Towards A Teaching Model for Managing Students' Cognitive Load In Activities That Integrate Computational and Mathematical Thinking

Giorgos Panselinas, Manolis Mavrikis, Eirini Geraniou, Popi Anastasiou, Emmanouil Tambouratzis, and Emmanouil Kartsonakis

***Abstract*—This paper proposes a model of teaching computational thinking as a sub-competence of a digital competence framework. This teaching model is based (a) on other models of teaching and learning programming aiming at managing students' cognitive load, (b) on exploiting the engaging nature of unplugged activities and (c) on using erroneous examples to address students' common errors and misconceptions. The teaching model emerged from the study of the implementation of the "Reach 20 first" competence assessment educational scenario at a Greek class with 11 students of low motivation and attainment regarding computing and mathematics. We investigated (a) the impact and the key-elements of the aforementioned teaching model on students' computational and mathematical thinking achievement and (b) the relationship between computational and mathematical thinking in computing activities. We present our findings discussing the possible implications on educational activities design and teacher support.**

***Index Terms*—Computational Thinking, Mathematical Thinking, Programming, Unplugged Activities, Teaching Model, Cognitive Load, Erroneous Examples.**

I. Introduction

In recent years there has been a widespread trend for the teaching of computer programming in both primary [1] and secondary education [2]. The aim is to help students develop computational thinking (CT), a skill that is not exclusively limited to computer science, but it concerns "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" [3].

CT draws on both mathematical thinking and engineering thinking, as it refers to task, data collection and analysis, data representation; logical reasoning, abstraction, algorithm design, decomposition, parallelization, automation, pattern generalization, pattern recognition, simulation [3].

Though computational and mathematical thinking hold a reciprocal relationship [4] we are concerned about the lack of strong academic background taking into account the challenges around cognitive load [5] and the role of mathematical thinking when undertaking computational thinking activities.

In our efforts to make this relationship explicit in students' eyes, we consider the need for specially designed classroom activity that supports students' developmental path to computational and mathematical thinking, while providing a 'bridge' for connecting computational and mathematical thinking.

This paper reports on a study that sought to address the following questions:

What is the impact of 'Reach 20 first' educational scenario on students' computational thinking sub-competence acquisition?

How computational and mathematical thinking were integrated when students investigated the generalization of the solution of a given computational thinking problem? How does mathematics contribute to computational thinking within this context?

What is the teaching model embedded in 'Reach 20 first' educational scenario that supports computational thinking? What key-elements of this teaching model support computational thinking sub-competence acquisition and computer science learning?

We look at the data from a classroom project implemented with eleven 15-year-old students working on the "reach 20 first" educational scenario. We present their results, reflecting upon the interplay between their computational and mathematical thinking and propose a teaching model. We finally discuss the properties of the proposed teaching model focussing on possible implications on instructional design.

II. Background

The educational scenario (Competence Assessment Scenario-CAS) uses an unplugged activity [6, 7] in order for students to design solutions to a computational problem [8].

Computational thinking requires a set of thinking skills such as Data collection and Analysis, Abstraction and Pattern recognition, Algorithmic design, Programming and Debugging, in order to implement a computer game strategy. Furthermore, the application of the "Reach 20 first" CAS helps in scaffolding mathematical thinking as it

Published on February 17, 2020.

G. Panselinas is with the Regional Directorate of Primary & Secondary Education of Crete, Heraklion, Greece (e-mail: panselin@gmail.com).

M. Mavrikis is with the UCL, Institute of Education, London, U.K (e-mail: m.mavrikis@ucl.ac.uk)

E. Geraniou is with the UCL, Institute of Education, London, U.K (e-mail: e.geraniou@ucl.ac.uk)

P. Anastasiou is with the UCL, Institute of Education, London, U.K (e-mail: p.anastasiou@ucl.ac.uk)

E. Tampouratzis is with the 1st Vocational High School of Agios Nikolaos, Agios Nikolaos, Greece (e-mail: tampouratzis@gmail.com)

E. Kartsonakis is with the Regional Directorate of Primary & Secondary Education of Crete, Heraklion, Greece (e-mail: manolis.kartsonakis64@gmail.com)

is used for pattern matching and for generalising a solution from a single computational problem to the solution to a whole class of similar problems [9].

The Competence Assessment Scenario-CAS is an educational scenario that was designed in the framework of the CRISS Project (https://www.crissh2020.eu/) in order to teach, evaluate and certify computational thinking as a sub-competence of the CRISS competence framework that is based on the European Digital Competence Framework for Citizens-DigiComp 2.0 [10,11].

CAS's teaching strategy uses transition through levels of abstraction (Formulating the problem to be solved through Execution of a program, Solving the problem through unplugged activity for pattern recognition to algorithm design in Greek, modelling the solution of the problem by modifying the first program) and use-modify-create approach [12] to manage students' cognitive load [5]. Also with the use of the Unplugged activity we have created an authentic and engaging context teaching several critical concepts such as algorithm, program, programmer, programming language and artificial intelligence [6, 7].

The key idea of fun, kinaesthetic, highly engaging "unplugged activities" enabling students, during one-hour class to explore computer science without having to first learn programming [13, 14] is used to pursue computational thinking that includes algorithmic design and Coding. We agree that computational thinking cannot be perceived without "an information-processing agent" that is human or machine [3].

Certain points during the solution of tasks that involve computational thinking also required mathematical thinking. To facilitate that, there is a need for specially designed activities and resources to support students' transition from computational to mathematical thinking and vice versa. This support requires appreciating the fact that transition between conceptual boundaries (such as mathematical and computational thinking) is not a straightforward process. For example, in the case of Logo, Gurtner [15] had considered "the type of connections generally expected, and very seldom observed, between Logo practice and mathematics" as transfer and suggested that "a rather long period of Logo practice (one that is rich in reflection) is necessary before transfer to mathematics can occur [16]. As such Gurtner's "bridging" metaphor is useful here to help us describe the connections students or educators can make between different domains and one way of building such bridges is through purposefully designed activities such as worksheets, collaboration activities to help the interplay between computational and mathematical thinking to become 'visible' [17].

III. Methodology

A. The study

The educational scenario was implemented in a class of 11 students of a vocational high school in Greece. As computing and mathematics class teacher told us in an interview, these students have had low motivation and attainment regarding Computing and Mathematics learning. This view is supported by Koutsampelas and Tsakloglou [18] who inform us that Greek vocational students "are usually located at the lower part of the income distribution".

The educational Scenario was implemented in two successive days in May of 2019 using 5 lessons of 40 minutes. In lesson 1, students played a PC game (human against PC) that is called "Reach 20 first". The game is played with one pawn starting from step 0 and each of the two players, on their turn, move the pawn one or two steps forward. The one that reaches the step 20 first is the winner (Fig. 1).

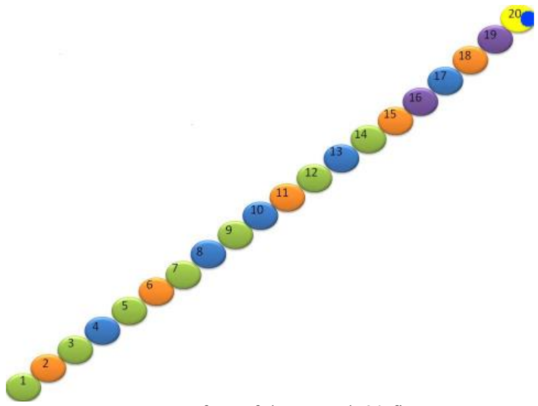

Fig. 1. User Interface of the "Reach 20 first" game

The computer (PC) plays always first. The game has been coded in Snap! [19]. Most students managed to defeat the computer. Then, in lesson 2, they played 3 times against an "intelligent paper" that contains a winning algorithm. Each time, students are asked to write down how the intelligent piece of paper and the human interact constructing the representation of a ladder along with human and intelligent piece of paper moves (Fig. 2).

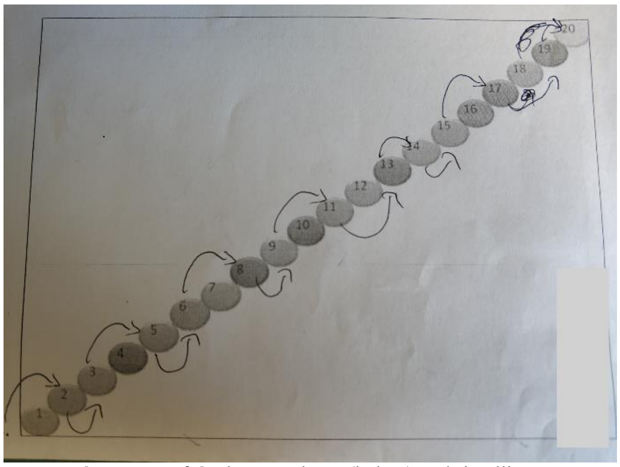

Fig. 2. The moves of the human player (below) and 'intelligent paper' (above) as represented with 'arrows' on a 'ladder'

The aforementioned procedure helped students to recognise patterns and lead them, in Lesson 3, to design collaboratively in small groups an algorithm in Greek language that supposed to beat every human that plays against it. They tested their algorithms in paper the same way they tested the "intelligent paper", each time modifying their algorithm. Finally, they individually typed their algorithm on a word processor and uploaded it to the CRISS platform for evaluation by the teacher.

Below you can see an algorithm designed by a student:

1. As soon as the paper plays first, the paper takes 2

steps forward.

2. Wait for the human to play
3. If human takes 1 step the paper opts for the opposite, that is 2 steps
4. Repeat the above until the paper reach 20!

1) Extract 1: The final algorithm of Student 9

In lesson 4, students modified the code of the first program implementing the game in order to make the PC unbeatable in "Reach 20 first" and uploaded a screenshot on CRISS platform (Fig. 3).

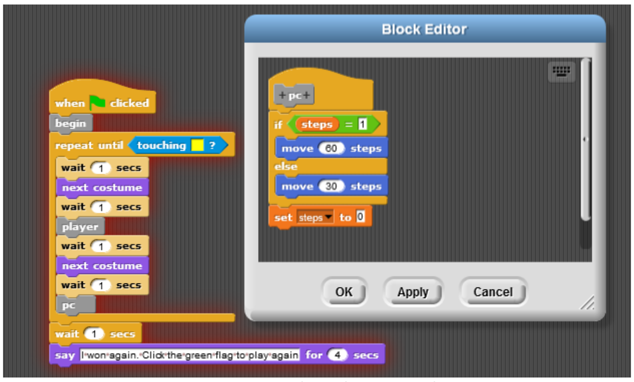

Fig. 3. Screenshot of the final modified code

Finally, in lesson 5, students are given a worksheet and are asked to answer the following questions:

Express in your own words, using mathematical language where possible, why the algorithm is beating every human player. The following questions are there to help you:

1. Observe that in each case the sum of both the human and the PC steps are 3. What are the specific steps that the PC follows in order to win?
2. Would the PC have won if its first step was other than 2?

Investigate the general solution:

3. Play a game with different numbers e.g. target 30 and steps 2 and 5. Can you describe the strategy in order for the PC to always win?
4. Does this strategy work also for target 30 and steps 3 and 4?
5. Can you designate a general rule in order for this strategy to work?

Hint: If S is the sum of possible human step plus PC step in each cycle of the game and T is the target, what is the remainder of this division T/S? What is your conclusion?

2) Extract 2: The worksheet

One of the authors (GP) was the teacher and creator of the educational scenario, while the usual computing teacher of the class kept down observation notes regarding students' interaction. The teacher as the 'more knowledgeable other' scaffolded classroom talk through different levels of abstraction to mathematical generalisation that was manifested (or not) by students' answers to the questions posed by the above mentioned individual worksheet [20, 5].

B. Data Collection and Analysis

The data comprises students' notes on intelligent paper plays (e.g. Fig. 2), students' algorithms (e.g. Extract 1), screenshots of the final code (e.g. Fig. 2), their answers on the aforementioned worksheet, observation notes from students' interactions, as well as an interview with computing and mathematics class teachers. Student names were coded for anonymity. Those data were used for interaction and content analysis in a way that validates results through triangulation of data [21].

IV. Results

A. Students' computational thinking sub-competence acquisition

We evaluated the students' notes on intelligent paper plays, the final algorithms of the students and the modified code from the screenshots using rubrics. We found out that 5 out of 11 students have designed an algorithm that can lead a human "information-processing agent" [3] to win in "Reach 20 First" game and 7 out of 11 students have modified the code in the right way to make the PC win the game against every human. Thus, 5 out of 11 students were certified for the computational thinking sub-competence by the CRISS digital competence framework [11].

B. Integrating computational and mathematical thinking

We analysed the observation notes and the written answers on the worksheet of the 9 of the students who filled it in, in order to respond to our second research question.

B.1. Why the algorithm is beating every human? (a) Observe that in each case the sum of both the human and the PC steps are 3. What are the specific steps that the PC follows in order to win? (b) Would the PC have won if its first step was other than 2? Below we cite the written answers of the 6 students that showed some understanding - in written answers- of how the winning algorithm is working. In italics you will read some wrong estimation.

Student 1: "Because as soon as the algorithm plays first and its first move is two, then it moves the pawn to the same steps each time we play against it". "These specific steps are 0,2,5,8,11,14,17,20". "I think not, because it would not be able to step to the same specific numbers, this change would have broken the 'chain' (of events)"

Student 3: "PC wouldn't have won because if the PC played other than 2 then the human would take the chance to move the pawn to the specific right steps (numbers) and win"

Student 5: "Because it moves the pawn to the same steps each time we play against it"

Student 7: "0,2,5,7,11,14,17,20", "the PC wouldn't have won because it would lose its lead and the specific numbers"

Student 8: "Because whatever we do the PC (algorithm) puts us in the same difficult position of 17 and from this step on whatever we do we lose", "*I think that PC would have won even if its first step was other than 2*"

Student 9: "The specific steps are 0,2,5,7,11,14,17,20", "the PC would lose if its first step was other than 2 because it always follows certain steps and if its first step was 1 this change would have broken the "chain" (of events)"

1) Extract 3: The written answers that show some understanding of why the algorithm is beating every human

In general, the answers demonstrate that most of the students who filled in the worksheet acquired some level of understanding that the winning strategy relies on starting first and making a certain move that permits an 'adaptive' strategy of moving to certain positions and finally to a position from which human players could only lose (in this case step 17).

B.2 Investigate the general solution. Only two students provided written answers that showed some understanding of the general solution. In italics you will read some wrong estimation.

Student 1: (a) "Yes there is a strategy: it has to step on 2,9,16,23,30, *first move 2 steps and then 7 steps*", (c) "It has to start moving 2 steps and then follows a certain strategy whether the target is 20 or 30.", "One of the two steps must be the remainder of the division 20=6X3+2, 30=4X7+2. The number 2 is left over"

Student 9: (a) "the strategy is to start with 2 steps and then moves by 7", (b) "No it doesn't" (c) One of the steps must be the remainder of the division of the target number by the sum of the two possible steps"

2) Extract 4: The written answers that show some understanding of the general solution

As we can see from the aforementioned data 6 out of 11 students managed to gain some understanding of how the winning algorithm works but only 2 of them managed to try a general solution. By using the observation notes of the scaffolded interaction among the teacher and the students we can see that in the last minutes of the lesson the teacher, student 1 and student 9 had the following discussion:

Teacher: If S is the 'sum of possible human step' plus 'PC step' in each cycle of the game and T is the target number, what must be the remainder of this division T/S in order to apply our strategy?

Student 8: These are mathematics. I don't know mathematics

/*Teacher writes on the whiteboard (3X6)+2=20 with steps 1 and 2 and (4X7)+2=30 with steps 2 and 5. Earlier he has drawn two representation of ladders (e.g Fig. 2) with target numbers 20 and 30 and steps 1 or 2 and 2 or 5 accordingly*/

Teacher: What must be the remainder of the division of the target number by the sum of the two possible moves?

Student 1: It must be 2

Student 8: The first move must be the remainder of the division!

Teacher: Right!! Must be one of the possible moves!!

3) Extract 5: Student 9 finds the general solution

It is interesting that mathematics in this scenario were used to generalize a solution to a given problem in order for the students to come up with a general rule that will help them design the solution-algorithm for a whole class of problems. Mathematics was once more difficult for these students but the scaffolding that led to the 'aha' moment for student 9 is successful. However, it is interesting that she prefers to "speak" mathematically in her language rather than write mathematical symbols.

C. The proposed teaching model

Our third research question required us to look at the teaching model embedded in this scenario. Reflecting on this case points out a teaching model for computational thinking and programming at school that is described below:

1. RUN and observe how a program with a logical flaw interacts with a human, thus formulating the problem to be solved
2. Declare that an "intelligent piece of paper" [6, 7] interacts with humans in a different and desirable way that solves the aforementioned problem. INVESTIGATE how the "intelligent piece of paper" interacts with humans by *executing* the algorithm of the intelligent paper three or more times, each time interacting with different individuals. Write down how the intelligent paper and the human interact each of the 3-4 times *constructing a suitable representation*. This helps students to *recognise patterns* that will lead them to the next step, that is to
3. DESIGN an algorithm collaboratively and/or individually in their own piece of paper that interact with humans the same way the "intelligent piece of paper" does (first in some human language).
4. Then RUN and DEBUG the students' algorithms with the help of a human as paper 'servant' actually their algorithm 'servant'. The students' algorithms are tested against different humans. Each time the students *modify* their algorithms so as to be right and rigorous, thus making the transition from human language to CS speak [22].
5. Finally, students MODIFY the code of the first flawed program so as to interact with humans the way the first "intelligent piece of paper' does.
6. Students can use then *pattern matching and/or mathematics* to MAKE a new program that solves a different problem.

V. Discussion and Conclusion

A. The key-elements of the proposed teaching model and possible implications

The aforementioned teaching model uses the Run, Investigate, Modify and Make phases of the PRIMM teaching and learning model of Sentance et al. [5] but only Modify and Make in the same way. Students Run and Investigate how a program with a logical flaw is running and thus formulating a problem to be solved. It follows an unplugged activity with an "intelligent piece of paper" that helps to Investigate how the program should run. Then students Design and Test an algorithm in human language towards a CS-speak language. This algorithm informs the Modification of the code of the flawed program. In this way, the teaching model we propose uses the Level of Abstraction (LoA) framework as last modified by Armoni [23] with four levels: 'execution'; 'program'; 'algorithm'; 'problem', Abstraction Transition Taxonomy teaching model [22] and use-modify-create approach [12] to manage students' cognitive load [5]. Furthermore, through the MAKE phase it has the potential to support students'

DOI: http://dx.doi.org/10.24018/ejers.2020.0.CIE.1792

generalisations in a way that nurture their creativity giving them ownership of problems and solutions. Also, with the use of the unplugged activity we have created an authentic and engaging context for teaching several critical computer science concepts such as algorithm, program, programmer and programming language along with computational thinking and programming [6, 7].

In the aforementioned example of "Reach 20 first" scenario apart from using computational thinking to solve a problem, students tackled effectively with the misconception of the allegedly inherent artificial intelligence on computers [6, 7]. Therefore, due to Run and Test Phase of a flawed program we propose that the aforementioned teaching model can also use erroneous examples for presenting students with common errors and misconceptions in a way that supports conceptual change [24].

Furthermore, this teaching model "creates the problem in the students' minds" because they need their own problem to learn programming as Guzdial [25] says.

The aforementioned properties of the teaching model in hand may inform the instructional design of educational scenarios that share the aforementioned properties and support computational thinking, mathematical thinking while also deal with basic computer science teaching concepts and misconceptions.

B. Computational and Mathematical thinking

Our first and second research question revolved around the impact of the 'Reach 20 first' scenario on students' computational thinking and the reciprocal relationship of computational and mathematical thinking. We were particularly concerned with the lack of strong academic background and motivation in the cohort under investigation and thus structured the teaching model in a way that would manage students' cognitive load. We were aware of both the challenges around cognitive load and the barriers of mathematical thinking when undertaking computational thinking activities. The results showed that most of the students engaged with the unplugged activity that facilitated their subsequent modification of the code in Snap!. The teaching model, scaffolding through talk [20] and the resources including the worksheets were instrumental in supporting the students to reach that far. On reflection, an introduction of certain concepts through previous examples or other 'bridging' strategies [17] could have helped the students more to express their ideas mathematically. In addition, perhaps with more sufficient time the transition to generalisation could be smoother. We see the proposed teaching model as an opportunity for students to express algebraically the rules that underpin a problem before further abstracting it and implementing more general solutions.

VI. Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732489. We would like to thank Ioannis Kanellos who gave us the first idea of how to use mathematics in "Reach 20 first" educational scenario.

References

[1] L Benton, C Hoyles,, I Kalaš, and R Noss (2017). Bridging Primary Programming and Mathematics: Some Findings of Design Research in England. Digital Experiences in Mathematics Education, 3, 115–138.

[2] L Calao, A Moreno-Leon, J.H.E Correa and G Robles, 2015 Developing Mathematical Thinking with Scratch: An Experiment with 6th Grade Students. Design for Teaching and Learning in a Networked World, 17-27. Springer International Publishing.

[3] J.M Wing (2011). Research Notebook: Computational Thinking--What and Why? The magazine of Carnegie Mellon University's School of Computer Science.

[4] D Weintrop, E Beheshti, M Horn, K Orton, K Jona, L Trouille and U Wilensky (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127-147. doi:10.1007/s10956-015-9581-5

[5] Sue Sentance, Jane Waite & Maria Kallia (2019) Teaching computer programming with PRIMM: a sociocultural perspective, Computer Science Education, 29:2-3, 136-176, DOI: 10.1080/08993408.2019.1608781

[6] P McOwan and P Curzon. The intelligent piece of paper. Queen Mary University of London. http://www.cs4fn.org/teachers/activities/intelligentpaper/intelligentpaper.pdf

[7] G Panselinas and E Tiliannakis. 2017. Intelligence, Games, Algorithms and Computers: Making the computer always win at "Reach 20 first" game. In Proceedings of the 11th Panhellenic Conference of Greek Computing Teachers Association, Chalkida, http://pdkap.sch.gr/2017/wp-content/uploads/2017/05/pekap2017-final43.pdf (in Greek)

[8] V Barr and C Stephenson (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads, 2, 48–54

[9] P Curzon and P.W McOwen. 2017. The power of Computational Thinking, 1st ed. New Jersey: World Scientific, DOI: https://doi.org/10.1142/q0054

[10] S Carretero, R Vuorikari and Y Punie. 2017. The Digital Competence Framework for Citizens, Available at: http://publications.jrc.ec.europa.eu/repository/bitstream/JRC106281/web-digcomp2.1pdf_(online).pdf Pdf

[11] M Mavrikis, L Guardia, M Cukurova and M Maina, on behalf of the CRISS Consortium .2018. A Digital Ecosystem for Digital Competences: The CRISS Project Demo. In: Pammer-Schindler V., Pérez-Sanagustín M., Drachsler H., Elferink R., Scheffel M. (eds) Lifelong Technology-Enhanced Learning. EC-TEL 2018. Lecture Notes in Computer Science, vol 11082. Springer, Cham

[12] I Lee, F Martin, J Denner, B Coulter, W Allan, J Erickson, L Werner (2011). Computational thinking for youth in practice. ACM Inroads, *2(1), 32*

[13] T Bell, J Alexander, I Freeman and M Grimley (2009). Computer science unplugged: school students doing real Computing without computers. *New Zealand Journal of Applied Computing and Information Technology, 13(1), 20–29.*

[14] P Curzon, P.W McOwen, Q Cutts and T Bell (2009). Enthusing and Inspiring with Reusable Kinaesthetic Activities. In *Proceedings of the ITICSE* 2009.

[15] J.L. Gurtner. 1992. Between Logo and mathematics: A road of tunnels and bridges. In: Hoyles C. & Noss R. (eds.) Learning mathematics and Logo. MIT Press, Cambridge MA: 247–268.

[16] G Salomon and D.N Perkins (1987) Transfer of cognitive skills from programming: When and how? Journal of Educational Computing Research 3, 149–169.

[17] E Geraniou and M Mavrikis (2015). Building Bridges to Algebra through a Constructionist Learning Environment. Constructivist Foundations, 10 (3), 321-330.

[18] Christos Koutsampelas & Panos Tsakloglou (2015). The progressivity of public education in Greece: empirical findings and policy implications, Education Economics, 23(5),596-611, DOI: 10.1080/09645292.2014.884999

[19] B Harvey, D Garcia, J Paley and L Segars. 2012. Snap!: (build your own blocks). In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, ACM (2012), p. 662

[20] G Panselinas and V Komis (2009). Scaffolding through talk in groupwork learning. Thinking Skills and Creativity, 4, 86-103

[21] U Flick, Triangulation in qualitative research in U Flick (2004). Triangulation in qualitative research. A companion, 2004, pp. 178–183.

[22] Q Cutts, S Esper, M Fecho, S.R Foster and B Simon. 2012. The abstraction transition taxonomy: developing desired learning

outcomes through the lens of situated cognition. In: Clear, A., Sanders, K. and Simon, B. (eds.) Proceedings of the Ninth Annual International Conference on International Computing Education Research, Auckland, New Zealand, 10-12 Sept 2012. ACM: New York, NY, USA, pp. 63-70. ISBN 9781450316040 (doi:10.1145/2361276.2361290)
[23] M Armoni (2013). On teaching abstraction in computer science to novices. Journal of Computers in Mathematics and Science Teaching, 32(3), 265–284.
[24] D Tsovaltzi, B.M. McLaren, E Melis, and A.K. Meyer (2012) Erroneous examples: effects on learning fractions in a web-based setting, Int. J. Technology Enhanced Learning, 4 (3/4), 191–230.
[25] M. Guzdial, 2019. Computer Science Teachers as Provocateurs: All learning starts from a problem. Computing Education Research Blog. https://computinged.wordpress.com/. Last access on 14/6/2019.

Giorgos Panselinas holds a degree in Computer Science (1992), a Master's degree in Education (Didactics of Computer Science) (2000) from the University of Crete Greece, a PhD degree in Education (ICT in Education) from the University of Patras Greece (2006) and a second Master's degree in Adult Education (2014) from the Hellenic Open University. For his PhD he studied 'scaffolding' through talk in groupwork computer assisted learning.

He has been a teacher of Computer Science in Greek secondary education since 1994. He has been a school advisor in Secondary Education since 2007. He has worked in a number of R&D Projects relevant to design, development, and assessment of educational software, Information and Communication technologies (ICT) in Education and Computer Science in Education. He is the Editor of the training material of the "train the trainers" course for ICT/Computer Science teachers' trainers. His publications and research interests concern the sociocultural approach to teaching-and-learning in schools with particular interest in educational discourse, computer-assisted learning and computing at school.

Dr. Panselinas has conceived the idea and he is the main coordinator of the Pan-Hellenic Computing at School Festival (www.digifest.info).

Manolis Mavrikis is a Reader in Learning Technologies at the UCL Knowledge Lab. He holds a B.Sc. in Mathematics from University of Athens, Greece with an emphasis in education, M.Sc. with distinction in Informatics and Ph.D. in Artificial Intelligence in Education from the University of Edinburgh. His research interests developed over 20 years of experience lie at the intersection of learning sciences, human–computer interaction and artificial intelligence. Manolis is currently an Editor-in-Chief for the British Journal of Education Technology and director of the MA in Education and Technology at UCL

Eirini Geraniou is an Associate Professor in Mathematics education at the UCL Institute of Education. She holds a B.Sc. in Mathematics from University of Crete, Greece and an MSc and a PhD in Mathematics Education from the University of Warwick, UK. Her research interests focus mainly on the use of digital technologies for the teaching and learning of mathematics. She implements mathematics education research in actual classroom settings, supporting students as well as teachers throughout their interactions with innovative digital media. Eirini is currently the subject lead of the Teach First teacher training programme at UCL and also leads the master's module for Digital Technologies for Learning Mathematics at UCL Institute of Education

Popi Anastasiou is a Greek-Cypriot educator, currently working as a researcher at the CRISS 2020 Project, part of the UCL Knowledge Lab Team. Anastasiou obtained a bachelor's degree in Primary Education from the University of Thessaly in 2006, and a first Master of Arts (MA) in Education Management from Kings College London (KCL) in 2007. The author continued with two more master's degrees; a Master of Research (MRes) in Education and Social Sciences from KCL in 2008, and a Master of Sciences (MSc) in Digital Education from the University of Edinburgh in 2015. Anastasiou has recently submitted her doctoral thesis in Educational Technology at the Open University UK (2019).

She has a long research experience working in both European and local projects, she has also contributed to the design and creation of the Badged Open Course (BOC) "Teaching and Learning Tricky Topics", available at the Open University website. In the meantime, she has peer-reviewed papers for prestigious journals and conferences. In addition, she has worked as a primary teacher both in the supplementary schools of the Greek-Cypriot community in North London and in public schools in Cyprus. Finally, she worked as a part-time lecturer teaching qualitative research methods at the master's degree in Special Education Needs at the University of Nicosia.

Emmanouil Tampouratzis was born in Piraeus in 24/11/1976. In 2001 he got a degree in Department of Computer Engineering of the Technological Educational Institute in Piraeus. He fulfilled his military services in 2002. From 6/2000 until 3/2001 he worked as a Computer Technical in UNIFON S.A. From 9/2002 until 6/2003 he worked as a Teacher of Computer Sciences in Single High School of Agios Nikolaos. From 9/2003 until 6/2004 he worked as a Teacher of Computer Science in 11th Single High School in Piraeus. Finally, from 9/2004 until today he is working as a Teacher of Computer Science in Vocational High School of Agios Nikolaos. He has participated and has attended in many seminars and trainings. He has awarded with the: 1st prize for the digital essay "Website for Hydrobot" (20/10/2012)

Emmanouil Kartsonakis studied Electrical Engineering at the Department of Electrical Engineering, Technological Educational Institute of Crete (1982-1985), then at the Department of Physics, University of Crete (1985-1989), followed by his doctoral thesis at the Department of Philosophy and Social Studies of the University of Crete in History and Philosophy of Sciences (1989-1994). He has been working in public education since 1990.

He has worked as a Research Associate in Departments of TEI of Crete (1995-2011) and as Associate Lecturer at the Hellenic Open University (2006-2018). He has lectured and taught at European universities as a Visiting Professor. He has presented many papers in international and national conferences, he has published 35 papers in international journals and conference proceedings and he has authored and translated scientific books. Member of conference committees, an independent reviewer of journals and a member of many scientific associations. He has been involved in national research projects and he has an extensive experience in European programs.