the construction of knowledge is to be fostered, for example through *unplugged* activities that are more suitable to group work and shared meta-cognition;
- *constructionism*: the production of personal projects and artifacts must be encouraged.

In this perspective, for educational purposes visual programming languages (Section 3), have been developed and unplugged activities have been designed. In particular visual programming languages allow novices to concentrate on the logical aspects of programming without having to strive with unnatural textual syntactic rules. Moreover, they make it possible to realise small but meaningful projects, keeping students motivated, and support a constructionist approach where students are encouraged to develop and share their projects — video games, animated stories, or simulations of simple real world phenomena.

2.2 Computers Unplugged

Offline or unplugged programming activities were often used to explain important concepts or vocabulary to students without actually using a PC, laptop, or smartphone, e.g., x/y coordinates, the need for precise instructions for computers/robots, or variables and lists. Examples are to program a classmate like a robot, paint instructions, pack a rucksack, or send "broadcast messages" to colleagues.

Unplugged activities in small groups have become popular over the years to introduce basic computer science concepts in non vocational contexts. They offer a number of advantages:

inexpensive set up: they usually require very basic and inexpensive materials, so they can be easily proposed in different contexts;

no technological hurdles: they do not involve the use of technology, with which not all teachers are at ease;

a constructivist environment: indeed

- by manipulating real objects or dramatising processes, pupils can observe what happens, formulate hypotheses, validate them through experiments, i.e. develop a scientific approach to the construction of their knowledge;
- by working in group, pupils are encouraged to participate, share ideas, verbalise and uphold their deductions.

As the results of the activities, be they the execution of a procedure, the design of a solution or the construction of an object, are always shared in the class, unplugged activities also have a constructionist flavour and can be the first phase of a more structured constructionist proposal.

The following two examples, taken from CSUnplugged⁸⁴ and ALaDDIn⁸⁵, illustrate typical unplugged approaches to introduce children to programming.

In CSUnplugged “Rescue Mission”, pupils are given by the teacher a very simple language with only three commands: 1 step forward, 90 degrees left, 90 degrees right. The task is to compose a sequence of instructions to move a robot from one given cell on a grid to a given other cell. Pupils are divided into groups of three where each one has a role: either programmer, bot, or tester. This division of roles is done to emphasise the fact that programs cannot be adjusted on the fly; they must be first planned, then implemented, then tested and debugged until they work correctly. So the programmer must write down the instructions for the task, then pass them to the tester, who will pass them on to the bot and will observe what happens; its role is to underline what doesn’t work and hand them back to the programmer, who can then find the bug and fix it.

ALaDDIn “Algomotricity and Mazes” is an activity designed according to a strategy called algomotricity [Lonati et al. 2011; Bellettini et al. 2012, 2013, 2014], where pupils are exposed to an informatic concept/process by playful activities which involve a mix of tangible and abstract object manipulations; they can investigate it firsthand, make hypotheses that can then be tested in a guided context during the activity, and eventually construct viable mental models. Algomotricity starts “unplugged” [Bell, Rosamond, and Casey 2012] and ends with a computer-based phase to close the loop with pupils’ previous acquaintance with applications [Taub, Armoni, and Ben-Ari 2012].

“Algomotricity and Mazes” focuses on primitives and control structures. The task is that of verbally guiding a “human robot”(a blindfolded person) through a simple path. Working in groups, initially pupils are allowed to freely interact with the “robot”, then they are requested to propose a very limited set of primitives to be written each on a sticky note, and to compose them into a program to be executed by the “robot”. Also, they have the possibility of exploiting three basic control structures besides sequence (if, repeat-until, repeat-n-times). Groups may try their solutions as they wish and, when they are ready, each group is asked to execute its own program. Then the conductor may decide to swap some programs, so that a program is executed by the “robot” of another group. This allows the instructor to emphasise the ambiguity of some instructions or the dependency of programs on special features of the “robot” (e.g., step/foot size). In the last phase, students are given computers and a slightly modified version of Scratch. They are requested to write programs that guide a sprite through mazes of increasing complexity and foster the use of loop control structures⁸⁶.

2.3 Notional machines

Alan Perlis, in his foreword to “Structure and Interpretation of Computer Programs” (SICP) [Abelson, Sussman, and Sussman 1996], sets the stage for learning to program: “*Our traffic with the subject*

⁸⁴<https://csunplugged.org/>

⁸⁵<http://aladdin.di.unimi.it/>

⁸⁶The pictures shown in the first page show the first and last phase of the “Algomotricity and Mazes” activity, respectively.

matter of this book involves us with three foci of phenomena: the human mind, collections of computer programs, and the computer. Every computer program is a model, hatched in the mind, of a real or mental process.” And then: “The source of the exhilaration associated with computer programming is the continual unfolding within the mind and on the computer of mechanisms expressed as programs and the explosion of perception they generate. If art interprets our dreams, the computer executes them in the guise of programs!”.

This seems an important intuition for approaching programming from a constructionist perspective: programs are a join point between our mind and the computer, the interpreter of the formal description of what we have in mind. Thus, programs appeal to (or even exhilarate) our curiosity and ingenuity and are wonderful artifacts to share and discuss with other active minds. Such a sharing, however, assumes that the interpreter is a “common ground” among peers. When a group of people program the same ‘machine’, a shared *semantics* is in fact given, but unfortunately people, especially novices, do not necessarily write their programs for the formal interpreter they use, rather for the *notional machine* [Sorva 2013; Berry and Kölling 2014] they actually have in their minds.

A notional machine is an abstract computer responsible for executing programs of a particular kind [Sorva 2013] and its grasping refers to all the general properties of the machine that one is learning to control [Boulay 1986]. The purpose of a notional machine is to explain, to give intuitive meaning to the code a programmer writes. It normally encompasses an idealized version of the interpreter and other aspects of the development and run-time environment; moreover it should bring also a complementary intuition of what the notional machine cannot do, at least without specific directions of the programmer.

To introduce a notional machine to the students is often the initial role of the instructors. Ideally this should be somewhat incremental in complexity, but not all programming languages are suitable for incremental models: in fact most of the success for introductory courses of visual languages (see 3.5) or lisp dialects (see 3.4 and 3.1) is that they allow shallow presentations of syntax, thus focusing the learners on the more relevant parts of their notional machines⁸⁷.

An explicit reference to the notional machine can foster meta-cognition and during team work it can help in identifying misconceptions (see 2.5). But how can the notional machine be made explicit? The discussion with novice programmers should be guided by an effort of tracing the computational process and visualizing the execution, in order to make as clear as possible what one expects the notional machine will do and what it actually does.

2.4 Abstract programming patterns

One of the most relevant competencies to be mastered when learning computer programming is that of recognizing how a relatively low number of abstract patterns can be applied to a potentially infinite spectrum of specific conditions. This is often a challenge for novices, given that most of the times the discipline is taught using a two-step procedure based on: (i) introducing one or more primitive tools (say functions, variables, or control flow statements), and (ii) showing some (small number of) examples highlighting how these tools can be combined together in order to solve specific problems. This might lead to the rise of *misconceptions* of pupils w.r.t. the above mentioned tools (see 2.5 for more details).

⁸⁷On the other hand, several languages of widespread use by experienced programmers are in this sense more complex to handle in the context of introductory courses on programming. This is due to the fact that they force the novice to perform true leaps of faith in accepting an intricate syntax required even to write the simplest programs, and definitely obfuscating the underlying notional machine. Just to state an example, implementing in Java a canonical “hello, world!” requires the programmer to define a class, add to the latter a public, static, void method, provide a contract for this method, written in terms of an array of strings, and actually write the only relevant line of code, yet invoking a static method on a class variable of a system class.

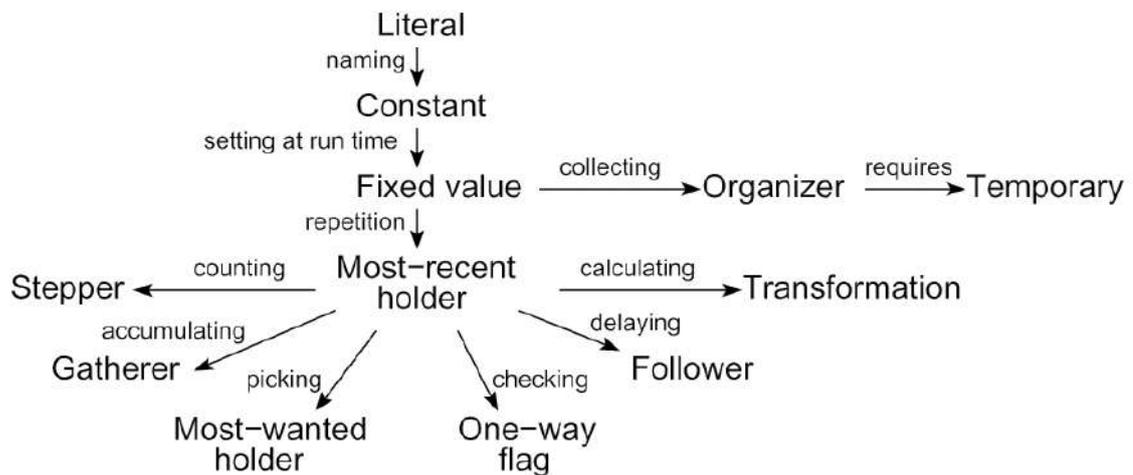


Figure 2: Roles of variables, organized in a constructionist-like hierarchy where the predecessor of an arrow is a prerequisite for learning the corresponding successor (source: [Sajaniemi 2002])

The concept of *role of variables* [Sajaniemi 2002; Proulx 2000] has been proposed in order to suggest a more constructionist-like learning path in which knowledge about variables is built exploiting some concepts at an intermediate level between those of the operational definition of a variable as the holder of a mutable value of a given type (say, of a float variable storing numeric quantities) and its specific use cases in order to solve given problems (for instance, that of computing the maximal value within a sequence of numeric quantities). Here, the key concept is related to an *abstract* use of variables — regardless of their type — following a small number of *roles* (namely: *fixed value*, *stepper*, *follower*, *most-recent holder*, *most-wanted holder*, *gatherer*, *transformation*, *one-way flag*, *temporary*, and *organizer*). Just to state a couple of examples,

- *most-wanted holder* identifies the role of a variable storing the most appropriate value found at any intermediate execution time in order to solve a problem (e.g., the variable typically containing the maximal encountered value while scanning a sequence), and
- the *follower* role applies to all variables whose values is always copied from other variables when the latter are about to be changed (like in the naive algorithm used in order to teach how to generate the Fibonacci sequence without introducing recursion or the golden ratio).

Keeping up with the example of finding a maximal value, instead of jumping right away to the algorithmic solution, there is a great opportunity in letting pupils reason about how this problem is in fact a special case of the more general quest for an optimal value which can be found exploiting a *greedy* strategy. This strategy consists in using a *most-wanted holder* to be compared each time with a new element of the sequence, possibly updating the former if the new element allows us to find a more precise solution. This general-purpose method easily fits the search of the maximal value, as well as the minimal one, but it allows us to efficiently handle less obvious cases such as that of finding the distinct vowels occurring in a sentence.

The roles themselves fit the constructionist approach also because they can be gradually introduced following the hierarchy illustrated in Figure 2, starting from the concept of *literal* (e.g., an integer value or a string) and building knowledge about one role on the top of the knowledge of already understood roles.

Loop patterns [Astrachan and Wallingford 1998] are an analogous interesting teaching subject when the concept to be learned identifies with iterations. For instance, the *loop and a half* pattern is introduced as an efficient way of implementing a processing strategy to be applied to a sequence of elements

```
while True:
    value = input('insert a positive, odd value')
    if value > 0 and value %
        break
    print('the value is not valid')
```

Figure 3: A typical loop and a half pattern applied to the repeated validation of external input to a procedure

whose end can be detected only *after* having accessed at least one of the elements themselves. The pattern here resorts to rebuilding the almost extinguished repeat/until iterative structure [Roberts 1995] using an infinite loop whose body accesses the next sequence element and subsequently checks whether or not it is the last one. If it is, the loop is escaped through a controlled jump, otherwise some special actions (such as printing a warning) are executed before proceeding to the next iteration. In this way, there is no need of duplicating code lines (typically accessing the first element in the sequence outside the loop), and the check concerns the end of sequence rather than its logical negation (which could be harder to grasp if it involves non-trivial logical connectives). Figure 3 shows one of the canonical incarnations of this pattern, namely the possibly repeated check of a value given as input via keyboard, detecting and ignoring invalid entries; other incarnations concern in general the handling of external inputs (for instance when reading from a file), or the serialization of sequences which have been generated computationally. It is worth noting that a loop might encompass more than one pattern simultaneously: for instance the code in Figure 3 is also an example of *polling loop*. Loop patterns fit well within a constructionist-based learning path also because they naturally arise when critically analyzing a less efficient implementation of a loop: for instance, the previous polling loop could be the point of arrival of a reasoning scheme which started from the detection of a duplicated line of code in a quick-and-dirty implementation proposed by pupils.

In general, abstract programming patterns are provided in a short number, so as to be easy to fully cover this subject within a standard introductory course in computer programming; moreover, the related concepts are easily and rapidly grasped by experienced computer science teachers [Ben-Ari and Sajaniemi 2004], thus they can be embedded in already existing curricula with low effort.

2.5 Misconceptions

Sorva defines a **misconceptions** as understandings that are deficient or inadequate for many practical programming contexts [Sorva 2013].

Some authors [Ben-Ari 2001] believe that computer science has an exceptional position in constructivist's view of knowledge constructed by individuals or groups rather than a copy of an ontological reality: in fact, the computer forms an "accessible ontological reality" and programming *features many concepts that are precisely defined and implemented within technical systems [...] sometimes a novice programmer "doesn't get" a concept or "gets it wrong" in a way that is not a harmless (or desirable) alternative interpretation. Incorrect and incomplete understandings of programming concepts result in unproductive programming behavior and dysfunctional programs* [Sorva 2013].

According to Clancy [Clancy 2004] there are two macro-causes of misconceptions: *over- or under-generalizing* and *a confused computational model*. High-level languages provide an abstraction on control and data, making programming simpler and more powerful, but, by contrast, hiding details of the executor to the user, who can consequently find mysterious some constructs and behaviors.

Much literature about misconceptions in CSEd can be found: we list some of the most important causes of misconceptions, experienced especially by novices, divided into different areas, found mainly in [Clancy 2004; Sirkiä 2012; Sorva 2013] and in works they reference. For a complete review see for example [Qian and Lehman 2017].

English Keywords of a language do not have the same meaning in English and programming. For example, the word *while* in English indicates a constantly active test, while the construct `while` can test the condition again only at the beginning of the next iteration. Some students believe that the

loop ends at the precise moment the condition is falsified. Similarly, some of them think of the `if` construct as a test continuously active and awaiting the occurrence of a condition, others believed that the `then` branch is executed as soon as the condition becomes true.

Syntax Although one may think the syntax is one of the biggest sources of misconception, studies show that it is a problem only in the very early stages. In particular, some students were able to write syntactically valid programs, which, however, were not useful for solving the given problem, or were semantically incorrect.

Mathematical notation Reported by many authors, classical is the confusion that generates the assignment with the `=` symbol (for example, seen as an equation or as a swap of values between variables) or the increment (`a = a + 1`) thought of as an impossible equation.

Examples of over-generalization Some authors found a series of non-existent constraints (e.g., methods in different classes that must have different names, arguments that can only be numbers, “dot” operator usable just in methods) dictated by the fact that the students had not seen any counterexample for such situations.

Similarities The analogy “a variable is like a box” can foster the idea that - like a box - it can contain more elements at the same time. The analogy “programming with the computer is like conversing with it” can bring to attribute *intentionality* to the computer and therefore to think that it:

- has a hidden intelligence that understands the intentions of the programmer and helps him achieve his goal (the so-called “superbug”);
- has a general vision, knowing also what will happen in lines of code that it is not currently running.

Some aspects of programming are particular carriers of misconceptions.

Sequence Many misconceptions are due to lack of understanding of the program flow: all lines active at the same time “magic” parallelism, order of instructions not important, difficulty in understanding the branches.

Passing parameters Students present difficulties in this area, for example by confusing the types of passing (by value, by reference ...), making confusion with the return value or with parameters scope.

Input Input statements are particularly problematic. Students do not understand where the input data come from, how they are stored and made available to the program. Some of them believe that a program remembers all the values associated with a variable (its “history”).

Memory allocation There are considerable difficulties in understanding the memory model of languages where allocation happens implicitly.

3. Programming languages for learning to program

Ptydepe, as you know, is a synthetic language, built on a strictly scientific basis. [...] There are many months of intensive study ahead of you, which can be crowned by success only if it is accompanied by diligence, perseverance, discipline, talent and a good memory. And, of course, by faith. [Václav Havel]

From a constructionist viewpoint of learning, programming languages have a major role: they are a key means for sharing artifacts and expressing one’s theories of world. The crucial part is that artifacts can *be executed* independently from the creator: someone’s (coded) mental process can become part of the experience of others, and thus criticized, improved, or adapted to a new project. In fact, the origin of the notion itself of constructionism goes back to Papert’s experiments with a programming environment (LOGO, see 3.1) designed exactly to let pupils tinker with math and geometry (a similar approach applied to physics had a smaller impact) [Papert 1980]. Does this strategy work even when the learning objective is the programming activity itself? Can a generic programming language be used to give a concrete reification of the computational thinking of a novice programmer? Or do we need something specifically designed for this activity? Alan Kay says that programming languages can be categorized in two classes: “agglutination of features” or “crystalization of style” [Kay 1993]. What is more important for learning effectively in a constructivist way? Features or style?

In the last decade, a number of block-based visual programming tools have been introduced which should help students to have an easier time when first practicing programming. These tools, often based on web-based technologies like Adobe Flash and later JavaScript, CSS, and HTML5, as well as an increase in the number of modern smartphones and tablets, opened up new ways for innovative coding concepts [Kahn 2017]. In general, they focus on younger learners, support novices in their first programming steps, can be used in informal learning situations, and provide a visual/block-based programming language which allows students to recognize blocks instead of recalling syntax [Tumlin 2017]. Many popular efforts for spreading computer science in schools, like [Goode, Chapman, and Margolis 2012] or the teaching material from Code.org [Code.org 2018] rely on the use of such block based programming environments. In addition, such tools are broadly integrated in primary through secondary schools, and even at universities, thus they have been adopted into many computing classes all over the world [Meerbaum-Salant, Armoni, and Ben-Ari 2010].

3.1 LOGO

LOGO was designed (since 1967) for (constructionist) educational purposes by Wally Feurzeig, Seymour Papert and Cynthia Solomon [Papert 1980]. Its syntax was heavily influenced by Lisp (at the time the standard language for Artificial Intelligence research) and LOGO featured a graphical (at least in principle) environment: the instructions the programmer writes are directed to a “turtle” (a small isosceles triangle in which the acutest angle marks the head) who moves around the screen, possibly leaving a colored trace. The turtle should help learners (typically 6th-8th graders) with a sort of self-identification: its movements have a clear correspondence with their movements in the real world, (although the turtle moves in a 2D space). The patterns drawn by the turtle can be the way the learners build their understanding of 2D geometry, discovering in the process even deep mathematical truths as the fact that a circle can be approximated by a high number of straight segments [Abelson and DiSessa 1986] (see Figure 4).

```
TO CIRCLE
  REPEAT FOREVER
  [
    FORWARD 1
    RIGHT 1
  ]
```

Figure 4: A procedure to draw a circle in LOGO

LOGO was conceived to empower learners of geometry and kinematics, not programming. Programming is *just* a means of expression, but one with a great epistemic potential. According to Papert: “in teaching the computer how to think, children embark on an exploration about how they themselves think. The experience can be heady: Thinking about thinking turns every child into an epistemologist, an experience not even shared by most adults” [Papert 1980]. Also, by expressing something in a way even the LOGO turtle can “understand” can be fruitful even for real world activities. Juggling, for example, a complex motoric activity which requires dedicated training, can be analyzed with LOGO: the identification of *proper* sub-activities (*i.e.*, sub-routines like TOP-RIGHT to recognize when one juggling ball is at the top of its trajectory going to the right, or TOSS-LEFT to throw the ball with the left hand) may shorten significantly the time for acquiring juggling skills (from days to hours, according to [Papert 1980]). And here ‘proper’ should be understood as appropriate to the task, but also as “fitting properly with the programming language idiomatic way of describing computational processes”. LOGO had many independent implementations and its approach is still very popular, even Python has a `turtle` package in its standard library.

3.2 Smalltalk

Smalltalk [Goldberg and Kay 1976] also has its roots in constructionist learning. Back in the early seventies, at the Learning Research Group within the Xerox Parc Research Center, people were envisioning a world of personal computing devices: they should have intuitive user interfaces and an explicit “programmability”. Smalltalk, with whose lineage traces clearly to LOGO and Lisp, was designed

with a general audience in mind, since everyone should be comfortable with programming and computing devices should become ubiquitous in learning environments “along the lines of Montessori and Bruner” [Kay 1993]. Thus, although clearly designed to foster a personal learning experience, Smalltalk was not directed specifically to children and it has conquered a wide professional audience. In Smalltalk everything is an ‘object’ able to react to ‘messages’. There follows a highly consistent object-oriented approach and code can be factored out by inheritance and dynamic binding. Smalltalk introduces also the idea that everything in the system is programmable: in fact, the tool-chain itself and the application a programmer is writing are indistinguishable and available for modifications, even at run-time. By design, such a dynamic environment encourages a trial-and-error approach. A common practice for Smalltalk developers is programming with the constant support of the debugger: instead of creating a method before its call, one sends a message that an object “does not understand”, then they use the debugger to catch the exception and write the code that is needed. A specific Smalltalk system for children was designed later as an evolution of Squeak Smalltalk: E-toys [Kay et al. 1997] provided a world of “sprites”, funny characters that can be moved (concurrently) around the screen by programming them in Smalltalk. E-toys then evolved in Scratch (see 3.5), where the programming part was replaced by visual blocks.

3.3 BASIC, Pascal

The search for programming languages suitable for students in fields other than hard sciences and mathematics was in fact started in the sixties. In 1964 at Dartmouth College rose BASIC (Beginner’s All-purpose Symbolic Instruction Code) [Kurtz 1978]. It seems legit to mention BASIC in a paper on constructionism and programming: for years BASIC has been the elective language for personal projects and even before widespread Internet connectivity, several communities shared BASIC programs in Bulletin Board Systems and magazines. Its popularity among self-taught programmers, however, was due mainly to its availability on personal and home computing devices. Moreover, the language was typically implemented using an interpreter, thus naturally fostering the trial-and-error and incremental learning styles typical of a constructionist setting. A generation grown with BASIC still thinks it is a wonderful approach to get children hooked on programming (see for example [Brin 2016]). However, many believe BASIC is not able to foster good abstractions and fear that BASIC programmers will bring bad habits to all their future computational activities⁸⁸.

In 1970 Niklaus Wirth published Pascal [Wirth 1993], a small, efficient ALGOL-like language intended to encourage good programming practices using structured programming and data structuring. For about 25 years, Pascal (and its successors like TurboPascal or Modula-2) was the most popular choice for undergraduate courses and a whole generation of computer scientist learned to program through its discipline of well-structured programs popularized by Wirth in his book “Algorithms + Data Structures = Programs”. Only Java had a similar success in undergraduate courses. However, while Java popularity was (and is) influenced by trends in software industry, Pascal was appealing mainly for its intrinsic discipline, which matched the academic sentiment of the time. Alan Perlis, in his foreword to “Structure and Interpretation of Computer Programs” (see 3.4), says: “Pascal is for building pyramids—imposing, breathtaking, static structures built by armies pushing heavy blocks into place. Lisp is for building organisms—imposing, breathtaking, dynamic structures built by squads fitting fluctuating myriads of simpler organisms into place. The organizing principles used are the same in both cases, except for one extraordinarily important difference: The discretionary exportable functionality entrusted to the individual Lisp programmer is more than an order of magnitude greater than that to be found within Pascal enterprises. Lisp programs inflate libraries with functions whose utility transcends the application that produced them. The list, Lisp’s native data structure, is largely responsible for such growth of utility. The simple structure and natural applicability of lists are reflected in functions that are amazingly nonidiosyncratic. In Pascal the plethora of declarable data structures induces a specialization within functions that inhibits and penalizes casual cooperation. It is better to have 100 functions operate on

⁸⁸A recent anecdote: Brian Kernighan — one of the designers of C — called a book written by a BASIC programmer “the worst C programming textbook ever written”! See <https://wozniak.ca/blog/2018/06/25/Massacring-C-Pointers/index.html> for the full story.

one data structure than to have 10 functions operate on 10 data structures. As a result the pyramid must stand unchanged for a millennium; the organism must evolve or perish.”

Structured programming had also some LOGO-like descendants, the most famous one is probably Karel [Pattis 1981], in which the programmer controls a simple robot that moves in a grid of streets (left-right) and avenues (up-down). A programmer can create additional instructions by defining them in terms of the five basic instructions, and by using conditional control flow statements with environment queries.

3.4 Scheme, Racket

Scheme [Abelson et al. 1998] is a language originally aimed at bringing structured programming in the lands of Lisp (mainly by adding lexical scoping). The language has been standardized by IEEE in 1999 and nowadays it has a wide and energetic community of users. Its importance in education, however, is chiefly related to a book, “Structure and Interpretation of Computer Programs” (SICP) [Abelson, Sussman, and Sussman 1996], which had a tremendous impact on the practice of programming education. The book derived from a semester course taught at MIT. It has the peculiarity to present programming as a way of organizing thinking and problem solving. Every detail of the Scheme (which, being a Lisp dialect, has lightweight syntax) notional machine is worked out in the book: in fact at the end, the reader should be able to understand the mechanics of a Scheme interpreter and to program one by her/himself (in Scheme). The book, which enjoyed widespread adoption, was originally directed to MIT undergraduates and it is certainly not suitable either for children or even adults without a scientific background: examples are often taken from college-level mathematics and physics. Its emphasis on “organized abstraction” and “procedural epistemology” makes it a fundamental reading for anyone reflecting on teaching and learning how to build complex systems.

A spin-off of SICP explicitly directed to learning is Racket. Born as ‘PLT Scheme’, one of its strength is the programming environment DrScheme [Findler et al. 2002] (now DrRacket): it supports educational scaffolding, it suggests proper documentation, and it can use different *flavours* of the language, starting from a very basic one (Beginning Student Language, it includes only notation for function definitions, function applications, and conditional expressions) to multi-paradigm dialects; this flexibility is relatively easy in a Lisp-like world, since most of the “syntax” is in fact provided by macros, that can be active or not⁸⁹. The DrRacket approach is supported by an online book “How to design programs” (HTDP)⁹⁰ and it has been adapted to other mainstream languages, like Java [Allen, Cartwright, and Stoler 2002] and Python. The availability of different languages directed to the progression of learning should help in overcoming what the DrRacket proponents identify as “the crucial problem” in the interaction between the learner and the programming environment: beginners make mistakes before they know much of the language, but development tools yet diagnose these errors as if the programmer already knew the whole notional machine. Moreover, DrRacket has a minimal interface aimed at not confusing novices, with just two simple interactive panes: a definitions area, and an interactions area, which allows a programmer to ask for the evaluation of expressions that may refer to the definitions. Similarly to what happens in visual languages, Racket allows for direct manipulation of sprites, see an example in Figure 5.

⁸⁹A small syntactic improvement of Racket over Lisp, very useful for beginners, is that nested parentheses can be matched easily by using different bracket families: for example, (- {/ [* (+ 2 3) 4] 2 } 1)

⁹⁰Current version: <http://www.htdp.org/2018-01-06/Book/index.html>

```

(define (picture-of-rocket.v3 height)
  (cond
    [(<= height (- 60 (/ (image-height  ) 2))]
    (place-image  50 height
      (empty-scene 100 60)))
    [(> height (- 60 (/ (image-height  ) 2))]
    (place-image  50 (- 60 (/ (image-height  ) 2))
      (empty-scene 100 60)))))

```

Figure 5: Racket code for “landing a rocket”

The authors of HTDP claim that “program design — but not programming — deserves the same role in a liberal-arts education as mathematics and language skills.” They aim at systematically designed programs thanks to systematic thought, planning, and understanding from the very beginning, at every stage, and for every step. To this end the HTDP approach is to present “design recipes”, supported by predefined scaffolding that should be iteratively refined to match the problem at hand. This is indeed very close to the idea of micropatterns discussed in 2.4.

3.5 Scratch, Snap!, Alice, and others

EToys worlds (see 3.2) with pre-defined — although programmable — objects, evolved in a generic environment in which everything can be defined in terms of ‘statement’ blocks. Scratch [Resnick et al. 2009], originally written in Smalltalk (but this is hidden to most users: only a “secret” key combination can bring the Smalltalk environment alive), is the most popular and successful visual block based programming environment. Launched in 2007 by the MIT Media Lab, the Scratch site has grown to more than 25 million registered members with over 29 million Scratch projects shared programs.

Unlike traditional programming languages, which require code statements and complex syntax rules, here graphical programming blocks are used that automatically snap together like Lego bricks when they make syntactical sense [Ford 2009]. In visual programming languages, a block represents a command or action and they are arranged in scripts. The composition of individual scripts equals the construction of an algorithm. The building blocks offer the possibility, e.g., to animate different objects on the stage, thus defining the behavior of the objects. In addition to the basic control structures, there are event-triggering building blocks/conditions for event-driven programming [Georgios and Kiriaki 2009]. Familiar concepts such as variables, variable lists, Boolean logic, user interface design, etc. are provided as well. Furthermore, most visual programming environments offer the possibility to integrate graphics, animations, music, and sound to create video games, movies, and interactive stories. In that way, creative and artistic talents of the students are displayed in their games, stories, and applications. Thereby, these visual languages offer the same programming logic and concepts as other (text-based) programming languages.

Some characteristics of the Scratch environment [Maloney et al. 2010] are particularly relevant in the constructionist approach.

Liveness The code is constantly executed and can be changed on the fly, immediately seeing the runtime effects of the change; this encourages users to tinker with the code.

No error messages When you play with Lego bricks, they stack together or they don’t - the same happens in Scratch; program always run: syntax errors are prevented from the block shapes and connections, and also runtime errors are avoided by doing something “reasonable” (e.g., in the case of an out-of-range value); this is particularly important not to frustrate kids and to keep them iterating and developing: “*A program that runs, even if it is not correct, feels closer to working than a program that does not run (or compile) at all*” [Maloney et al. 2010].

Other characteristics are useful to help novices avoiding misconceptions that often arise when starting to learn to program.

Execution made visible A glowing yellow border surrounds running scripts; moreover version 1.4 (and Snap!, for example) provides a “single-stepping” mode, where each block is highlighted when it is executed; this is very helpful in program reading and debugging, and helps students form a correct mental model of the notional machine underlying the program execution.

Making data concrete You can see in a variable box, automatically shown, its current value: again, this is helpful for making the underlying machine model visible.

Finally, other characteristics introduce important software engineering and development concepts.

Open source Each shared project has a “see inside” button that brings you to the project source; you can read and edit the blocks to see what happens.

Remixing If you edit someone else’s project, you create a remix: you are the author, but the system automatically gives credits to the original author (at any depth, keeping track of multiple remixes in a tree) and suggests you to explicitly declare what changes you made.

The main limitation of Scratch programs is that they do not scale well from the abstraction point of view: only since version 2 you can “make a new block” that is, a procedure with optional parameters. These blocks have no possibility to return a value (like a number or a boolean) and so can’t be nested inside other blocks, forcing you to modify global variables if needed.

Snap!⁹¹ (originally BYOB, Build Your Own Blocks) is an extended reimplementa-tion of Scratch with functions and continuations. These added capabilities make it suitable for a serious introduction to computer science for high school or college students: in fact, Snap! is used as the basis for an Advanced Placement CS course at Berkeley⁹².

The Scratch approach was also ported to mainstream programming languages: in Alice [Dann, Cooper, and Pausch 2008] visual blocks are in fact Java instructions. Alice worlds are 3D: this choice makes it very attractive and appealing to pupils, who can program amazing 3D animations. It also adds many complexities, since moving objects in a 3D space is not trivial.

Recently, several block-based development environments for web-browsers were published (even the last version of Scratch is web based, thus abandoning its Smalltalk inner soul). The most popular is probably Google’s Blockly⁹³, which allows for programming both with blocks and textual programming languages (Javascript and Python): the programmer can see the source code in different interchangeable ways. MIT has developed a version of Blockly that can even be used to create Android applications, to be executed on mobile phones and that can take advantage of sensors and Google services like maps (App Inventor⁹⁴). Even Apple recently proposed a block-based interface for its Swift programming language⁹⁵ and other Android-based visual programming language environments exist. For example, Pocket Code allows the creation of games, stories, animations, and many types of other apps directly on phones or tablets, thereby teaching fundamental programming skills [Slany 2014]. The free and open source project Catrobat⁹⁶ was initiated 2010 in Austria at TU Graz. This team develops free educational apps for teenagers with the aim to introduce young people to programming. With a playful approach, young people can be engaged and game development can be promoted with a focus on design and creativity. The drag and drop interface provides a variety of bricks that can be joined together to develop fully fledged programs. The app is freely available for Android at the Google’s Play Store⁹⁷ and soon it will be available on the Apple App Store for iOS. The target audience for Pocket

⁹¹<https://snap.berkeley.edu/>

⁹²<https://bjc.berkeley.edu/>

⁹³<https://developers.google.com/blockly/>

⁹⁴<http://appinventor.mit.edu>

⁹⁵<https://www.apple.com/swift/playgrounds/>

⁹⁶<https://www.catrobat.org/>

⁹⁷<https://catrob.at/pc>

Code are teenagers between the age of 11 and 17 years old who have access to or own an Android smartphone.

Since its first versions, Scratch had blocks able to connect and program external (physical) robots. In fact most of the mentioned environments can connect to physical devices and sensors, with the goal of increasing the constructionist appeal of block programming, and opening to the world of “tinkering” with electronics. Resources like the ‘Makey Makey’⁹⁸ tool became popular for activities during coding workshops with innovative forms of production and do-it-yourself work [Schön, Ebner, and Kumar 2014; Sheth, Bell, and Kaiser 2012].

All in all, visual programming languages seem to provide an easier start and a more engaging experience for learners. The ease of use, simplicity, and desirability of new visual programming environments enables young people to imagine complex goals. A study which compared three classes that used either block-based (Scratch), text-based (Java), or hybrid blocks/text (Snap!/JavaScript) programming languages showed that students generally found block-based programming to be easier than the text-based environments [Weintrop and Wilensky 2015]. Some researchers, however, argue that students are not fully convinced that a visual language can help them learn other programming languages [Lewis et al. 2014].

3.6 Common features

The above short survey of programming languages for education shows they have some recurrent traits that link them to the themes discussed in Sect. 2.

Personification The interpreter becomes a “persona”, computation is then carried out through anthropomorphic (or, better, zoomorphic, since animals are very common) actions. This seems to contradict a famous piece of advice coming from no less than E. W. Dijkstra [Dijkstra 1985]. Speaking of anthropomorphism in computer science, he noted: “The trouble with the metaphor is, firstly, that it invites you to identify yourself with the computational processes going on in system components and, secondly, that we see ourselves as existing in time. Consequently the use of the metaphor forces one to what we call ‘operational reasoning’, that is reasoning in terms of the computational processes that could take place. From a methodological point of view this is a well-identified and well-documented mistake: it induces a combinatorial explosion of the number of cases to consider and designs thus conceived are as a result full of bugs.” The *reasoning in terms of the computational processes*, however, is what is probably needed for a novice in order to familiarize with the notional machine. Some sort of operational reasoning seems also important to foster the basic intuition needed by a constructivist approach, in which concrete, public actions are key.

Visualization and tracking Computational processes that evolve in time are described by static texts: the mapping between the two is not trivial and it requires an understanding of the notional machine. Educational programming environments often try to make the mapping more explicit with some visualization of the ongoing process: the trace left by the LOGO turtle, or some other exposition of the changing state of the interpreter.

Appeal Engagement of learners is crucial: to this end it is important to give learners powerful libraries and building blocks. It is not clear, however, how to properly balance amazing effects in order to avoid they become a major distraction: sometimes children may spend their (limited) time in changing the colors of the sprites, instead of trying to solve problems. While they surely learn something even in these trivial activities, they also risk to lose their opportunity of recognizing the thrill which comes when solving computational problems by themselves.

⁹⁸<https://makeymakey.com/>

4 Learning to think computationally

The expression “Computational Thinking” (CT) has a very special relationship with Papert’s constructionism: the first ascertained attestation is in Papert’s seminal book *Mindstorms* [Papert 1980]. As already stated in 3.1, Papert was not focused on teaching computer science concepts, but on the use of computation as a means of self expression and as a tool for learning ideas and concepts of other disciplines.

The expression almost completely disappeared (except for a Papert’s work about math [Papert 1996]) till 2006.

In 2006 Jeannette Wing brought the expression “Computational Thinking” back to the discussion [Wing 2006], gaining a massive attention⁹⁹. In that seminal article, Wing didn’t give a definition, but tightened the concept to the computer science discipline, stating “*Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science*” or that “*thinking like computer scientists*” is a fundamental skill for everyone.

In the following years she proposed (along with Cuny, Snyder and Aho) a formal definition: CT is “*the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent*” [Wing 2010]. Wing argues that children who are introduced to CS learn more than just programming and that this opens a new way of thinking [Wing 2008]. Moreover, students should think first about possible solutions to a given problem (problem solving skills), and second implement their ideas by using a computing device (programming skills [Lye and Koh 2014]). In order to successfully implement their own solutions, students have to apply different programming concepts, such as loops and conditions, as well as practices, such as abstraction and debugging [Kafai and Burke 2013]. For teachers, however, developing computational thinking skills in students is actually a challenging task [Sentance and Csizmadia 2015]. After Wing’s articles, the discussion about the nature of CT and how to teach it in fact exploded.

In [Corradini, Lodi, and Nardelli 2017] authors analysed different CT definitions and frameworks, finding out that all definitions agreed *on the fact that CT is a **way of thinking** (thought process) for **problem solving*** but specifying that *it is not just problem solving: the formulation and the solution of the problem must be expressed in a way that allows a **processing agent** (a human or a machine) to carry it out.*

Moreover, authors in [Corradini, Lodi, and Nardelli 2017] noticed that each definition listed some characteristics of CT, from thinking habits to specific programming concepts. They classified them in four categories. We quote here their findings.

Mental processes: Mental strategies useful to solve problems.

- *Algorithmic thinking:* use algorithmic thinking to design a sequence of ordered steps (instructions) to solve a problem, achieve a goal or perform a task.
- *Logical thinking:* use logical thinking and reasoning to make sense of things, establish and check facts.
- *Problem decomposition:* split a complex problem in simpler subproblems to solve it more easily; modularize; use compositional reasoning.
- *Abstraction:* get rid of useless details to focus on relevant information / ideas.
- *Pattern recognition:* discover and use regularities in data, problems.
- *Generalization:* use discovered similarities to make predictions or to solve more general problems.

Methods: Operational approaches widely used by computer scientists.

⁹⁹Currently (August 2018), the paper has more than 3800 citations, according to Google Scholar

- *Automation*: automate the solutions; use a computer or a machine to do repetitive tasks.
- *Data collection, analysis and representation*: gather information/data, make sense of them by finding patterns, represent them properly; store, retrieve and update values.
- *Parallelization*: carry out tasks simultaneously to reach a common goal, use parallel thinking.
- *Simulation*: represent data and (real world) processes through models, run experiments on models.
- *Evaluation*: implement and analyze solutions to judge them, in particular for what concerns effectiveness, efficiency in terms of time and resources.
- *Programming*: use some common concepts in programming (eg. loops, events, conditionals, mathematical and logical operators).

Practices: Typical practices used in the implementation of computing machinery based solutions.

- *Experimenting, iterating, tinkering*: in iterative and incremental software development, one develops a project with repeated iterations of a design-build-test cycle, incrementally building the final result; tinkering means trying things out using a trial and error process, learning by playing, exploring, and experimenting.
- *Test and debug*: verify that solutions work by trying them out; find and solve problems (bugs) in a solution/ program.
- *Reuse and remix*: build your solution on existing code, projects, ideas.

Transversal skills: General ways of seeing and operating in the world; useful life skills enhanced by thinking like a computer scientist.

- *Create*: design and build things, use computation to be creative and express yourself.
- *Communicate and collaborate*: connect with others and work together to create something with a common goal and to ensure a better solution.
- *Reflect, learn, meta-reflect*: use computation to reflect and understand computational aspects of the world.
- *Be tolerant for ambiguity*: deal with non-well specified and open-ended real-world problems.
- *Be persistent when dealing with complex problems*: be confident in working with difficult or complex problems, persevering, being determined, resilient and tenacious.

As you may notice, a lot of concepts are involved. Sometimes this led to critiques [Hemmeldinger 2010]: some of these concepts are not exclusively associated with CS, but taught in other disciplines (e.g., math) or are general skills that children have been learning for a long time before the birth of CS. Anyway CS brings to the discussion some characteristic problem solving methods (e.g., the possibility to effectively execute a solution/a model/an abstraction by running an implementation of its algorithm [Martini 2012]). Constructionist's spirit emerges mainly in practices and transversal skills, but we should pay attention not to forget mental processes and CS methods: losing this connection may lead to misconceptions about the nature of computational thinking, de-empowering Wing's idea of the importance of "thinking like computer scientists" in many human activities.

The most used technique to teach CT is teaching to program (with languages suitable for the learner's age and capacity). Programming is in fact the main way computer scientists express their solutions and so the main way they learn to think computationally. Teaching to program leads to problems and opportunities discussed in the previous sections. Nonetheless, unplugged activities (see 2.2) have been proposed as well, as they help kids to act like the computer, becoming aware of the underlying notional machine (see 2.3).

5 Learning to program in teams

Working in teams requires new skills, especially because software products (even the ones in the reach of novices) are often tangled with many dependencies and division of labour is hard: it inevitably requires appropriate communication and coordination. Therefore, it is important that novice programmers learn to program in an “organized” way, thus discovering that as a group they are able to solve more challenging and open-ended problems, maybe with interdisciplinary contributions.

To this end, agile methodologies seem to fit well with the needs of a constructionist team of learners and they are increasingly exploited in educational settings (see for example [Kastl, Kiesmüller, and Romeike 2016; Missiroti, Russo, and Ciancarini 2016]):

- agile teams are typically small groups of 4–8 co-workers;
- agile values [Beck et al. 2001] (individuals and interactions over processes and tools; customer collaboration over contract negotiation; responding to change over following a plan; working software over comprehensive documentation) relate well with constructivist philosophies;
- agile teams are self-organizing and emphasize the need of reflecting regularly on how to become more effective, and tune and adjust their behavior accordingly;
- typical agile techniques like pair programming, test driven development, iterative software development, continuous integration are very attractive for a learning context.

5.1 Iterative software development

For program development cycles, concepts of agile and iterative software development can be used to leverage this process and to see first results very quickly [DeMarco-Brown 2013; Davies and Sedley 2009]. Figure 6 visualizes the process of game design in reference to agile methods. As the scope of this paper is limited, only a simplified life cycle is visualized to describe the process of how a team works in iterations to deliver or release software.

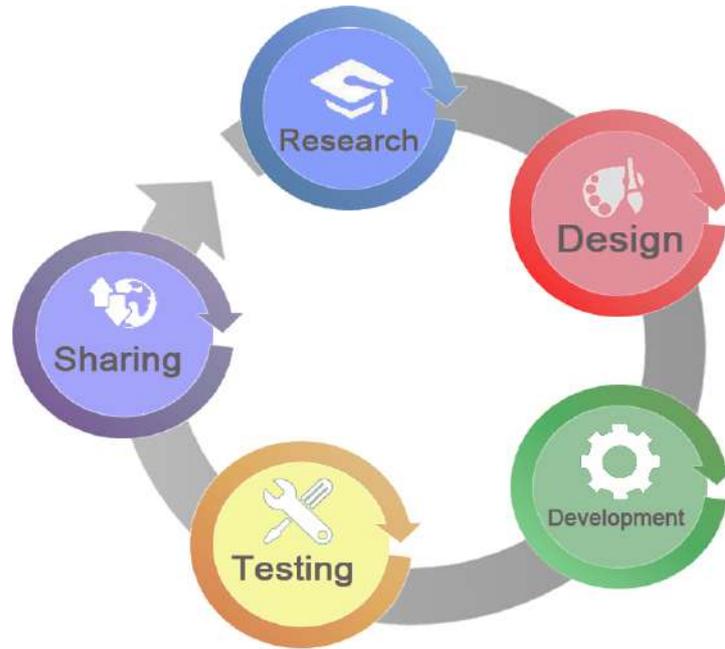


Figure 6: Agile Development Cycle (DeMarco-Brown 2013; Davies and Sedley 2009)

The first step, *Research*, includes the development of the core idea by producing a simple game concept or a storyboard. In this phase the story, title, genre, and theme of the game should be selected, as well as a rough concept about the structure and gameplay.

In the second step, during the *Design*, the artwork, game content and other elements (characters, assets, avatars, etc.), and the whole gaming world is produced.

The *Development* phase, includes all of the actual programming, followed by *Testing* the code and the software (playtesting). Several iterations between testing and bug fixing are possible.

As a final step, the *Release* phase is planned. This could include several beta releases or a final release for end users. The agile model required to get started with the project works to bring customer satisfaction by rapid, continuous delivery of useful software.

“In an iterative methodology, a rough version of the game is rapidly prototyped as early in the design process as possible. This prototype has none of the aesthetic trappings of the final game, but begins to define its fundamental rules and core mechanics.” [Salen and Zimmerman 2003, p. 11]

To phrase it differently, before focusing on every detail of the project, focus on the smallest step the games needs for playing it, e.g., limited interface control but basic game functionality.

Collaborative work and the connection with a real world problem makes their learning valuable [Jaime and Ruby 2011]. Project work is always student-centered and task oriented. The final artefacts of this project work can be shared with a community, thus fostering ownership. The benefits of learning through projects include enhanced students’ participation within active learning and self-learning, enhanced communication skills, addressing of a wider set of learning styles, and improved critical thinking skills. This collaboration is needed for developing the game idea, communicating, sharing, and managing assets and codes, playtesting and documentation.

Teachers have to consider how to support collaboration and communication during the whole game production process [Ferreira et al. 2008]. They must therefore stick to certain design patterns and iterative cycles (e.g., agile) and explain game elements and rules. The project in general should foster the teamwork in producing game assets and software. Teachers have to take into account the different preferences of students, e.g., if they feel more confident in the role of developers or artists, or that

students do not shy away from switching roles. Finally, the teacher has to ensure that the ideas for the project are simple and clear, as well as reduce the size and complexity of the game projects.

5.2 Test-driven development and debugging

Within the agile framework, a special constructivist emphasis is provided by test-driven development, or TDD for short [Beck 2003]. This technique reverses the classical temporal order between the implementation of code and the test of its correctness. Namely, the specification of the programming task at hand is actually provided with a test that *defines* correct behavior. The development cycle is then based on the iteration of the following three-step procedure:

1. write a test known to fail according to the current stage of the implementation,
2. perform the smallest code update which satisfies all considered tests, including the one introduced in previous point, and
3. optionally refactor the produced code.

TDD makes testing the engine driving the overall development process, and suggesting a good *next test* is one of the hardest-to-find contributions that a facilitator might give in an active programming learning context. Such suggestion has indeed the role of letting pupils aware that their belief at a broad level (“the program works”) is false, thus an analogous belief at a smaller scale (for instance, “this function always returns the correct result”) should be false, too. This amounts to the destruction of knowledge necessary to build new knowledge (aka a working program) in a constructivist setting. Even the refactoring step corresponds well to the constructivist re-organization of knowledge that follows the discovery of more viable solutions. In fact, most of the developing activities consist in *realizing* that a more or less complex system which was thought to correctly work actually it is not able to cope with a new arising test case. This applies of course also to the simplest coding tasks faced by students engaged in learning the basis of computer programming.

Once pupils are convinced that their implementation is flawed, the localization of the code lines to be reconsidered is the other pillar of an active learning setting. Again, a paramount contribution for a successful learning process should be provided by a facilitator suggesting suitable debugging techniques (e.g., proposing critical input values to the wrong program, hinting specific points in the execution flow to be verified, or giving advice about variables to be tracked during the next run).

6 Conclusions

Literature on learning to program through a constructionist strategy mostly focuses on how to bring the abstract and formal nature of programming languages into manipulation of more concrete (or even tangible) “objects”. Many proposals aim at overcoming the (initial) hurdles which textual rules of syntax may pose to children. Also, several environments have been designed in order to increase the appeal of programming by connecting this activity to real world devices or providing fancy libraries. Instead, more work is probably needed to make educators and learners more aware of the so-called notional machine behind the programming language. Programming environments could be more explicit about the complex relationship between the code one writes and the actions that take place when the program is executed. Moreover, micro-patterns should be exploited in order to enhance problem solving skills of novice programmers, such that they become able to think about the solution of problems in the typical way that make the former suitable to automatic elaboration. Agile methodologies, now also common in professional settings, seem to fit well with constructionist learning. Besides the stress on team working, particularly useful seems the agile emphasis on having running artifacts through all the development cycle and the common practice of driving development with explicit or even executable “definitions of done”.

References

Abelson, H., and A.A. DiSessa. 1986. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. Artificial Intelligence Series. AAAI Press.

- Abelson, H., R.K. Dybvig, C.T. Haynes, G.J. Rozas, N.I. Adams, D.P. Friedman, E. Kohlbecker, et al. 1998. "Revised5 Report on the Algorithmic Language Scheme." *Higher-Order and Symbolic Computation* 11 (1): 7–105. <https://doi.org/10.1023/A:1010051815785>.
- Abelson, H., G.J. Sussman, and J. Sussman. 1996. *Structure and Interpretation of Computer Programs*. Second. MIT press.
- Allen, Eric, Robert Cartwright, and Brian Stoler. 2002. "DrJava: A Lightweight Pedagogic Environment for Java." *SIGCSE Bull.* 34 (1). New York, NY, USA: ACM: 137–41. <https://doi.org/10.1145/563517.563395>.
- Astrachan, Owen, and Eugene Wallingford. 1998. "Loop Patterns." In *Proceedings of the Fifth Pattern Languages of Programs Conference*.
- Beck, Kent. 2003. *Test-Driven Development: By Example*. Addison-Wesley Professional.
- Beck, Kent, and Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change*. Second. Addison-Wesley Professional.
- Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, et al. 2001. "Manifesto for Agile Software Development." <http://agilemanifesto.org/iso/en/manifesto.html>.
- Bell, T., F. Rosamond, and N. Casey. 2012. "Computer Science Unplugged and Related Projects in Math and Computer Science Popularization." In *The Multivariate Algorithmic Revolution and Beyond*, edited by Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, 398–456. Berlin, Heidelberg: Springer-Verlag. <http://dl.acm.org/citation.cfm?id=2344236.2344256>.
- Bellettini, Carlo, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, and Mauro Torelli. 2012. "Exploring the Processing of Formatted Texts by a Kynesthetic Approach." In *Proc. of the 7th Wipsce*, 143–44. WiPSCE '12. New York, NY, USA: ACM. <https://doi.org/http://dx.doi.org/10.1145/2481449.2481484>.
- Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, and Mauro Torelli.. 2013. "What You See Is What You Have in Mind: Constructing Mental Models for Formatted Text Processing." In *Proceedings of ISSEP2013*, 139–47. Commentarii Informaticae Didacticae 6. Universitätsverlag Potsdam. <http://opus.kobv.de/ubp/volltexte/2013/6368/pdf/cid06.pdf>.
- Bellettini, Carlo, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, Mauro Torelli, and Luisa Zecca. 2014. "Extracurricular Activities for Improving the Perception of Informatics in Secondary Schools." In *Proceedings of ISSEP2014*, edited by Yasemin Gülbahar and Erin c Karata s, 8730:161–72. Lecture Notes in Computer Science. Springer. https://doi.org/http://dx.doi.org/10.1007/978-3-319-09958-3_15.
- Ben-Ari, Mordechai. 2001. "Constructivism in Computer Science Education." *Journal of Computers in Mathematics and Science Teaching* 20 (1). Association for the Advancement of Computing in Education (AACE): 45–73.
- Ben-Ari, Mordechai, and Jorma Sajaniemi. 2004. "Roles of Variables as Seen by Cs Educators." In *ACM Sigcse Bulletin*, 36:52–56. 3. ACM.
- Berry, Michael, and Michael Kölling. 2014. "The State of Play: A Notional Machine for Learning Programming." In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, 21–26. ITiCSE '14. New York, NY, USA: ACM. <https://doi.org/10.1145/2591708.2591721>.
- Boulay, Benedict Du. 1986. "Some Difficulties of Learning to Program." *Journal of Educational Computing Research* 2 (1): 57–73. <https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9>.
- Brin, David. 2016. "Why Johnny Can't Code." https://www.salon.com/2006/09/14/basic_2/.
- Clancy, Michael. 2004. "Misconceptions and Attitudes That Interfere with Learning to Program." In *Computer Science Education Research*, edited by Sally Fincher and Marian Petre, 85–100. Routledge.

Code.org. 2018. "HOour of Code." Retrieved July 17, 2018 from <https://hourofcode.com/de>.

Corradini, Isabella, Michael Lodi, and Enrico Nardelli. 2017. "Conceptions and Misconceptions About Computational Thinking Among Italian Primary School Teachers." In *Proceedings of the 2017 Acm Conference on International Computing Education Research*, 136–44. ICER '17. New York, NY, USA: ACM. <https://doi.org/10.1145/3105726.3106194>.

Dann, Wanda P, Stephen Cooper, and Randy Pausch. 2008. *Learning to Program with Alice*. Prentice Hall Press.

Davies, Rachel, and Liz Sedley. 2009. *Agile Coaching*. The Pragmatic Bookshelf.

DeMarco-Brown, Diana. 2013. *Agile User Experience Design - A Practitioner's Guide to making it work*. In Elsevier Inc.

Dijkstra, Edsger W. 1985. "On Anthropomorphism in Science. EWD936." <https://www.cs.utexas.edu/users/EWD/ewd09xx/EWD936.PDF>.

Ferreira, Alexandre, Eliane Pereira, Junia Anacleto, Aparecido Carvalho, and Izaura Carelli. 2008. "The common sense-based educational quiz game framework 'What is it?'." In *ACM International Conference Proceeding Series*.

Findler, Robert Bruce, John Clements, Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Paul Steckler, and Matthias Felleisen. 2002. "DrScheme: A Programming Environment for Scheme." *Journal of Functional Programming* 12 (2). Cambridge University Press: 159–82.

Ford, Jerry Lee. 2009. *Scratch programming for Teens*. In Computer Science Books.

Georgios, Fesakis, and Serafeim Kiriaki. 2009. "Influence of the Familiarization with 'Scratch' on Future Teachers' Opinions and Attitudes about Programming and ICT in Education."

Goldberg, Adele, and Alan Kay. 1976. *Smalltalk-72 Instruction Manual*. Xerox.

Goode, Joanna, Gail Chapman, and Jane Margolis. 2012. "Beyond curriculum: the exploring computer science program." In *Magazine ACM Inroads*.

Hauswirth, Matthias, Andrea Adamoli, and Mohammad Reza Azadmanesh. 2017. "The Program Is the System: Introduction to Programming Without Abstraction." In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 138–42. Koli Calling '17. New York, NY, USA: ACM. <https://doi.org/10.1145/3141880.3141894>.

Hemmendinger, David. 2010. "A Plea for Modesty." *ACM Inroads* 1 (2). New York, NY, USA: ACM: 4–7. <https://doi.org/10.1145/1805724.1805725>.

Horn, Michael S., and Robert J. K. Jacob. 2007. "Designing Tangible Programming Languages for Classroom Use." In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, 159–62. TEI '07. New York, NY, USA: ACM. <https://doi.org/10.1145/1226969.1227003>.

Jaime, Sanchez, and Olivares Ruby. 2011. "Problem solving and collaboration using mobile serious games." In *Elsevier Ltd*.

Kafai, Yasmin, and Q. Burke. 2013. "Computer programming goes back to school." In *Phi Delta Kappan*.

Kahn, Ken. 2017. "A half-century perspective on Computational Thinking." In *Technologias, Sociedade E Conhecimento*.

Kastl, Petra, Ulrich Kiesmüller, and Ralf Romeike. 2016. "Starting Out with Projects: Experiences with Agile Software Development in High Schools." In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, 60–65. WiPSCE '16. New York, NY, USA: ACM. <https://doi.org/10.1145/2978249.2978257>.

Kay, Alan C. 1993. "The Early History of Smalltalk." *SIGPLAN Not.* 28 (3). New York, NY, USA: ACM: 69–95. <https://doi.org/10.1145/155360.155364>.

- Kay, A., K. Rose, D. Ingalls, T. Kaehle, J. Maloney, and S. Wallace. 1997. "Etoys & SimStories." *Walt Disney Imagineering*.
- Kurtz, Thomas E. 1978. "BASIC." *SIGPLAN Not.* 13 (8). New York, NY, USA: ACM: 103–18. <https://doi.org/10.1145/960118.808376>.
- Lewis, Colleen, Sarah Esper, Victor Bhattacharyya, Noelle Fa-Kaji, Neftali Dominguez, and Arielle Schlesinger. 2014. "Children's perceptions of what counts as a programming language." In *Journal of Computing Sciences in Colleges*.
- Lonati, Violetta, Mattia Monga, Anna Morpurgo, and Mauro Torelli. 2011. "What's the Fun in Informatics? Working to Capture Children and Teachers into the Pleasure of Computing." In *Proceedings of Issep2011*, edited by I. Kalaš and R.T. Mittermeir, 7013:213–24. Lecture Notes in Computer Science. Springer-Verlag. https://doi.org/http://dx.doi.org/10.1007/978-3-642-24722-4_19.
- Lye, S.Y., and J.H.L. Koh. 2014. "Review on teaching and learning of computational thinking through programming: What is next for K-12?" In *Computers in Human Behavior*.
- Maloney, John, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. "The Scratch Programming Language and Environment." *Trans. Comput. Educ.* 10 (4). New York, NY, USA: ACM: 16:1–16:15. <https://doi.org/10.1145/1868358.1868363>.
- Martini, Simone. 2012. "Lingua Universalis." *Annali della Pubblica Istruzione* 4-5. Le Monnier: 65–70. <https://hal.inria.fr/hal-00909609>.
- Meerbaum-Salant, O., M. Armoni, and M. Ben-Ari. 2010. "Learning computer science concepts with scratch." In *Proceedings of the Sixth International Workshop on Computing Education Research*, 69–76.
- Missiroli, Marcello, Daniel Russo, and Paolo Ciancarini. 2016. "Learning Agile Software Development in High School: An Investigation." In *Proceedings of the 38th International Conference on Software Engineering Companion*, 293–302. ICSE '16. New York, NY, USA: ACM. <https://doi.org/10.1145/2889160.2889180>.
- Papert, Seymour. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York, NY, USA: Basic Books, Inc.
- Papert, Seymour. 1996. "An Exploration in the Space of Mathematics Educations." *International Journal of Computers for Mathematical Learning* 1 (1): 95–123. <https://doi.org/10.1007/BF00191473>.
- Papert, Seymour, and Idit Harel. 1991. "Constructionism." In. Ablex Publishing Corporation.
- Pattis, Richard E. 1981. *Karel the Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons.
- Proulx, Viera K. 2000. "Programming Patterns and Design Patterns in the Introductory Computer Science Course." In *ACM Sigcse Bulletin*, 32:80–84. 1. ACM.
- Qian, Yizhou, and James Lehman. 2017. "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review." *ACM Trans. Comput. Educ.* 18 (1). New York, NY, USA: ACM: 1:1–1:24. <https://doi.org/10.1145/3077618>.
- Resnick, Mitchel. 1996. "Distributed Constructionism." In *Proceedings of the 1996 International Conference on Learning Sciences*, 280–84. ICLS '96. Evanston, Illinois: International Society of the Learning Sciences. <http://dl.acm.org/citation.cfm?id=1161135.1161173>.
- Resnick, Mitchel 2007. "All I Really Need to Know (About Creative Thinking) I Learned (by Studying How Children Learn) in Kindergarten." In *Proceedings of the 6th Acm Sigchi Conference on Creativity & Cognition*, 1–6. C&C '07. New York, NY, USA: ACM. <https://doi.org/10.1145/1254960.1254961>.
- Resnick, Mitchel 2014. "Give P's a Chance: Projects, Peers, Passion, Play." In *Constructionism and Creativity: Proceedings of the Third International Constructionism Conference*. Austrian Computer Society, Vienna, 13–20.

Resnick, Mitchel 2017. *Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play*. MIT Press.

Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, et al. 2009. "Scratch: Programming for All." *Commun. ACM* 52 (11). New York, NY, USA: ACM: 60–67. <https://doi.org/10.1145/1592761.1592779>.

Roberts, Eric S. 1995. "Loop Exits and Structured Programming: Reopening the Debate." In *ACM Sigcse Bulletin*, 27:268–72. 1. ACM.

Sajaniemi, Jorma. 2002. "An Empirical Analysis of Roles of Variables in Novice-Level Procedural Programs." In *Human Centric Computing Languages and Environments, 2002. Proceedings. IEEE 2002 Symposium on*, 37–39. IEEE.

Salen, Katie, and Eric Zimmerman. 2003. "Rules of Play - Game Design Fundamentals." In *the MIT Press Cambridge Massachusetts*.

Schön, S., M. Ebner, and S. Kumar. 2014. "Implications of new digital gadgets, fabrication tools and spaces for creative learning and teaching." *Special Edition*.

Sentance, S., and A. Csizmadia. 2015. "Teachers' perspectives on successful strategies for teaching Computing in school." In *IFIP TC3 Working Conference*.

Sheth, Swapneel Kalpesh, Jonathan Schaffer Bell, and Gail E. Kaiser. 2012. "Increasing Student Engagement in Software Engineering with Gamification." *Columbia University Computer Science Technical Reports*.

Sirkiä, Teemu. 2012. "Recognizing Programming Misconceptions: An Analysis of the Data Collected from the Uuhistle Program Simulation Tool." Master's thesis, Department of Computer Science; Engineering, Aalto University.

Slany, W. 2014. "Tinkering with Pocket Code, a Scratch-like programming app for your smartphone." In *Proceedings of Constructionism 2014*.

Sorva, Juha. 2013. "Notional Machines and Introductory Programming Education." *Trans. Comput. Educ.* 13 (2). New York, NY, USA: ACM: 8:1–8:31. <https://doi.org/10.1145/2483710.2483713>.

Taub, Rivka, Michal Armoni, and Mordechai Ben-Ari. 2012. "CS Unplugged and Middle-School Students' Views, Attitudes, and Intentions Regarding CS." *TOCE* 12 (2): 8. <https://doi.org/10.1145/2160547.2160551>.

Tumlin, Nath. 2017. "Teacher Configurable Coding Challenges for Block Languages." In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*.

Weintrop, David, and Uri Wilensky. 2015. "To block or not to block, that is the question: students' perceptions of blocks-based programming." *IDC '15 Proceedings of the 14th International Conference on Interaction Design and Children*.

Wing, Jeannette M. 2006. "Computational Thinking." *Commun. ACM* 49 (3). New York, NY, USA: ACM: 33–35. <https://doi.org/10.1145/1118178.1118215>.

Wing, Jeannette M. 2008. "Computational thinking and thinking about computing." In *Philosophical Transactions of the Royal Society*.

Wing, Jeannette M 2010. "Computational Thinking: What and Why?" *Link Magazine*.

Wirth, N. 1993. "Recollections About the Development of Pascal." *SIGPLAN Not.* 28 (3). New York, NY, USA: ACM: 333–42. <https://doi.org/10.1145/155360.155378>.

WG7: Constructionism in Upper Secondary and Tertiary Levels

Ana Isabel Sacristán, *asacrist@cinvestav.mx*

Center for Research and Advanced Studies (Cinvestav), Mexico

Lina Kaminskienė, *l.kaminskiene@pvt.vdu.lt*

Vytautas Magnus University, Lithuania

Mihaela Sabin, *Mihaela.sabin@unh.edu*

University of New Hampshire, USA

Richard Akrofi Kwabena Baafi, *richbaafius@yahoo.com*

Eotvos Lorand University, Hungary

Abstract

In the constructionist paradigm, the fundamental premise is to create student-centred learning situations for students to consciously engage in constructing shareable, tangible objects, through meaningful projects. In Papert's vision, one particularly valuable means of doing that is in programming the computer because, in doing that, the student "establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building" (Papert, 1980, p. 5). Since the 1980s, there have been countless experiences and studies exploring and documenting the use of the constructionist paradigm, many of the first ones using Logo computer programming, but mostly with young students at primary or middle-school levels. However, experiences in upper secondary and university levels are scarcer. Laurillard (2002), nonetheless, advocates for constructionist and collaborative technology-based learning environments in higher education; she says (p. 42): "the aim of university teaching is to make student learning possible [...] not simply impart decontextualised knowledge, but must emulate the success of everyday learning by situating knowledge in real-world activity" helping students reflect on their experience of the world and ways of representing it.

The purpose of this working group was to share constructionist experiences in upper secondary and tertiary educational levels, particularly those involving computer programming and/or computational thinking and environments; and to reflect on the challenges, needs and differences of constructionist technology-based implementations in the various educational levels, and on how to promote such implementations in upper levels.

The guiding questions for the working group were: 1. What are the characteristics of constructionist implementations in upper educational levels? In upper secondary school? At university level? How are they different from lower levels? What are the particular challenges? 2. How is, or what could be, the role of digital technologies and computer programming in such implementations? 3. How can constructionist implementations be integrated and promoted in higher education? What is required for that? 4. Can real-life data, phenomena and problems be harnessed for developing such implementations?

Keywords

Constructionism; upper secondary level; tertiary level; learning environments; project-based learning; computer science or technology-enhanced learning

Introduction

Despite countless experiences and studies exploring and documenting the use of the constructionist paradigm, most of them have been with young students at primary or middle-school levels. However, experiences in upper secondary and university levels are scarcer. But there are advocates (e.g. Laurillard, 2002) for constructionist and collaborative technology-based learning environments in higher education taking into account how students learn. Laurillard (2002) considers that “the aim of university teaching is to make student learning possible [...] not simply impart decontextualised knowledge, but must emulate the success of everyday learning by situating knowledge in real-world activity” (p. 42) helping students reflect on their experience of the world and ways of representing it. However, another issue is that implementing constructionist exploratory learning environments in school cultures (at any level) is problematic and complex, as has been discussed also by Laurillard (2002) and others.

The members of this group have all been involved with constructionist or innovative learning experiences and environments in secondary school, university level, and even postgraduate level. We are interested in contributing and sharing with others, practices and models of applying constructionism in different educational contexts and levels, as well as help expand the foundation and background of this area for research.

We all range from very different backgrounds and fields. Some of us are involved only in high-school and upper secondary level, while others' work is more centred at university level, and even graduate level and even adult education. Some of us work in STEM education (mathematics, science and engineering), others in business and language education, while others are more specifically involved in computer science education. And some of us come from developing countries (Ghana or Mexico) and others from developed countries (USA or Lithuania), and it is clear that there is a technological gap between these. Nevertheless, as is exemplified in this paper, we all have commonalities working in constructionist environments with mostly project-based and student-centred learning; also, most of us are interested in issues such as the affective aspect, for instance in how constructionist tools in post-secondary computing education shape learners' dispositions to engage with peers on complex and collaborative projects.

Key elements of constructionism

One task that we did in the group, was to compile some of the key elements of constructionism that we consider as fundamental or that we have used in the learning environments with which we have been involved.

As I expressed in the plenary paper for this conference (Sacristán, 2018), the fundamental premise of the constructionist paradigm, as stated by Papert and Harel (1991), is that it shares “constructivism's connotation of learning as ‘building knowledge structures’ [...] then [adds] the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity” (p. 1), that is, something shareable.

Papert (1980) also placed emphasis on a different kind of learning culture, where:

- educators act as support for learners to build their own intellectual structures;
- create conditions for construction and invention (rather than providing ready-made knowledge),
- give students objects-to-think and materials drawn from the surrounding culture,
- there is an “emotionally supportive working conditions [that] encourage them to keep going” (p. 197), with
- students having creative and personally defined end-products that they can genuinely be excited about.

This latter point is the affective dimension which is a fundamental component of Papert's constructionist paradigm. Papert considered that affect had an indispensable role in creating and sustaining learners' intellectual engagement (Papert, 1980/1993). And as we will see further below, this affective dimension is an important motivation for most of us in the working group.

Papert (1980) also proposed to use the versatility of computers to create microworlds which he defined as incubators for knowledge and powerful ideas; that is, “self-contained world[s], [...] each with its own set of assumptions and constraints”, where learners “get to know what it is like to explore the properties of a chosen microworld undisturbed by extraneous questions [...] and] learn to transfer habits of exploration from their personal lives to the formal domain of scientific theory construction” (p.117). “places” where these ideas can easily hatch and grow.

In particular, Papert (1980) promoted computer programming as a means for learning, explaining that “in teaching the computer how to think, [students] embark on an exploration about how they themselves think” (p.19). He also emphasized the value of debugging, stating that errors are of benefit because they lead to the need to understand what went wrong, and through that understanding, to fix them (Papert, 1980).

Analysing some of the main descriptions of constructionism, in that paper (Sacristán, 2018), I summarised there that I felt a constructionist learning environments could be defined as:

- student-centred learning situations
- where students consciously engage in constructing (e.g., program) shareable, tangible objects,
- through creative personally meaningful projects (e.g. computer-based)
- where they have objects-to-think-with, access/develop powerful ideas,
- and have opportunities for explorations,
- and thinking about their own thinking (analysing) through debugging (fixing).

In that paper (Sacristán, 2018) added, that in my opinion, ideally,

- there should be a medium (e.g. digital tools) for an exploratory and expressive activity (such as computer programming/coding, or building/describing models or structures using an expressive medium or software).
- Students need to be actively involved and mostly in charge of their constructions and explorations.
- Activities should take place within a structured of learning environments (microworlds) and collaborative learning environment with the characteristics described above.

As described further below, the projects that the members of the working group have engaged in, most of them have promoted:

- Project-based activities and learning
- Engaging in authentic or real-life based problems. Some cases have included experiences or work aspects related to future professions or workplace-bound and relevant aspects of work, including professional types of digital tools
- Engaging in making personally-relevant products (such as programming computational artifacts or “apps”, or innovative projects)
- Flexibility (e.g. flexible content)
- Developing personal strategies
- Sharing the resulting products
- Using communication and collaboration platforms (including those that professionals use)
- A guiding engagement of teachers and even future employers providing expert mentorship

In the following sections, we provide examples of the projects and experiences that each of the team members have been involved with and that have fulfilled most of the above characteristics. Although not all the examples are from upper secondary school and tertiary levels (some are from lower secondary schools) we hope that we can learn lessons from those examples.

Examples of constructionist experiences in secondary schools and tertiary level

Focussing on the school culture and learning environment in high-schools in Ghana

A first example, by the co-author of this paper Baafi, is an experience of constructionism in upper secondary school, which focuses on how the lessons, learning environment and classroom culture are organized. It is an example that doesn't necessarily focus on the use of computers or digital technologies, but rather on a constructionist culture in the classroom

Baafi has been teaching and researching the impact of school learning environments on students' performance in senior high schools in Ghana. Taking into account that the various types of instructional strategies that teachers use in the classroom influence significantly students' learning (Kaya et al., 2011; Onweh & Akpan, 2014), it is important for the teacher to organize the lesson (and accordingly select appropriate teaching materials) in a manner that enhances students' learning and understanding. Thus, Baafi considers the teacher to be an "architect" in selecting the best ways for helping students learn and generate knowledge through the teaching-learning process. He also considers that an important way in which constructionism can be implemented at the upper secondary level is through project-based learning and computer applications, in an activity designed to help develop innovative skills in learners.

In this sense, in the high-schools in Ghana where Baafi has been teaching, constructionism is used as a teaching and learning strategy to provide useful learning experiences for students, and help them to develop and demonstrate their abilities in solving dilemmas in complex tasks (Kynigos, 2015). Most students share knowledge through project-based learning. This offers them the opportunity to practically connect classroom learning to a given activity where they implement what they learn. The content of the lesson is prepared addressing students' learning needs in order to select appropriately relevant teaching techniques, so that students can engage and be motivated by the tasks and find innovative solutions. Thus, teachers create a learning environment which employs different teaching-learning methods such as presentations, discovery, inquiry, discussions and team-teaching in order to help students construct creative learning skills in solving problems (see Figures 1-3). Students may also be arranged in groups, to encourage collaborative learning. Generally, students enjoy generating shared ideas with their colleagues during learning activities (Olsen et al., 2018). Teachers may also facilitate collaborative discussions by having students ask questions to bring about debates on ideas, which also helps build a learning relationship between the teacher and the learners. In this way, students develop collaborative learning strategies through sharing and constructing ideas.



Figure 1. A classroom discussion session in a high-school in Ghana



Figure 2. A team collaborative project presentation on ICT



Figure 3. Team-teaching in the high-school in Ghana

Emphasis is thus placed on a learning environment that motivates students. For instance, some of Baafi's high-school students were able to solve a task that appeared difficult initially, after a field-trip has offered the students new ideas where new ideas emerged and which allowed them to collaborate in finishing their term's project. Thus, the growth of students' capabilities allows them to manage their learning environment in order to realise their potentials fully (Viray, 2017). Likewise, the teacher can be an agent of motivation by engaging students in tasks that can test their potentials, helping them become conscious of their capabilities and thus facilitating them to develop their cognitive capacity.

Technology-enhanced personalisation strategies for learning

Just as Baafi emphasizes the learning culture and environment, designing scenarios that empower students, so is Kaminskienė's case, whose experience with the constructionist approach has been mainly in technology-enhanced learning using personalisation strategies in tertiary level and adult education (e.g. in business organisations) (Kaminskienė, Trepulė, Rutkienė, Arbutavičius, 2014; Teresevičienė et al., 2015; Kaminskienė, Rutkienė, & Trepulė, 2015); and to a lesser extent in secondary level. Kaminskienė defines personalisation as something that allows developing different learning scenarios with the facilitation of the teacher; however, it is most important that the learning scenario be constructed mainly by the learner. The tools for the implementation of personalized learning should address the core personalized learning components and characteristics described by Johns and Wolking (2015):

- Student reflection and ownership (scenarios),
- Flexible content (restructured learning/teaching material),
- Tools, and learning environments.

These characteristics are important for the implementation of constructivist implementations at secondary or university level.

Personalisation is usually based on several criteria: the most typical ones are relate to the current level of knowledge, a learner's motivation and learning aims, learning styles, digital competences and others. These criteria relate to the important aspect that Papert (1980) emphasized of student-centred learning, and of designing a learning culture where learners can carry out their own meaningful projects. They also relate to the extended version of the concept of microworld by Hoyles and Noss (1987) in which the student's knowledge and component is central along with the other components: the technological one, the pedagogical design and a new role of teachers, and a social environment fostering collaboration and where products can be shared and collaboratively discussed.

Kaminskienė explains that as the education level changes (secondary or tertiary), learners grow in autonomy. and the skills required in personalisation can be systematically developed. Learning autonomy should not be understood as learning alone, nor that collaborative learning is excluded. On the contrary, personalisation can improve roles and responsibilities in group work.

In a research case study that Kaminskienė and colleagues carried out, fifteen, 8th grade, students in a Lithuanian secondary school engaged in digital storytelling, during English language lessons, as a

technology based pedagogical tool, used to provide student-centered learning strategies. Digital storytelling is project-based learning that enables learners to organize and present their individual stories, bringing more personal essence and improving learners' engagement. Digital storytelling is a constructivist approach in the classroom, that increases peer-to-peer teaching and collaboration. It gives students a chance to develop their English language and investigate various e-tools that can help them express and digitalise their stories.

The research indicated that levels of student engagement fluctuated between moderate and high. The use of software and conducting searches for digital media, took these levels to very high. In all cases students liked using technology, searching the internet, and watching other's digital stories. Data from feedback questionnaires indicated that 80% of the students consider this type of learning different from traditional classroom.

Throughout the study, students worked collaboratively and engaged with digital content. They did more work while working in groups and directly using applications and digital resources. This research also observed collaboration between groups where different groups helped each other with technical issues. This increased their levels of communication. Students have stated that they much enjoyed working within their groups and asked for more group-work on each lesson. The findings revealed, however, that students could better understand their individual contribution to the team work if they are well instructed by teachers and have flexibility to work with the tools and content.

Results demonstrate (see 0), that 50% of the group consider this type of learning very interesting, 80% think it differs from other lessons and they outline that learning process through Digital Storytelling was interactive and funny. 71% of children outlined that they have improved ICT skills while working on their digital stories, while 50% confirmed the improvement of communication skills. 36% thinks team-working skills were also improved throughout this interaction. All children mentioned that they totally enjoyed working within their teams, as it helped them to collaborate better.

	Team Working Skills	ICT Using Skills	English Language Skills	Creative Skills	Public Speaking Skills	Responsibly Working Skills	Communication Skills	Responses
All Data	5 (36%)	10 (71%)	5 (36%)	4 (29%)	3 (21%)	2 (14%)	7 (50%)	14

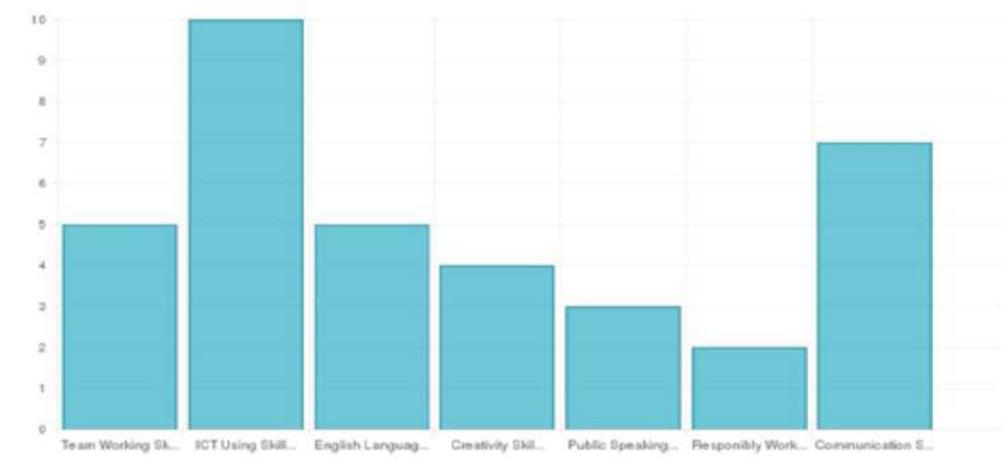


Figure 4. Data from secondary students engaging in digital storytelling for English learning

A summary of some of the successful aspects of the above experience, include:

- More engaged and motivated students (80% of students indicated that working collaboratively while developing digital stories was an engaging experience).
- Broader possibilities to personalise learning (technologies allow to differentiate curriculum according to different characteristics of learners yet striving at the end achieving the planned learning outcomes).
- Developing multiple competences (in the above case, in the English language); skills in using technology-enhanced tools; and engaging in teamwork.

On the other hand, some challenging aspects included:

- More focus on the form, rather than on the content. For example, the emphasis on digital storytelling could create a misunderstanding that the form (the e-tools and digital materials used for the development of digital stories are more important than the content (the problem, idea, etc.). In fact, 71% of students indicated that they improved their ICT skills vs. 36% who commented that they had improved in their English language skills..
- More regular feedback is required in cases of personalised learning, project-based learning, which is a challenge for teachers.

Constructionist implementations related to STEM fields in higher education

Sacristán has been involved also in several constructionist projects in higher education in Mexico and Canada, where university students engage in computer programming and/or expressive activities (involving topics and data related to real-life phenomena, meaningful for their area of study) for mathematical exploration or learning that usually include sharing, collaboration and discussion. These are described in more detail in Sacristán (2017a) as well as in the plenary paper for this conference (Sacristán, 2018).

One project involved a distance-learning environment (a virtual laboratory) where university and continuing education students engaged in exploratory modelling tasks of various types of real-life mathematical problems students (Olivera, Sacristán & Pretelín-Ricárdez, 2013). The tasks, carried out through various digital means and interactive tools, involved building models through collaboration; sharing, discussing and reflecting through the online platform; and even proposing new explorations. The online exchanges constituted additional means for learning, since they forced students to express their ideas, thus helping them clarify their own understandings. Although this project of a virtual mathematics laboratory was successful, after a couple of years it could not be continued due to the differences with the established curricula, which points to the challenges of implementing constructionist approaches in institutional settings.

Another experience in which Sacristán was involved, was one where university engineering students engaged in producing working models of certain real-life behaviours (e.g. of water behaviour or simulated mechanical systems) within videogames (Sacristán & Pretelín-Ricárdez, 2017). One of the aims of this project was for students to develop know-how for their future profession as engineers on how to apply mathematical knowledge and modelling. The videogame constructions involved several stages of model-building tasks that included some collaborative programming work and whole class discussions. The projects were meaningful personal projects, so were highly motivating for the students. This experience was so successful that it is became a regular course in the engineering program.

In a third project (Mascaró, Sacristán & Rufino, 2014), undergraduate and post-graduate environmental sciences and biology students engage in sequences of constructionist and collaborative, computer-programming tasks in the R programming language, for the learning of probability, statistics and experimental analysis concepts. These tasks, presented through R-code “worksheets” with guidelines and examples, use data adapted from real research situations, so that they are meaningful to students. Research findings point in particular to improvements in the affective dimension, with many students losing their fear of statistics, and being motivated to engage in the tasks; moreover, many students appropriated themselves of the software (e.g. building their own R scripts) for their own research with an apparent clearer understanding of statistical concepts. This statistics programme has been integrated into regular courses for several years but only by the one teacher who designed the sequences of tasks.

A final project in which Sacristán has been involved, is the Mathematics Integrated with Computers and Applications (MICA) program at a Canadian university (see Muller, Buteau, & Sacristán, 2015; Buteau, Sacristán & Muller, 2018). In the MICA program, university students, in addition to traditional mathematics courses, have non-traditional courses where they engage in programming, using VB.net, their own interactive mathematical digital environments, also called Exploratory Objects (or EOs), through which they explore their own stated mathematical conjectures, theorems, or real-world situations, and need to engage in various types of modelling of the mathematical ideas they explore and program. This is an example of a constructionist program that has been sustained successfully as

part of that university's curriculum for over 15 years, although some conservative faculty members have challenged it.

Students adapting and building apps for Computer Science learning

Sabin's experiences are in computer science education mainly at university and post-graduate levels in the USA, but also with high-school girls. In particular she has been involved in developing constructionist learning environments in which CS students adapt or build apps, such as with a constructionist tool, QuizPower, described further below.

Sabin considers that constructionist learning environments should use innovative tools that are accessible to beginners with no prior disciplinary knowledge (low floors), but are powerful enough to be of interest and meaningful to expert users too (high ceiling). Another premise is that these tools should facilitate connections to personal knowledge and experiences of all students, who bring in very diverse cultural and life backgrounds (wide walls). In CS/IT degree programs at university level, constructionist implementations involve computing tools and platforms that professionals use in the workplace. There are also tools specifically designed to engage learners from a low ceiling entry point to a high ceiling of compellingly complex problems and projects (e.g., pythontutor.com for Python programming, App Inventor for app development, Raspberry Pi for physical computing, or Tableau for dashboards) There is some prerequisite knowledge that is needed for using these tools, such as run-time environment, execution flow, stack frame, object encapsulation, event-driven programming.

In any case, it is important that the innovative tools should make ideas and abstractions learnable by building computational artifacts from which students transfer familiar intellectual constructs into understanding new powerful ideas and more formal computing concepts. Block-based programming platforms, such as Scratch and App Inventor, are constructionist tools that turn programming and computational artifacts into construction materials with which students learn abstractions through discovery with concrete objects. Thus, learners follow Papert's call to "make something new" with programming, "play with it, build with it" (Papert 1993, p. 120). QuizPower (see Figures 5 & 6) is an example of a constructionist tool built with App Inventor (Meehan and Sabin, 2013). The app provides beginner students or non CS majors a way to learn computing fundamentals in an engaging and effective manner. More advanced students can adapt the app to create quizzes for other courses by designing a new web data store and gain more practice and deeper understanding of client-server programming and databases. Students in a general education, introductory mobile app development course for all majors played a vital role in giving feedback and informing the design of the quiz questions and their organization. Computing students in more advanced CS courses can use the app to learn more about web service development and deployment using App Inventor, PHP, and MySQL and the integration of App Inventor development and packaging service with full-stack execution environments, such as XAMPP.

Sabin's experiences in post-secondary education includes computing curriculum development, project-based learning with client-driven open source software, assessing collaborative and experiential learning, and creating the constructionist tool described above, QuizPower.

Constructionism as a teaching practice in computing education situates learners and teachers together in a project-based learning environment in which students make and share computational artifacts with the help of innovating computing tools and modes of media for the purpose of learning and mastering computing concepts and professional practices.

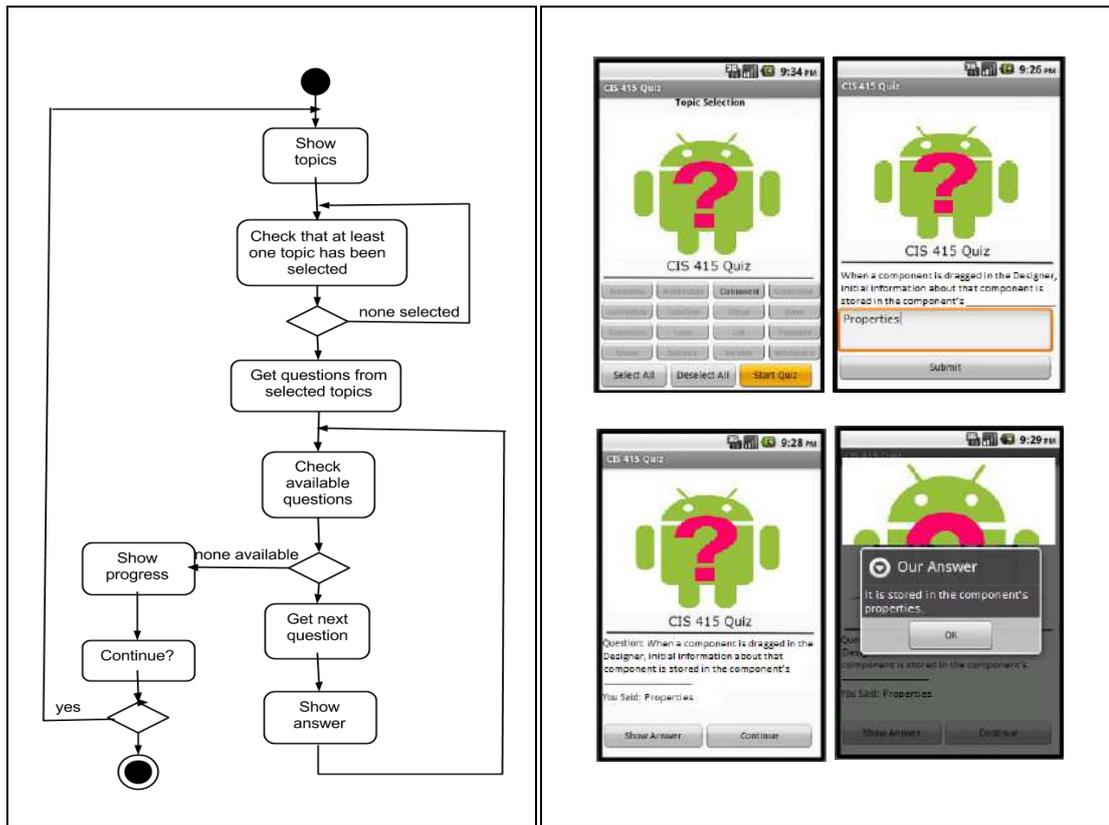


Figure 5. Activity diagram for the QuizPower user interface



Figure 6. QuizPower user experience example

The importance of situating learning activities in supportive contexts is taken into account in the ACM/IEEE Task Group on Information Technology Curricula (2017) report that defined curricular IT guidelines for undergraduate programs. The report proposes an operational definition of IT competencies as a triad of three interplaying dimensions, knowledge, skills, and dispositions, that do not manifest in a vacuum or, what Papert calls "instructionist" environment, where formal and impersonal learning is imparted through lecturing and tutorials. Instead, developing and demonstrating IT competencies require an intentional context that is personally meaningful and conducive of direct interactions between learners and concrete things or "objects-to-think-with" (Papert 1993, p.168). Key characteristics of the context in which students form IT competencies are described by Sabin, Alrumaih and Impagliazzo (2018) and include:

- Workplace-bound experiences and relevant IT aspects of work
- Active involvement of employers to support internship and co-op programs and to provide expert mentorship
- Authentic problems and engagement of diverse teams in relevant IT aspects of work
- Collaborative and project-based activities leveraged by using professional tools
- Deliberate and critical reflection on practice to participate effectively in decision-making and to stay engaged in a process of continuous learning.

Some examples of IT learning situated in constructionist contexts are practicum and internship experiences, projects with real clients, reflective journals of individual contributions to a complex team project, technical presentations judged by external partners, and capstone projects with external evaluators.

In a framework proposed and discussed by MacKellar, Sabin, and Tucker (2013), client-driven open source projects have the quality of representing the "source of concepts to think new ideas" that

constructionist projects, in Papert's view, should have (Papert, 1993, p. 168). They also afford students the opportunity to experience new, state-of-the-art technologies from the vibrant and invested free and open source software community, which creates excitement and forms a sense of belonging to the computing profession (Sabin, 2011). With real clients who interact closely and regularly with the students during the project development process, a culture of sharing, reflection, critique, and discussion is almost inescapable. In implementing the proposed framework by MacKellar, Sabin & Tucker (2013), by three different higher education institutions, an advantage was that the clients were non-profit organizations who accommodated learners' "false theory building" through experimentation, risk taking, and failures. Two institutions used an open source codebase and architecture that had been previously developed in an academic course at another institution, which helped with striking a desirable balance between relying on a traditional self-contained toy project and engaging students in an industrial-strength full-scale open source project. The experience report recommends that schools develop and share exemplary projects that could be added to a collection of client-driven open source projects. Client-driven open source software projects create opportunities for tackling complex computing problems with higher level of engagement among diverse students.



Figure 7. Students engaged in designing and building apps

At the secondary level, Sabin has studied how to increase confidence and interest in computing for girls. According to a report by Google (2014), encouragement to study computer science and connecting personal interests to computing areas influence women to pursue degree in computing. Guided by these observations, a one-week summer camp for girls in grades 7-9 was designed to improve girls' perceptions of computer science (Sabin, Deloge, Smith & DuBow, 2017). The camp curriculum incorporated inquiry and equity teaching practices and purposefully combined teaching of computing content with culturally responsive pedagogies (Margolis et al., 2014). By using App Inventor to build mobile apps, the girls experimented with component-based design and event-driven implementation; shared among themselves app designs through gallery walks; worked in pairs to test and refine their apps and media assets; and expressed their own interests in personally meaningful apps that told stories about the girls' favorite places or hobbies. The analysis of a pre- and post-survey found that the camp appeared to have positively influenced the girls, primarily in terms of improving computing confidence and positive perceptions of computing careers (Sabin et al., 2017).

Commonalities of the above experiences

As can be seen in the above examples, there are many commonalities to the above experiences, as are already described in the final list of key characteristic of constructionism given in the first section of this paper, such as:

- Project-based learning: Almost all the above experiences involve this, whether it is projects in Ghana, or digital story telling, or constructions, or programming videogames, EOs or apps.
- These products or artefacts imply making personally-relevant products, which is motivating and deals with the affective aspect. This affective component is all of the examples discussed above.
- It also implies creating learning environments and using personalised learning strategies
- with emphasis on the importance of the classroom culture, where
- students are empowered and teachers are more guides and mentors rather than "instructors". And where
- students engage in collaborative work

- With many also involving using authentic or workplace-bound problems (such as the authentic tasks in the virtual mathematical laboratory, or for some of the videogame designs, or for the EOs, or as client-driven project) or work bound tools (such as R, VB.net and others mentioned by Sabin)

Addressing some of the guiding questions and some conclusions

The guiding questions for the working group were:

- What are the characteristics of constructionist implementations in upper educational levels? In upper secondary school? At university level? How are they different from lower levels? What are the particular challenges?
- How is, or what could be, the role of digital technologies and computer programming in such implementations?
- How can constructionist implementations be integrated and promoted in higher education? What is required for that?
- Can real-life data, phenomena and problems be harnessed for developing such implementations?

In terms of the first question, at the beginning of the paper, we have already stated some of what we consider are key characteristics of constructionism. But an additional aspect that was addressed by the working group members, was that as the educational level increases, so does the autonomy of the students, and also that at university level learners are more self-regulated. This may help when implementing student-centred and project-based learning.

In terms of the last question, many of the above projects integrate authentic tools, problems or data, as described in the previous section.

As to the other questions, these are somewhat addressed in the following section.

Some detected challenges and possible strategies for more successful constructionist implementations in higher educational levels

Constructionism is still an innovative approach in the education, and many typical barriers still exist, including those related to the integration of technologies in general (see Sacristán, 2017b), and others such as:

- Regulatory and institutional requirements;
- The barrier to change ways of working, which is one of the greatest barriers to innovation, and which Sacristán calls the inertia of school cultures
- Instructor's role as mentor and learner is counter learners' common expectation that instructors impart knowledge and have all the answers.
- Course evaluations are designed with a preset course curriculum in mind (course objectives) and a 'sage on the stage' instructor model (answered questions effectively)
- Fear of using technologies; fear of technological risk.
- Increasing digital gaps between the digital and networked society and the traditional curriculum of educational systems. As well as in terms of social, economic and geographical stratification.
- Lack of personal leadership;
- Lack of institutional support.
- Technocentrism: focus on the technology, rather than on the subject matter and content
- More regular feedback is required in cases of personalised learning, project-based learning, which is a challenge for teachers.
- The difficulty of transference (e.g. of "constructionist methodologies or projects")
- Collaborative projects need to mediate diverse personal interests
- Assessing student learning in constructionist learning environments
- Another barrier to implementing constructionist learning environments that use project-based learning in university upper-level courses relates to deemphasizing grading and evaluation of computational artifacts. If computational artifacts contribute to a high-stakes real-world project,

evaluating student performance is predominantly determined by the quality of work products rather than participatory activities and collaborative processes.

In order to integrate constructionist implementations be integrated and promoted in higher education, it is clear we have to develop an appropriate learning culture in higher education institutions. University degree programs as a whole should enjoy a constructionist culture. They have to be integral part of the program curricula. So there needs to be a shift in the culture.

Innovations in how school cultures work may be needed, but also it may be that compromises need be made between the utopic constructionist visions (such as those of Papert) and institutional requirements. For example, in Mascaró & Sacristán (2018), in these conference proceedings, we give an example of “constructionist-compatible” ways that were developed to assess students’ learning, in order to grade them for the university courses in which they are enrolled.

There may also need to be considerable investments (time, human resources training) and work/preparation of students. The implementation of constructivist approaches in tertiary level may be related to technology enhanced learning. Nevertheless, as has been noted by researchers (e.g. Schneckenberg, 2010), technology-driven innovation in universities depends much on active involvement of the faculty in organisational change. Schneckenberg (2010) speaks of the need of institutions to develop a so-called eCompetence model, which is understood as a complex unit of personal and institutional dimensions (covering micro, meso and macro levels); the eCompetence model is composed of competence, academic staff, competence development measures, eStrategy in universities, and pervasive potential of ICT in educational contexts.

In the case of computing education in higher levels, Sabin points out that Psenka and colleagues (2017) discussed the applicability of constructionism to create learning environments for engineering design and offered a set of observations to be considered by the engineering design education. All observations point to conditions and strategies that are instrumental to designing constructionist learning environments for computing education in higher education. For example, project-based learning and inclusion of real-world problems have been largely adopted in the undergraduate curricula in the USA. The literature is vast and findings are compelling to continue on this path. In the USA, the ABET Computing Accreditation Commission updated the Computer Science program accreditation criteria to reflect curriculum guidelines for undergraduate computer science programs and includes a student outcome for learning to “apply design and development principles in the construction of software systems of varying complexity” (ABET, 2018), supporting the guideline that all CS graduates be involved in at least one substantial project.

Sabin also points out, that in the case of Computer Science degrees, there are tools used widely by computing professionals (e.g. GitHub, a version control platform; Slack, a communication tool; and Amazon Web Services, a cloud deployment service); Sabin sees all three as constructionist tools that a CS/IT degree program should integrate across program curricula.

In terms of collaborative projects of significant scope and scale and possibly involving real clients, there are, however, considerable challenges. Such projects constrain the playfulness of more relaxed environments in which students have conditions to be “let loose”, and unplanned, emergent learning is more likely to happen. There is also the tension between giving students the freedom to select their own projects and having adequate mentoring capacity in large classes with too many individual projects.

We end with some questions of interest to implementing constructionism in higher education:

- How to let undergraduate students loose to create computational artefacts as learning materials when constrained by grades and highly specialized curriculum in upper level courses?
- What curricular changes of undergraduate courses are needed to facilitate natural and spontaneous ways of making ideas concrete in the classroom?
- How to design or adapt innovative computing tools with which students can have multiple ways of knowing through senses, movement, and other interactions to make concepts and theories “simple and concrete” (Papert 1980/1993, p.7)?

- What preparation and learning do teachers need to change the learning culture in the classroom and take into account the learners' affect and their dispositions to tinker, embrace persistent struggle, and reflect upon their learning experiences?

References

- ABET (2018). *Criteria for Accrediting Computing Programs 2018-2019*. <http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2018-2019/>
- ACM/IEEE Task Group on Information Technology Curricula (2017). *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology (IT2017)*. ACM (Association for Computing Machinery) and IEEE Computer Society. doi:10.1145/3173161
- Buteau, Sacristán & Muller (2018). Teaching in a Sustained Post-Secondary Constructionist Implementation of Computational Thinking for Mathematics. In *Proceedings Constructionism 2018*, Vilnius, Lithuania.
- Google (2014). *Women Who Choose Computer Science—What Really Matters* (White Paper). Retrieved from https://docs.google.com/file/d/0B-E2rcvhnIQ_a1Q4VUxWQ2dtTHM/
- Hoyles, C., & Noss, R. (1987). Synthesizing mathematical conceptions and their formalization through the construction of a Logo-based school mathematics curriculum. *International Journal of Mathematical Education in Science and Technology*, 18, 581–595. doi:10.1080/0020739870180411
- Johns, S., & Wolking, M. (2015). *The Core Four of Personalized Learning: The Elements You Need to Succeed* (White Paper). Education Elements. Retrieved from <https://www.edelements.com/core-four-elements-of-personalized-learning>
- Kaminskienė, L., Rutkienė, A., Trepulė, E. (2015). Integration of technology enhanced learning within business organizations: which strategy to choose? *Turkish Online Journal of Educational Technology*, 14 (4,) 78-92.
- Kaminskienė, L., Trepulė, E., Rutkienė, A., Arbutavičius, G. (2014). A responsive paradigm for technology enhanced learning (TEL) integration into business organizations. *European Journal of Open, Distance and E-Learning (EURODL)* 17 (2), 194-206. Budapest: EDEN.
- Kaya, I., Habacı, I., Küçük, S., Adigüzelli, F., & Habacı, M. (2011). Development of instructional strategies scale: Reliability and validity. *World Applied Sciences Journal*, 15(4), 507–516. Retrieved from [http://idosi.org/wasj/wasj15\(4\)11/8.pdf](http://idosi.org/wasj/wasj15(4)11/8.pdf)
- Kynigos, C. (2015). Constructionism: Theory of Learning or Theory of Design? In S.J. Cho (ed.) *Selected Regular Lectures from the 12th International Congress on Mathematical Education* (pp. 417–438). Springer, Cham. doi:10.1007/978-3-319-17187-6_24
- Laurillard, D. (2002). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*. 2nd Edition. Routledge.
- MacKellar, B.K., Sabin, M., & Tucker, A. (2013). Scaling a framework for client-driven open source software projects: a report from three schools. *Journal of Computing Sciences in Colleges* 28 (6): 140–147.
- Margolis, J., Goode, J., Chapman, G., Ryoo, J. (2014). That classroom “magic:” teaching practices for broadening participation in computer science. *Communications of the ACM*, 57 (7): 31-33 (July 2014). doi: 10.1145/2618107
- Mascaró, M., Sacristán, A. I. & Rufino M. (2014). Teaching and learning statistics and experimental analysis for environmental science students, through programming activities in R. In G. Futschek & C. Kynigos (Eds.), *Constructionism and Creativity - Proceedings 3rd Intl. Constructionism Conf. 2014* (pp. 407-416). Vienna, Austria: OCG.

- Meehan, D., & Sabin, M. (2013). QuizPower: a mobile app with App Inventor and XAMPP service integration. In *Proceedings of the 14th annual ACM SIGITE conference on Information technology education* (pp. 103–108). Orlando, Florida: ACM. doi:10.1145/2512276.2512300
- Muller, E., Buteau, C., & Sacristán, A. I. (2015). Through the Looking-Glass: Programming Interactive Environments for Advanced Mathematics. *Mathematics Today*, 51(6), 212–217.
- Olivera, M.A., Sacristán, A.I. & Pretelín-Ricárdez, A. (2013). Mathematical learning derived from virtual collaboration, exploration and discussion of free-fall videos, amongst continuing education students. In E. Faggiano & A. Montone (Eds), *Proceedings of the 11th International Conference on Technology in Mathematics Teaching* (ICTMT11) (pp. 232-237). Bari, Italia: University of Bari.
- Olsen, J., Preston, A. I., Algozzine, B., Algozzine, K., & Cusumano, D. (2018). A Review and Analysis of Selected School Climate Measures. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, 91(2), 47–58. doi:10.1080/00098655.2017.1385999
- Onweh, V. E., & Akpan, U. T. (2014). Instructional strategies and students academic performance in electrical installation in technical colleges in Akwa Ibom State: Instructional skills for structuring appropriate learning experiences for students. *International Journal of Educational Administration and Policy Studies*, 6(5), 80–86. doi:10.5897/IJEAPS2014.0347
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.
- Papert, S. & Harel, I. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex. Retrieved from <http://www.papert.org/articles/SituatingConstructionism.html>
- Psenka, C. E., Kim, K.-Y., Okudan Kremer, O., E, G., Haapala, K. R., & Jackson, K. L. (2017). Translating Constructionist Learning to Engineering Design Education. *Journal of Integrated Design and Process Science*, 21(2), 3–20. doi:10.3233/jid-2017-0004
- Sabin, M. (2011). Free and Open Source Software Development of IT Systems. In *Proceedings of the 2011 Conference on Information Technology Education* (pp. 27–32). New York, NY, USA: ACM. doi:10.1145/2047594.2047601
- Sabin, M. Alrumaih, H. & Impagliazzo, J. (2018). A competency-based approach toward curricular guidelines for information technology education. In *2018 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1214–1221) Santa Cruz de Tenerife, Spain: IEEEExplore. doi:10.1109/EDUCON.2018.8363368
- Sabin, M., Deloge, R., Smith, A., & DuBow, W. (2017). Summer Learning Experience for Girls in Grades 7–9 Boosts Confidence and Interest in Computing Careers. *Journal of Computing Sciences in Colleges*, 32(6), 79–87. Retrieved from <http://dl.acm.org/citation.cfm?id=3069658.3069673>
- Sacristán, A. I. (2017a). Constructionist computer programming for the teaching and learning of mathematical ideas at university level. In Göller, R., Biehler, R., Hochmuth, R., Rück, H.- G. *Didactics of Mathematics in Higher Education as a Scientific Discipline*. khdm-Report 17-05 (pp. 124–131). Kassel, Germany: Universitätsbibliothek Kassel.
- Sacristán, A. I. (2017b). Digital technologies in mathematics classrooms: barriers, lessons and focus on teachers. In E. Galindo & J. Newton (Ed.), *Proceedings of the 39th PME-NA* (pp. 90–99). Indianapolis, IN: Hoosier Association of Mathematics Teacher Educators. <http://www.pmena.org/pmenaproceedings/PMENA%2039%202017%20Proceedings.pdf>
- Sacristán, A. I., & Pretelín-Ricárdez, A. (2017). Gaining modelling and mathematical experience by constructing virtual sensory systems in maze-videogames. *Teaching Mathematics and Its Applications: An International Journal of the IMA*, 36(3), 151–166. doi:10.1093/teamat/hrw019
- Sacristán, A.I. (2018). Constructionist experiences for mathematics across educational levels. In *Proceedings Constructionism 2018*, Vilnius, Lithuania.

Schneckenberg, D. (2010). What is e-Competence? Conceptual Framework and Implications for Faculty Engagement. In U.D. Ehlers & D Schneckenberg (Eds), *Changing Cultures in Higher Education* (pp. 239–256). Springer: Berlin, Heidelberg. doi:10.1007/978-3-642-03582-1_19

Teresevičienė, M., Volungevičienė, A., Trepulė, E., Žydžiūnaitė, V., Rutkienė, A., Tait, A. W., & Kaminskienė, L. (2015). *Technology enhanced learning integration into organizations*. Versus Aureus. <https://eltpykla.vdu.lt/handle/123456789/95>

Viray, J. S. (2016). Parental Involvement as Predictor of Student Academic Performance. *Imperial Journal of Interdisciplinary Research*, 2(6), 1379–1382. Retrieved from <http://www.imperialjournals.com/index.php/IJIR/article/view/1023>

Teachers' Day

The beauty in science and the science in beauty or mathematics, informatics and science teaching as an eye-opener of the beauty of ideas

Evgenia Sendova, jenny.sendova@gmail.com

Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences, Bulgaria

Abstract

In this talk I will discuss my involvement in different roles in some educational practices in Bulgarian and international context of opening students' eyes for the beauty of science in general with an emphasis on mathematics and informatics.

I had the chance of being involved in education relatively late in my professional carrier in the frames of the Research Group on Education experiment (1979-1991) where the leading principles embraced *learning by doing and exploring, integration of school subjects, making the computer a natural component of the curriculum and enhancing the creativity of both teachers and students*. The research team developing the educational resources in a newly designed subject *Language and Informatics* (integrating three natural languages – Bulgarian, Russian and English, with two artificial ones – mathematics and Logo) was given the freedom to implement the Logo educational philosophy (*constructionism*) in an attempt to make the children love the school in general (and math and science in particular). The experiment lasted for 12 years and what our team tried to transfer in the next educational projects in national and international setting was that mathematics and science could be viewed and taught as a *game of ideas, as painting with ideas*. Thus, we got involved with the design and implementation of educational resources – textbooks and dynamic scenarios in mathematics, IT, and informatics (5th-12th grade, Figure 1) as well as in the development of *computer microworlds*, tuned to specific domains (e.g. *Geomland*, a language-based mathematics laboratory for explorations in Euclidean geometry, Figure 2). Such microworlds were launched within various Educational national and European projects and used in numerous PD courses for teachers as platforms for testing and editing various ideas in math, science and arts allowing the participants to discover the **beauty of mathematics and science in terms of ideas** – an experience they would convey further to their students.

It is the very ideas behind some projects developed by the participants in PD courses at Sofia University on implementing IT in math classes on the theme *The beauty in mathematics and the mathematics in the beautiful* that I will present (Figure 3). Furthermore, I will discuss how educational resources developed in the Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences (IMI-BAS) in support of the inquiry based mathematics and science education (IBMSE) are being implemented in in-service teacher education (Figure 4).

The experience gained confirms our belief that the learners construct new knowledge with particular effectiveness when they are engaged in modeling and constructing something that is meaningful to themselves and to others around them (Figures 5 and 6).



Figure 1. Textbooks of the Research Group on Education (4th, 5th and 6th grade)

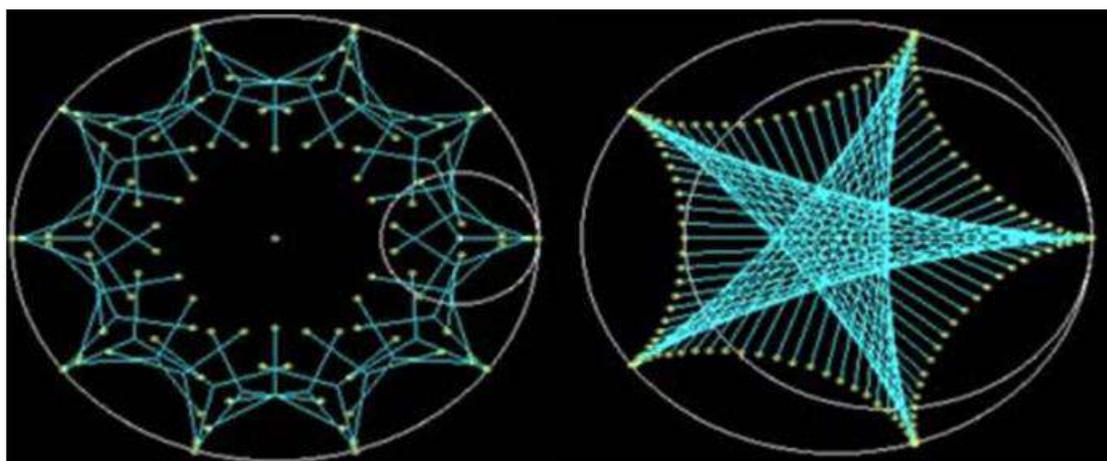


Figure 2. Different hypocycloids generated in Geomland (<http://sunsite.univie.ac.at/elica/PGS/INDEX.HTM>)

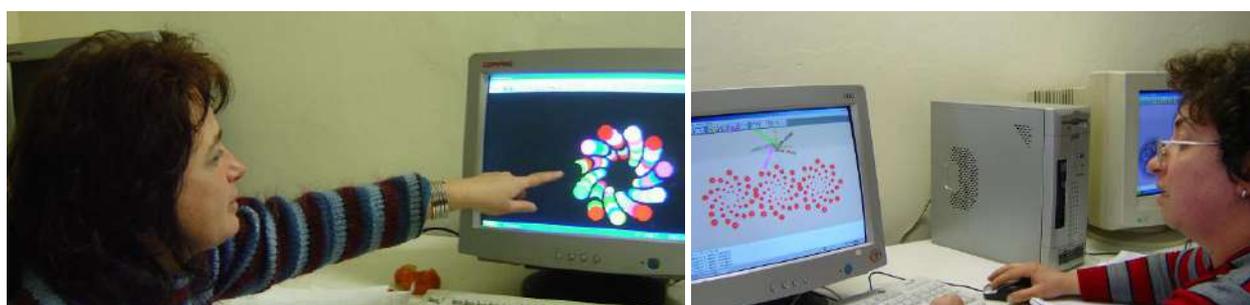


Figure 3. In search for the golden angle in nature – a PD course for IT teachers at Sofia University



Figure 4. Educational resources developed by the IMI-BAS team within the projects VirMathLab (www.cabinet.bg) – left; Mascil (www.math.bas.bg/omi/mascil) – middle, and Scientix (<http://www.scientix.eu/>) - right



Figure 5. Celebrating the STEM Discovery week (April, 2018, <http://www.scientix.eu/stem-discovery-week>) at the IMI-BAS - the Scientix Bulgarian contact point, with the event Mathematics in the world around me



Figure 6. A visit to a Math and Science Fair in Varna, Bulgaria, within an international workshop of the STEM PD Net project (<http://stem-pd-net.eu/en/>)

Keywords/phrases

Beauty; mathematics, informatics, IT and science education; creativity; Logo, Inquiry based learning

Ateities švietimas: naujos galimybės Lietuvai?

Rimantas Želvys rimantas.zelvys@fsf.vu.lt
Vilnius University, Lithuania

Viena iš trijų Lietuvos šimtmečio proga paskelbtų idėjų Lietuvai – „Mokytojas – prestižinė profesija iki 2025 metų“. Ar ši misija įmanoma? Teigiama užsienio patirtis rodo, kad iš principo tokios idėjos įgyvendinimas įmanomas. Tačiau kyla kitas ne mažiau svarbus klausimas: ar mes ją įgyvendinsime? Deja, tokia galimybė labai abejotina, jeigu įgyvendinimo sieksime tokiais būdais, kaip siekėme iki šiol. Šiuo metu matome pastangas įgyvendinti pokyčius švietime, tačiau vyrauja kiti prioritetai: klasių krepšelio įdiegimas, etatinis mokytojų darbo apmokėjimas, universitetų jungimas ir pedagogų rengimo centrų steigimas. Nė vienas iš šių prioritetų tiesiogiai neprisideda prie mokytojo prestižo kėlimo ar šviesesnės švietimo ateities. Mūsų manymu, tam, kad idėja Lietuvai būtų įgyvendinta, švietimo prioritetai turėtų būti kiti:

1. Aiškių švietimo tikslų ir ateities vizijos suformulavimas. „Pradėkite kampaniją, kuri padėtų iš naujo sukurti švietimo viziją, nes, jei nežinosite, kokios mokyklos norite, neverta nieko net pradėti“, sako vienas iš labiausiai pasaulyje žinomų švietimo ekspertų P. Salbergas. Ar mokyklai lemta būti pramonininkų ir verslininkų poreikius tenkinančiu ekonomikos priedeliu, ar mūsų siekis – pagal tarptautinio PISA tyrimo rezultatus pavyti ir pralenkti Estiją, ar ir toliau vadovaujames Lietuvos švietimo koncepcijos nuostata, kad pagrindinė švietimo misija – ugdyti aukštos kultūros, pilietiškai orientuotą ir socialiai atsakingą jaunąją kartą? Kiekvienas iš šių siekių reikalauja skirtingų įgyvendinimo priemonių, o mokytojo prestižas bus aukštas tuomet, kai suvoksime mokytoją visų pirma kaip kultūros nešėją, o ne kaip paslaugos teikėją.
2. Aukštesnis ir geresnis išsilavinimas. Daugelyje Europos Sąjungos šalių manoma, kad mokytojo darbui būtinas magistro išsilavinimas. Kodėl mūsų šalyje galvojama, kad, norint tapti vaikų gydytoju, reikia mokytis daugelį metų, o vaikų mokytojui tai nėra būtina? Ar pažinti jauno žmogaus sielą yra lengvesnis uždavinys, negu pažinti jauno žmogaus kūną? Profesijos prestižas tiesiogiai susijęs su išsilavinimu – vargu ar žemesnio išsilavinimo reikalaujanti profesija gali būti prestižinė.
3. Švietimas turi būti grįstas mokslu. Visuomenė, nepaisant visų pastarųjų dešimtmečių pokyčių, vis dar gerbia mokslininkus ir pasitiki mokslo žiniomis. Deja, politikai, priimdami sprendimus, vis rečiau remiasi moksliniais tyrimais, o pati edukologija žeminama ir abejojama, ar tai iš viso mokslas. Jei netinka pavadinimas – keiskime, jei netenkina mūsų mokslininkų lygis – kvieskime iš užsienio pripažintus autoritetus, bet negalima mokslo aplamai ignoruoti. Mokytojas turės didesnį prestižą tuomet, kai visuomenė žinos, jog savo darbe jis vadovaujasi naujausiais mokslo pasiekimais.
4. Turi būti aktyviai kuriamas teigiamas mokytojo įvaizdis. Savaiame jis tikrai nesusiklostys toks, kokio norime. Jei pageidaujame mus tenkinančio įvaizdžio, turime aktyviai ir kryptingai dirbti: bendrauti su žiniasklaida, plėtoti ryšius su visuomene, tinkamai propaguoti mokytojo profesiją. Pavyzdžių toli ieškoti nereikia – pakanka prisiminti, kiek pastangų pastaraisiais metais buvo dedama, gerinant profesinio rengimo įvaizdį. Kryptinga veikla šioje srityje jau duoda apčiuopiamų rezultatų. Beje, ir patys mokytojai turi saugoti savo įvaizdį ir jo negadinti neapgalvotais lozungais ir akcijomis.
5. Prestižas tiesiogiai susijęs su atlyginimu už atliekamą darbą. Egzistuoja neprestižinės, bet gerai apmokamos profesijos: yra nemalonių darbų, kurių niekas nenori imtis, todėl bandoma surasti norinčių juos atlikti už gerus pinigus. Tačiau nėra prestižinių, bet prastai apmokamų profesijų. Juk žmonės mąsto paprastai – jei atlygis menkas, tai ir darbas, matyt, nėra nei būtinas, nei reikšmingas.

Taigi, šiandien Lietuvoje inicijuojami pokyčiai sudaro potencialias galimybes sėkmingai kurti ateities švietimą. Tačiau tam, kad šios galimybės būtų realizuotos, būtinas kitoks mąstymas ir kiti prioritetai.

WS1: Dynamic Teaching Ideas for teaching Music Theory

Judith Bell, jbell@chisnallwood.school.nz

Chisnallwood Intermediate School, Christchurch, New Zealand

Abstract

In this workshop you will try out a range of effective, outside-the-square, tried-and-true theory activities based on Judith's creative work with the very popular Chisnallwood School Theory Club. You will learn about music theory games are suitable for a range of ages and easily adaptable for different levels and music abilities, but ideally from ages 7 – 12. Some activities use digital technology, while others are completely unplugged. They are suitable for use in groups – even of mixed levels – and you'll be amazed at the learning that comes out as well as the amount of fun.

WS2: Computer Science Unplugged for Teachers

Tim Bell, tim.bell@canterbury.ac.nz

University of Canterbury, New Zealand

Abstract

This workshop will demonstrate ideas from the "Unplugged" approach to teaching Computational Thinking topics. This approach provides opportunities for students to encounter the great ideas in computer science away from computers, and is a useful complement to classes that focus on "coding". We will look at how you can engage your own students with the material, and also "plugged in" activities, which connect the activities to programming. You will learn about free resources that you can use in your classroom immediately, and also how you can integrate these activities with other school subjects. The workshop will be suitable for both primary and secondary educators.

WS3: Developing Algebraic Habits of Mind in Students

Paul Goldenberg, pgoldenberg@edc.org

Education Development Center, Waltham, MA, USA

Cynthia J. Carter, CCarter@rashi.org

The Rashi School, USA

Abstract

Participants will see one puzzle-centric approach (elaborating on the work of mathematician W.W. Sawyer) aimed at developing the language and logic of algebra in grades 5 and up. Most of the focus of the workshop will be on the beginnings of that algebraic development, but we will also spend some time looking at activities - from paper-ripping to factorials of negative numbers - that help build students' comfort with the mathematical idea of extension, extending ideas that have "natural" meanings in the natural numbers and extending them to "unnatural" places, like negative numbers and fractions. Participants will be encouraged to pose problems of their own, extending problems presented in the workshop, and will receive a packet of materials they can adapt and use with their classes.

WS4: Puzzles & Programming to Develop Mathematical Habits of Mind in 6–10 year Olds

Paul Goldenberg, pgoldenberg@edc.org
Education Development Center, Waltham, MA, USA

Cynthia J. Carter, CCarter@rashi.org
The Rashi School, USA

Abstract

This workshop will focus on tested work using a puzzle-centric primary mathematics curriculum with an algebraic focus (inspired by mathematician W.W. Sawyer), and new work we are doing now to infuse that curriculum with programming (inspired by work by Jenny Sendova and others in Bulgaria and by the ScratchMaths work done by Richard Noss and Celia Hoyles of the UK and Ivan Kalaš of Slovakia). Puzzles help develop children's logic; programming provides an extra language to help them express that logic. Expressing and experimenting with one's mathematical ideas helps develop them further. Participants will solve and create puzzles that are suitable for young students and will see how computer programming by young students can support their mathematical learning. All materials that will be used or described are available free.

WS5: Powerful Ideas in Lower Primary Programming: High Time to Recognize Them

Ivan Kalaš, kalas@fmph.uniba.sk
Comenius University, Bratislava, Slovakia

Abstract

We think that using any of the dozens of existing programming environments in regular primary school setting, with general primary teacher does not work well, mostly because of the following reasons: (a) these environments are not designed for collaborative constructivist teaching where the teacher has to support everybody in the class, and (b) these environments often neglect basic, simple, but key important powerful ideas that pupils should discover and adopt before computational constructs that are considered in general as introductory (sequence, selection, repetition, variables...)

In the workshop we will use our latest development of Emil, a new programming environment and its systematic pedagogy constructed and trialled in a group of design primary schools, with pupils aged 8. Together with the participants we will experience, identify and discuss several powerful ideas that pupils should experience before all other powerful ideas that we have exploited in our ScratchMaths curriculum.

WS6: Snap! - Beauty & Joy of Computing (visually)

Witek Kranas, witek@oeiizk.waw.pl
OEIiZK, Poland

Abstract

It is worth noting that Snap! is not a toy for children. It was created to help you learn programming understood as creating and implementing algorithms. And although the floor, like in Scratch, is quite low, the ceiling is much higher. In our (Poland) realities, it can be reasonably used at the level of grades 7-8, although one can imagine an IT course for a high school based on Snap!, as the main programming environment.

Prepared for a Teachers Day workshop during Constructionism 2018 conference in Vilnius.

Based on my articles published in a magazine "W cyfrowej szkole" and conference "Informatyka w edukacji" materials, 2018.

About Snap! (snap.berkeley.edu)

Snap! is a Scratch clone created by Brian Harvey and Jens Mönig. In the first version, the environment was named BYOB (Build Your Own Blocks). This name contained the main reason for the creation of the environment - the possibility of creating own blocks by the user because in the version of Scratch 1.4 it was not possible. The creators decided to change name due to the other popular use of this abbreviation – *bring your own bottle*, added on party invitations. They write about it in the textbook: "*The name of the program has been changed, because some teachers have no sense of humour.*"

Snap! is programmed in HTML5 and can be run in almost any system, on any device. Creating new blocks is much more complex than in Scratch. You can choose a palette for a new block and specify whether it will be a command (fragment of a puzzle with a tab), a function (a rounded block) or a predicate (a hexagonal block). It is also possible to insert prefix parameters. In addition to a set of blocks almost identical to Scratch, Snap! contains several extensions - additional block libraries. There is also the possibility of switching on step work to trace scripts. The palette of blocks supporting lists has been greatly expanded, which means that it is possible to use full capabilities of this data structure (list as a variable, lists of lists ...).

Seemingly Snap! is a Scratch clone (version 1.4). The set of blocks is almost identical to Scratch, their appearance is very similar. So, is it worth to deal with the Snap! I will try to convince you that it is worth it.

- The first reason - running on tablets.

Snap! has been programmed in HTML5, so it will work on the vast majority of mobile devices basically regardless of the system. Installation of the offline version is also very simple, it does not require any additions. This reason may disappear at the end of 2018, if Scratch's team will launch a new version 3.0 using Google Blockly blocks.

- Second reason – easy localization (if you have local version of Scrtach).
- Third reason - a new block can be a function (return value).
- Fourth reason - additional blocks of libraries.

In addition to the standard set of blocks - almost identical to Scratch, Snap! offers several extensions, additional block libraries. We add them to the palettes by selecting from the menu File / Library.

- Fifth reason - lists like in Logo.
- Sixth reason - visualization of script execution (step work).

- Seventh reason - easy preparation of materials.
- Last (but first for the authors) – first class procedures

From Brian Harvey letter: *...most important to me: first class procedures. The FOR block, ... is written in Snap!, not a primitive, and that's possible only because we can capture the script in the C-slot as an input value, and the FOR block can then RUN it repeatedly. And of course MAP, KEEP, and COMBINE require this feature.*

About BJC (bjc.berkeley.edu)

The American computer science course "The Beauty and Joy of Computing", called here in brief BJC is intended for students aged 16-18, what means the level of high school (lyceum). It belongs to Advanced Placement courses, it finishes with an exam and gives you the opportunity to get points for university admissions.



The course contains detailed materials in a form that allows the student to work independently.

The course consists of six compulsory parts:

Unit 1: Introduction to Programming

Unit 2: Abstraction

Unit 3: Data Processing and Lists

Unit 4: How the Internet Works

- AP Explore Task

Unit 5: Algorithms and Simulations

- AP Create Task

Unit 6: How Computers Work

and two additional parts to be processed after the exam:

Unit 7: Fractals and Recursion

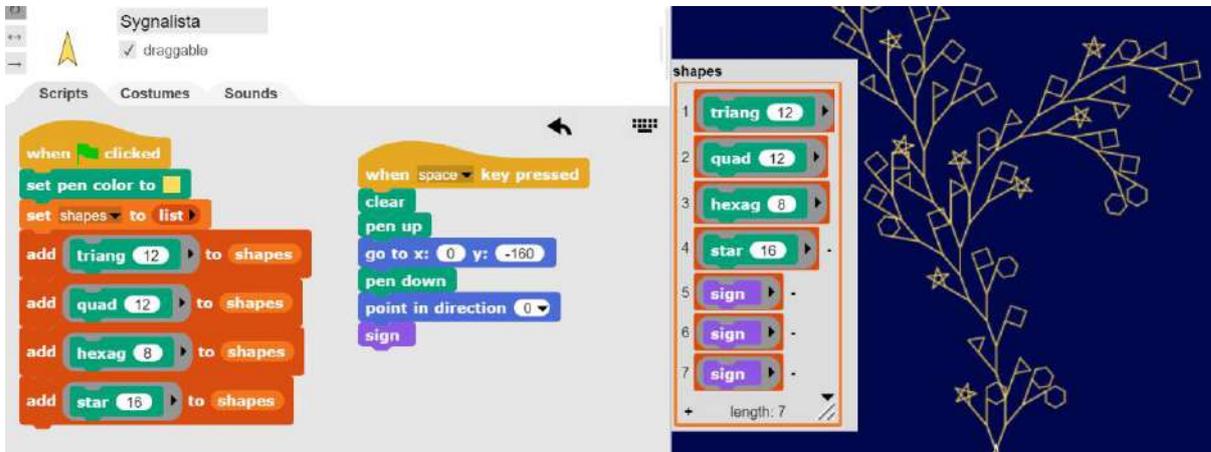
Unit 8: Recursive Functions

The authors write: *"In this course, you will create programs using the Snap programming language, you will learn some of the most powerful ideas of computer science, you will be*

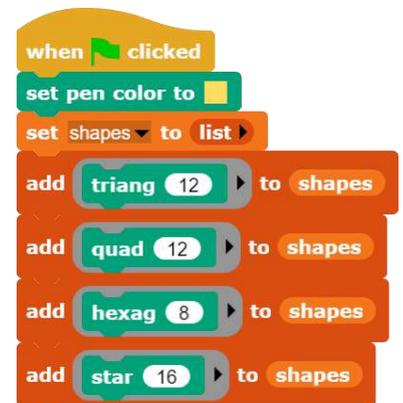
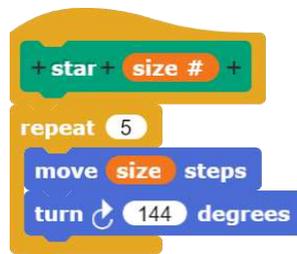
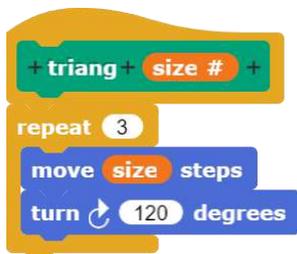
creative, and you will discuss the social implications of computing, thinking deeply about how you can be personally active in promoting the benefits and reducing the possible harms.”

Signalist (vee)

<https://snap.berkeley.edu/snapsource/snap.html#present:Username=witek&ProjectName=sygnalistaAng>

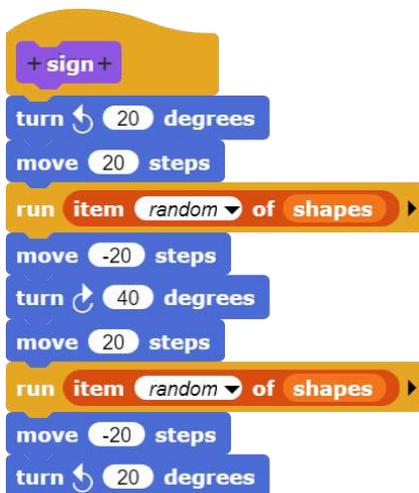


This example was prepared by the authors of Snap! to show environments strength. We start with a few new blocks drawing simple figures (*triang*, *star*, ...).



Now we create variable *shapes* and give it the value of an empty list.

Then, on this list, we place the created blocks so that they can be executed, i.e. in the grey envelope, (located in the *Operators* palette). You can do this by using the **add ... to ...** block, as in the green flag script in the picture.



It's time to create a main block (*sign*) that will draw a signal composed of two flags with random shapes. Running this block will draw two flags. But how can you get a picture like the top picture, where there are many more flags? Just put a few *sign* blocks on the list in grey envelopes. Add and delete them, for example, using the scripts shown in the last pictures.

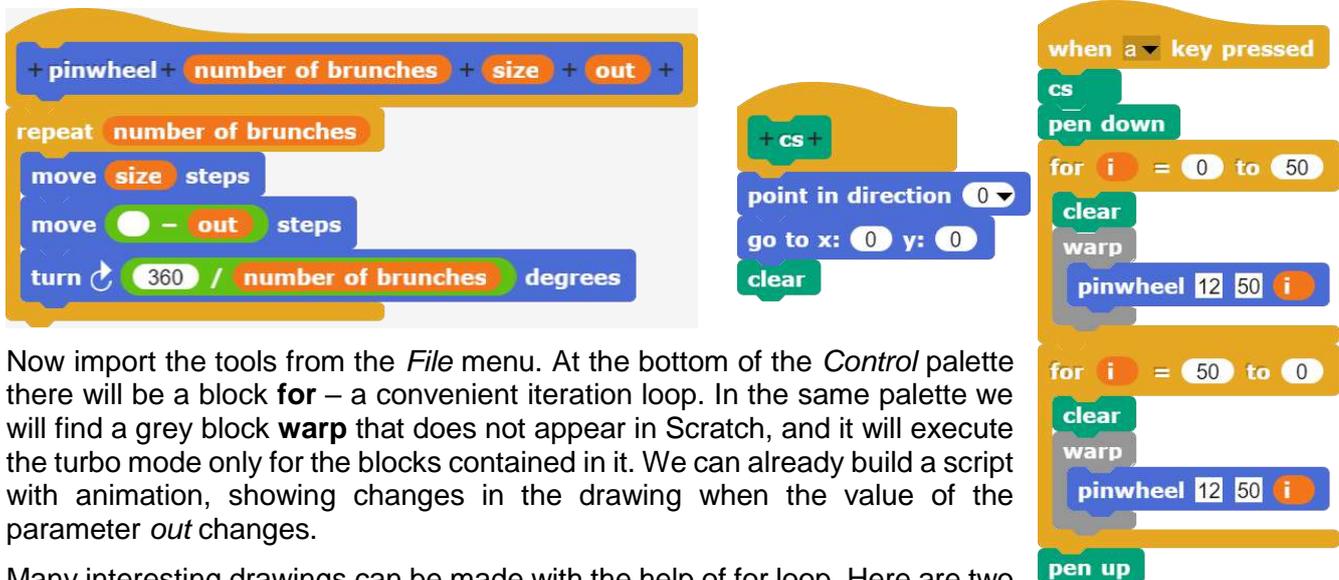


Well, now we have recursion without a base case and thus without a stop condition!

Polygon, pinwheel and for loop

<https://snap.berkeley.edu/snapsource/snap.html#present:Username=witek&ProjectName=BJC3dlaAnq>

The second example refers to programming in Logo. How to draw a polygon? We can create a new block in the *Motion* palette and in the block editor build a command script (i.e. the new block will have a blue colour and a tab layout). Let's add to the standard polygon (with a *number of brunches* and *size*) the third parameter *out* and the block, which will cause the section of its length to stick out from one side. Let's call this creation a **pinwheel**. The polygon is obtained in the extreme case, when the parameter *out* will be equal to zero. And what figure will we get when it's equal to *size*?



Now import the tools from the *File* menu. At the bottom of the *Control* palette there will be a block **for** – a convenient iteration loop. In the same palette we will find a grey block **warp** that does not appear in Scratch, and it will execute the turbo mode only for the blocks contained in it. We can already build a script with animation, showing changes in the drawing when the value of the parameter *out* changes.

Many interesting drawings can be made with the help of for loop. Here are two examples to try.



Which images will be done by scripts?

One more example of using **for** loop – countdown from 0 to 99. In the script, we use the function **if**, which allows you to not write zeros at the beginning of the countdown.

```

when key pressed
  cs
  for tens = 0 to 9
    for ones = 0 to 9
      say join if tens = 0 then else tens ones for 0.3 secs
  
```

Symmetric encryption

<https://snap.berkeley.edu/snapsource/snap.html#present:Username=witek&ProjectName=BJC43CCAng>

In part 4 of the BJC course "How the Internet Works" in the laboratory "Cybersecurity" there are topics related to cryptography, among them *Caesar Cipher Project*. Encryption is done by shifting letters in the (Latin) alphabet by a given number (1-26). This can be illustrated with the help of two independently rotating wheels containing all letters of the alphabet around the circumference. You can see it in the picture, which is a screenshot of a fragment of students BJC materials.

Symmetric Cryptography

You might have used a **substitution cipher** to encode your message, substituting each letter of the alphabet with some other letter. You could substitute letters in any order, like this:

```

ABCDEF GHIJKLMNOPQRSTUVWXYZ
PQVFUBZOTHWYELIXRNAMGDSCKJ
  
```

That's called a **general substitution cipher**.

Or you might just *shift* the letters in order. For example, this is a shift of 3:

```

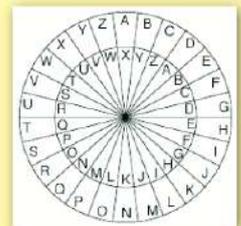
ABCDEF GHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
  
```

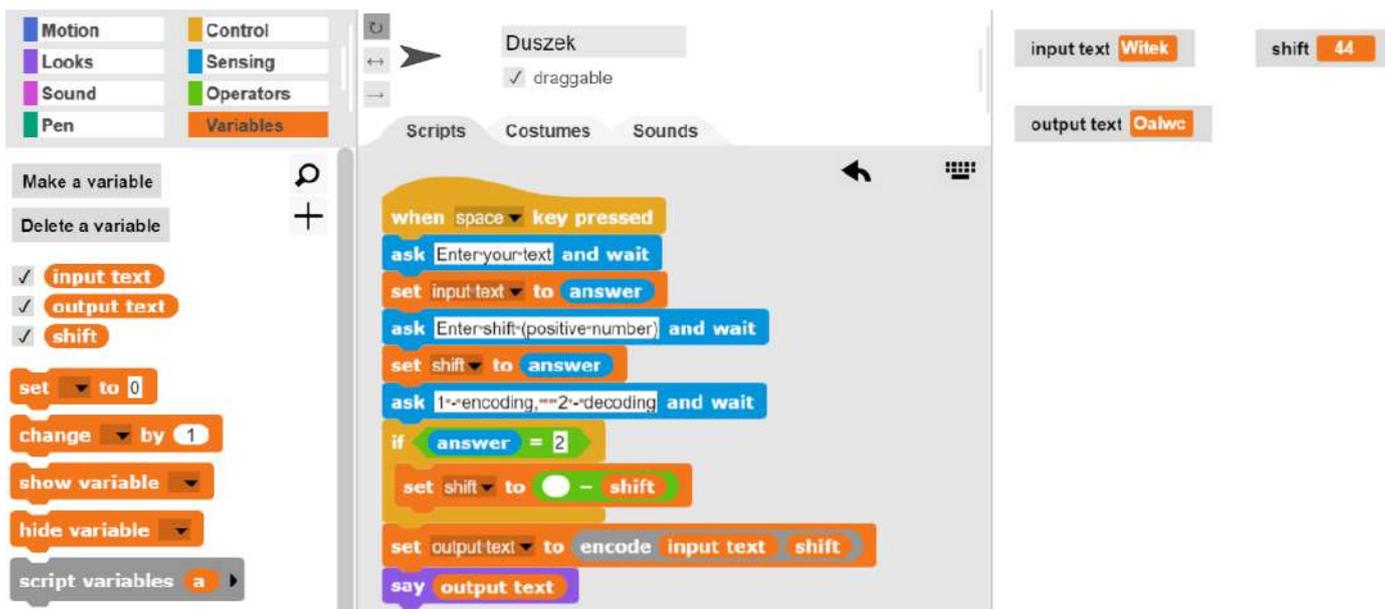
That's called a **Caesar Cipher** (named for Julius Caesar) or a **shift cipher**.

Substitution ciphers are examples of symmetric encryption because they use the same key for both encryption and decryption.

We start project in Snap! creating three global variables named: *input text*, *shift* and *output text*. In the script run by pressing the spacebar, we ask for the text to be encrypted, the letter's shift and ask to choose whether to encrypt (1) or decrypt (2).

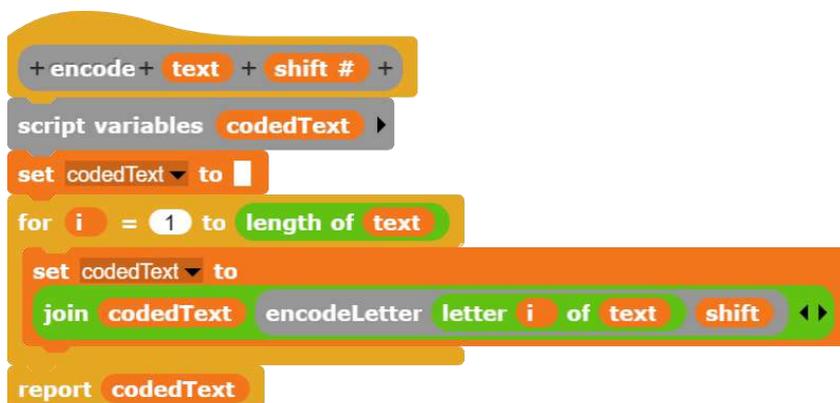
Imagine a pair of wheels, pinned together at their center, but free to rotate independently like this:





The last part of this script starts encryption. It is executed by the *encode* block, which must be defined. The new block will belong to the reporter category (i.e. function, rounded shape) and will have two parameters: the *text* to be coded and the letter *shift* in the alphabet.

Building this block requires a **for** loop, located in the tools library. So you have to import the tools using the *File* menu. In the block we create a local variable (*codedText*), which will store the coded text, and then for all letters of the input text we call the *encodeLetter* block, joining them into one encoded text.

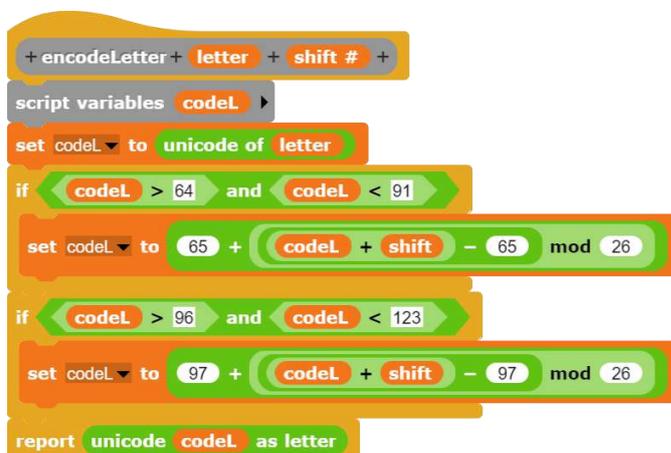


You still need to create this last block. A simple solution is to replace the letter with its ASCII code, add shift to it and change it back to the letter. Let's see how this block codes the letter Z with shift 1.



You can see that we have not closed the wheel. We have to modify this block by moving codes modulo 26 (the number of letters in the Latin alphabet). It's done by two blocks if identifying uppercase and lowercase letters. Now only the letters are coded but with cyclical behaviour.

And how to get decoding? Look closely at the green flag script.



Sierpiński gasket

<https://snap.berkeley.edu/snapsource/snap.html#present:Username=witek&ProjectName=BJC72SGA>

```

+triangle+ size # +
repeat 3
  move size steps
  turn 120 degrees

+Striangle+ size # +
if size > 5
  repeat 3
    move size steps
    Striangle size / 2
    turn 120 degrees
  
```

The last example is a fractal - a gasket (triangle) by Sierpiński. In the BJC course, recursion is introduced very slowly in additional chapters (7 and 8). Let's look at the introductory script for the Sierpiński triangle recursive construction. We have here, three times repeated, script drawing a triangle. It is worth analysing this script first, imagine how the next smaller and smaller triangles will be drawn, and then start. This is not a typical Sierpiński triangle, but let's try to draw it using recursion. You have to decide when to finish the duplication of triangles. The simplest way is to stop drawing when the side length is small enough. This implements a block *if* in *Triangle*. We draw only triangles with a side longer than 5, and in the middle we have a recursive call. Now let's consider the position of this call in the script. It can be moved to the beginning or the end. Try.

```

pen down
repeat 3
  set pen color to red
  move 100 steps
  repeat 3
    set pen color to blue
    move 50 steps
    repeat 3
      set pen color to green
      move 25 steps
      turn 120 degrees
    turn 120 degrees
  turn 120 degrees
pen up
  
```

But you can also have fun inserting it inside the rotation, if you swap the block *turn* (right) 120 degrees for two, e.g.

turn 30 degrees and turn 90 degrees, you can insert a recursive call between these two blocks.

Let's look at the block *StriangleA*, whose last parameter (*turn1*) is the angle of the first rotation (the second is easy to calculate) and the result of its operation.

```

+StriangleA+ n # + size # + turn1 # +
if n < 1
  stop this block
repeat 3
  move size steps
  turn turn1 degrees
  StriangleA n - 1 size / 2 turn1
  turn 120 - turn1 degrees

when a key pressed
hide
for i = 1 to 360
  go to x: -50 y: -60
  point in direction 0
  clear
  pen down
  warp
  StriangleA level 120 i
  pen up
  
```

Using blocks **for** and **warp** like in the second project, we can get a nice animation showing the change in the drawing when the angle of the first rotation changes.

WS7: ViLLE – E. Learning Path for Mathematics and Programming

Mikko-Jussi Laakso, *milaak@utu.fi*, **Petra Enges-Pyykönen**, *phenge@utu.fi*
University of Turku, Finland

Abstract

ViLLE is an exercise-based education environment that enables easy learning and teaching of mathematics, programming and other topics. The development is research-based, and the features and the methodology utilized have been thoroughly studied with various setups in the Centre for Learning Analytics at University of Turku. For students, ViLLE offers more than 15 000 carefully designed, motivating and activating exercises for learning mathematics and programming. All exercises are automatically assessed and provide immediate feedback. For teachers, ViLLE provides comprehensive learning analytics that visualize everything you need to know about your students' learning process – including automatic detection of misconceptions and real-time analysis of students' progress. ViLLE is used to transfer one lesson a week into an electronic learning experience. There are existing exercises and materials for all nine grades of primary and middle school. Moreover, there are programming exercises integrated into all levels to give students a head start in learning computational thinking and basics of computer science.

Using ViLLE provides evidence-based, scientifically proven results for all grades. In the studies conducted in Finland – the country that excels the Pisa assessments each year – it was confirmed that the students using ViLLE improve their learning significantly more than the control group learning mathematics with traditional pen and paper method. With matching skill levels before the experience, groups using ViLLE achieved at least 20 percent higher scores in the exams conducted at end of the school year. Students using ViLLE also make 70 % less errors than students in a control group. Moreover, the students find ViLLE as highly motivating and fun tool to use.

WS8: Constructionism in Action: Do we Need to Start from Scratch?

Evgenia Sendova, *jenny.sendova@gmail.com*
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria

Nikolina Nikolova, *nnikolova@fmi.uni-sofia.bg*
Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", Bulgaria

Abstract

Children in the digital era are surrounded by information and communication technologies. The development of digital competences and more specifically –the ability to express their creativity through computational thinking, is evaluated by the society as vital for the contemporary society. This makes it natural to introduce programming courses for students of younger age, new curriculum and even a new school subject – computer modeling.

Should the programming be taught per se though? Does the introduction of new syllabus put a threshold and ceiling on the performance of the teachers? What about programming languages with no threshold and no ceiling (the ceiling being only the user's imagination)? Bulgaria has a long-term experience in teaching programming, and even better – in learning through programming. The Logo philosophy promotes the programming as a means for learning and creative self-expression. It is in harmony with the family of contemporary programming languages, successors to Logo and developed specially for children.

There will be no threshold for the participants. However, this does not mean that we would start from scratch. Rather, we will start with the traditions of the Logo philosophy and Logo culture in international setting, we'll present the potential of their development through Scratch and of course, we'll work, create and have fun together! Most importantly, we'll rely on high enough ceiling!

WS9: Teaching Coding and Physical Computing

Gary S. Stager, gary@stager.org
Constructing Modern Knowledge, USA

Abstract

Learn how a project-approach to computer programming, robotics, and physical computing can serve a diverse student population while developing your own skills. This workshop will explore powerful ideas from computer science and engineering that may be employed in the solving of problems across the curriculum. A review of software and hardware options will be explored in addition to two focused programming and robotics activities. Participants will also have experience with Hummingbird Robotics, the BBC Micro:bit and other low-cost "microcontroller development boards" offering great potential for learning through making, tinkering, and engineering in the classroom.

WS10: The Essence of Programming at School – Logo in a Spiral Curriculum

Jacqueline Staub, Jacqueline.staub@inf.ethz.ch
ETH Zürich, Pädagogische Hochschule Graubünden, Chur, Switzerland

Abstract

School is responsible for priming and preparing pupils such that they develop a deep understanding of technology. Computer science education serves a vital role in fostering algorithmic thinking and problem solving skills, as exemplified by programming. This form of learning is constructive, enriches creativity and teaches precision. We have been introducing primary school pupils and their teachers to programming in Logo for more than a decade and thousands of children across Switzerland have learned to program using our curriculum and purpose-built programming environment. In this workshop, we give insights into how our curriculum guides pupils to progress individually and how we make pupils building up competence by recovering from their programming errors autonomously. This workshop caters towards educators and people interested in how to introduce computer science to novices. Participants gain practical insights into our curriculum and discuss its didactic structure.

WS11: How to Create and Sustain a Progressive Pedagogy in a Traditional Setting (Roundtable Discussion)

Carol Sperry Suziedelis, *Carol.Sperry@millersville.edu*
Millersville University, USA

Abstract

Format of the workshop: We will encourage a vibrant discussion of ideas, efforts, questions, and fears about creating dynamic and engaging projects as well as the nurturing of an atmosphere of free expression and exploration in the classroom. Without the support of each other, it may be difficult to relax and let go of the “controls” many think are necessary for a workable classroom. However, students and teachers must have the latitude to pursue the construction of their own knowledge, the freedom to make mistakes and “recalibrate,” and the time to discuss the powerful effects of the process. We will consider ways to do it all within the constraints of the average classroom setting.

Along the way, we will revisit some of Seymour Papert’s profound insights into the remarkable endeavour of teaching and learning.

I believe the prediction of “expected outputs” is usually precarious for a constructionist teacher. To “expect” certain outputs is, to my mind, a way of limiting them. So, I would say, I hope that we have a lively discussion that inspires ideas, creates new relationships, and bolsters the courage needed to follow a joyful and innovative path.

WS12: Joyful Learning of Geometry in Cultural Context. Analysis and Construction of Geometric Ornaments

Igor Verner, *ttrigor@technion.ac.il*
Technion – Israel Institute of Technology, Israel

Khayriah Massarwe, *massarwe@technion.ac.il*
Technion – Israel Institute of Technology, The Arab Academic College for Education, Israel

Daoud Bshouty, *daoud@technion.ac.il*
Technion – Israel Institute of Technology, Israel

Abstract

Mathematics education seeks to accommodate pedagogical approaches that enhance learning mathematics and make it relevant to today's students who prefer hands-on visual and joyful activities, and are socially inclined. The challenge of teaching is to expose students to the interconnection between real world practices and culturally rooted mathematical ideas. The subject Geometry is unique in its combination of intuitively rooted figural concepts and abstract logical statements. The geometry teacher should facilitate the learner to acquire the language of geometric terms and the ability to use this language correctly when communicating geometric ideas. We have been developed two courses that introduce prospective and in-service teachers to teaching and learning geometry in cultural context using the ethnomathematical approach. The teachers analyzed and constructed, by compass and straightedge, geometric ornaments from different cultures, posed and solved geometric problems related to the ornaments. In this workshop, we aim to engage the participants in constructing and analyzing of culturally meaningful geometric ornaments. The participants will solve geometric problems related to the ornaments and develop new problem by themselves. Then we will discuss the arguments for introducing activities with geometric ornaments in school geometry learning.

Author Index

A

Abee, Kazuhiro · 745
Adiguzel, Tufan · 777
Agatolio, Francesca · 498
Akçay, Nevin · 777
Angel-Fernandez, Julian · 150, 506, 686, 690
Angulo, Carol · 514
Anton, Gabriella · 380, 554
Asiain, Alfredo · 498
Aslan, Ümit · 125
Atieno, Loice Victorine · 838
Avci, Hulya · 777

B

Baafi, Richard Akrofi Kwabena · 925
Barendsen, Erik · 570
Baumann, Wilfried · 838
Bell, Judith · 520, 945
Bell, Tim · 21, 520, 945
Berger, Neeltje · 767
Beynon, Meurig · 437
Blikstein, Paulo · 29, 203, 472, 481, 754
Boran, Ian · 449
Brady, Corey · 263, 761, 769
Bray, Aibhín · 449
Broll, Brian · 769
Bshouty, Daoud · 657, 956
Buteau, Chantal · 528
Byrne, Jake Rowan · 138

C

Cabibihan, John-John · 150
Campos, Flavio · 536
Cañas, Alberto J. · 514
Carter, J. Cynthia · 945, 946
Castro, Ana Gabriela · 514
Catlin, Dave · 150
Černochová, Miroslava · 543
Chan, Monica · 781
Chukhnov, Anton · 160
Chytas, Christos · 547, 855
Clayson, James · 30
Csizmadia, Andrew Paul · 150, 794
Čuma, Radek · 543

D

Dabholkar, Sugat · 554
Dagienė, Valentina · 4, 169, 180, 305, 344, 731, 838
Davey, Caitlin · 561
Diethelm, Ira · 547
Dolgopolovas, Vladimiras · 180
Du, Xiaoxue · 193
Duca, Annalise · 686, 690, 773

E

Enges-Pyykönen Petra · 954

F

Fields, Deborah A. · 203, 214, 601, 754
Foss, Jonathan · 437
Futschek, Gerald · 38, 678, 794

G

Girvan, Carina · 506, 649, 686, 690
Giuliano, Angele · 686, 773
Göcking, Fenja · 672
Goldenberg, Paul · 39, 945, 946
Goodman, Lizbeth · 759
Greka, Kristina · 690
Grgurina, Natasa · 570
Griffin, Jean M. · 225
Grizioti, Marianthi · 357, 369, 649, 686
Gueorguiev, Ivaylo · 649, 686, 690
Gungor, Ali · 777
Guo, Yu · 238

H

Harada, Yasunori · 741
Harvey, Brain · 53
Heldens, Peter · 767
Hickmott, Daniel · 251, 577
Hjorth, Arthur · 68, 263, 274
Holbert, Nathan · 460, 561, 754
Holmquist, Stephanie · 150
Horn, Michael · 392
Horváth, Győző · 736
Howell, Stephen · 759, 767, 786

Hoyles, Celia · 69, 754

I

Igwe, Kay Chioma · 193

Ito, Kazunari · 585, 593, 745, 884

Ivanović, Mirjana · 608

J

Jasutė, Eglė · 180

Jatzlau, Sven · 285

Jevsikova, Tatjana · 180

Jung, Ungyeol · 295

Jūratė Baranova · 868

Juškevičienė, Anita · 305, 884

K

Kafai, Yasmin B. · 214, 601, 754

Kahn, Ken · 315, 765, 788

Kalaš, Ivan · 71, 946

Kaminskienė, Lina · 925

Kandlhofer, Martin · 150

Kishi, Nobuko · 325

Klašnja-Milićević, Aleksandra · 608

Klein, Markus · 791

Klimeková, Eva · 334

Knox, Tony · 449

Knuth, Alexander · 672

Koppensteiner, Gottfried · 791

Koza, Clemens · 791

Kranas, Witek · 947

Kryvoruchka, Liudmyla · 694

Kumar Luhana, Kirshan · 104

Kuno, Yasushi · 741

Kurvinen, Einari · 344

Kynigos, Chronis · 81, 357, 369, 506, 686, 773

L

Laakso, Mikko-Jussi · 344, 954

Lee, Gary C. F. · 781

Lee, Young-jun · 295

Lepuschitz, Wilfried · 686, 791

Lilija Duoblienė · 868

Lodi, Michael · 901

Luc Anckaert · 868

M

Mäkiäho, Pekka · 698

Malchiodi, Dario · 901

Marshall, Kevin · 767

Martin, Kit · 380, 392

Martinez, Sylvia · 121, 755

Mascaró, Maite · 615

Massarwe, Khayriah · 657, 956

Matsuzawa, Yoshiaki · 703

Mayerová, Karolína · 623

Maytarattanakhon, Athit · 160

Michaeli, Tilman · 405

Monga, Mattia · 901

Mönig, Jens · 82

Moro, Michele · 498

Morpurgo, Anna · 901

Mueller, Matthias · 104

Muller, Eric · 528

Muñoz, Leda · 514

N

Nakano, Issei · 703

Nakayama, Yuriko · 741

Niemelä, Pia · 415

Nikitopoulou, Sofia · 773

Nikolova, Nikolina · 954

Noguchi, Misako · 703

Noss, Richard · 69, 754

O

O'Sullivan, Katriona · 138

Oie, Yuichi · 593

Olędzka, Katarzyna · 855

Ortega Torres, Enric · 707

P

Partanen, Tiina · 415

Passey, Don · 838

Patarakin, Evgeny · 426

Peppler, Kylie · 754

Pereira de Souza, Elmara · 631

Pereira, Patrick Pais · 672

Pesek, Igor · 884

Petrosino, Tony · 761

Pina, Alfredo · 498

Pluhár, Zsuzsa · 884

Pope, Nicolas · 437

Poranen, Timo · 415, 698

Posov, Ilya · 160

Poviliūnas, Arūnas · 4

Pozdniakov, Sergei · 160

Prieto-Rodriguez, Elena · 251, 577

R

Riley, Clare · 767

Romeike, Ralf · 285, 405, 855

Rubio, Gabriel · 498

S

Sabin, Mihaela · 925

Sabitzer, Barbara · 711

Sacristán, Ana Isabel · 83, 528, 615, 925

Sanjosé López, Vicent · 707

Schindler, Christian · 104

Schmidt, Laura · 672

Seegerer, Stefan · 405

Selcuk, Hasan · 543

Sendova, Evgenia · 94, 855, 941, 954

Sharkov, George · 649, 690

Shaw, Mia S. · 214

Slany, Wolfgang · 104, 637, 855

Solaz Portolés, Joan-Josep · 707

Souza Moura, Luísa · 631

Spieler, Bernadette · 104, 637, 901

Stager, Gary S. · 120, 121, 955

Standl, Bernhard · 678, 794

Staub, Jacqueline · 756, 955

Stroup, Walter · 761

Stupurienė, Gabrielė · 169

Suhre, Cor · 570

Sullivan, Kevin · 138

Suziedelis, Carol Sperry · 122, 956

T

Takeuchi, Yugo · 723

Tan, Michael · 714

Tangney, Brendan · 449

Thanapornsanguth, Sawaros · 460, 561, 718

Todorova, Christina · 649, 686, 690

Tohyama, Sayaka · 723

Tutiyaphuengprasert, Nalin · 727

V

Valente, José Armando · 472, 481

van Veen, Klaas · 570

Vaniček, Jiří · 488, 884

Varbanov, Pavel · 649, 690

Vartiainen, Katriina · 698

Verner, Igor · 657, 956

Veselovská, Michaela · 623

Vincze, Markus · 506, 686

Vinikienė, Lina · 731, 794

Visnovitz, Márton · 736

Vittori, Lisa · 690

W

Waite, Jan · 794

Watanabe, Takeshi · 741

Weigend, Michael · 664, 672, 884

Wetzinger, Elisabeth · 678

Wilensky, Uri · 123, 125, 238, 263, 274, 392, 554, 761

Wilfried Baumann · 868

Winters, Niall · 315

Y

Yiannoutsou, Nikoleta · 506, 649, 690, 773

Yoneda, Takashi · 593

Yoshida, Aoi · 325, 593, 745

Yoshida, Mari · 325

Yoshizawa, Minori · 325

Z

Zamora, Natalia · 514

Želvys, Rimantas · 20, 944

Zhang, Jinbao · 750

Zwaneveld, Bert · 570



ISBN 978-609-95760-1-5