Association for Information Systems
AIS Electronic Library (AISeL)

MCIS 2016 Proceedings

Mediterranean Conference on Information Systems (MCIS)

2016

A Framework and an Instructional Design Model for the Development of Students' Computational and Algorithmic Thinking

Ioannis Ioannou
University of Cyprus, ioannis.ioannou@ucy.ac.cy

Charoula Angeli
University of Cyprus, cangeli@ucy.ac.cy

A FRAMEWORK AND AN INSTRUCTIONAL DESIGN MODEL FOR THE DEVELOPMENT OF STUDENTS' COMPUTATIONAL AND ALGORITHMIC THINKING

Research in Progress

Ioannis, Ioannou, University of Cyprus, Nicosia, Cyprus, ioannis.ioannou@ucy.ac.cy
Charoula, Angeli, University of Cyprus, Nicosia, Cyprus, cangeli@ucy.ac.cy

Abstract

The authors herein, describe their efforts towards designing technology-enhanced instruction for teaching Computational and Algorithmic Thinking. This study examined students' development of Computational and Algorithmic Thinking, by utilizing the framework of Technological Pedagogical Content Knowledge and the instructional design model of Technology Mapping. Different technological tools were used for both groups of participants; the experimental and the control group. In particular, the experimental group used educational robotics and the control group used a 3D interactive programming environment. Both groups were 8th graders coming from different secondary education schools in Cyprus. A pre-post test research design was adopted in each classroom intervention. To check whether the interventions facilitated students' development and understanding of Computational and Algorithmic Thinking concepts and competencies, an analysis of covariance (ANCOVA) was then conducted. According to the results, the framework of Technological Pedagogical Content Knowledge and the approach of Technology Mapping, which guided the design of the instructional intervention were effective in terms of fostering students' development and understanding of Computational and Algorithmic Thinking competencies and concepts, respectively.

Keywords: Computational Thinking, Algorithmic Thinking, Computer Science Teaching, Technological Pedagogical Content Knowledge, Technology Mapping.

1 Introduction

A number of studies (Dagdilelis, Satratzemi, & Evangelidis, 2004; Ioannou & Angeli, 2013) show that learners have misconceptions regarding the curriculum of secondary education computer science, such as, misconceptions on programming languages and basic computing concepts. In this study, the authors adopted the framework of Technological Pedagogical Content Knowledge (TPCK) and the instructional design model of Technology Mapping (TM), as proposed by Angeli and Valanides (2005), in order to redesign the lessons related to the teaching of the development of Computational and Algorithmic Thinking by utilizing the affordances of different educational software. In particular, educational robotics were used, namely Robomind (v4.3) and Lego NXT Mindstorm, as well as, a 3D interactive programing environment, namely Alice (v3.1). Robomind 4.3 is a Logo-like software with very simple programming language for novice programmers.

2 Literature

2.1 Computational Thinking

The term "Computational Thinking" was coined by Jeannette Wing in 2006, and since then, various discussions have risen seeking a robust definition. Many educators (Computer Science Teachers Association Task Force, 2011) and academics (National Research Council, 2010; Wing, 2007) have worked along with Wing's definition, proposing that Computational Thinking is a fundamental skill for everyone, not just for computer scientists. Computational Thinking should be added to every child's analytical ability, such as reading, writing and arithmetic (Wing, 2006).
Cuny, Snyder, and Wing (2010) stated that Computational Thinking is "the thought processed 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" (p. 1). Computational Thinking is equivalent to the logical reasoning (Henderson, Cortina & Wing, 2007), and also intersects with engineering as computers interact with the real world (Wing, 2011). It could be considered as a way of problem solving because it incorporates the same set of mental tools used in computer science (Wing, 2006). These tools are usually used to transform a difficult problem into one that can be solved more easily.

2.2 Technological Pedagogical Content Knowledge (TPCK)

Technological Pedagogical Content Knowledge (TPCK) was introduced to the education research community as a domain-general theoretical framework of what teachers need to know to teach with technology (Mishra & Koehler, 2006; Angeli & Valanides, 2005). The authors herein adopt the transformative model of TPCK for guiding the design of the lessons regarding the teaching of the development of Computational and Algorithmic Thinking. The transformative model, shown in Figure 1, conceptualizes TPCK as a unique body of knowledge where significant contributors to the development of TPCK are: content, pedagogy, learners, technology, and context. TPCK as a transformative body of knowledge is defined as knowledge about how to transform content and pedagogy with ICT for specific learners in specific contexts and in ways that indicate the added value of ICT (Angeli & Valanides, 2005). TPCK, as a unique body of knowledge, is better understood in terms of competencies that teachers need to develop in order to be able to teach with technology adequately (Angeli & Valanides, 2005). These competencies are related to knowing how to:

1. Identify topics to be taught with ICT in ways that signify the added value of the ICT tools.
2. Identify appropriate representations for transforming the content to be taught into forms that are pedagogically powerful and difficult to be supported by traditional means.
3. Identify teaching tactics, which are difficult or impossible to implement by other means.
4. Select tools with appropriate affordances to support 2 and 3 above.
5. Infuse computer activities with appropriate learner-centred strategies in the classroom.

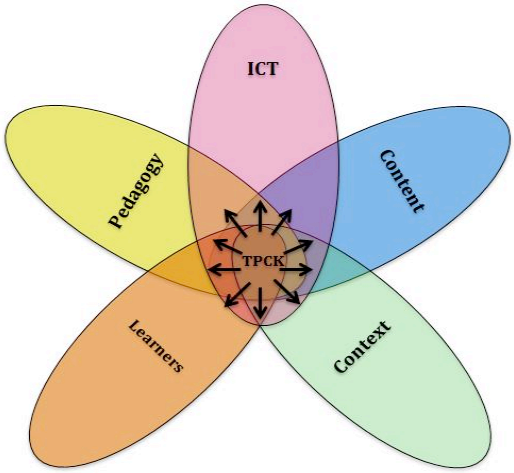

Figure 1. Technological Pedagogical Content Knowledge framework (adopted from Angeli & Valanides, 2005)

2.3 Technology Mapping (TM)

Technology Mapping (TM), was introduced as an approach for developing teachers' TPCK (Angeli & Valanides, 2005). TM was proposed as an approach for mapping tool affordances onto content and pedagogy in powerful and transformative ways, enabling teachers to develop complex and interrelated ideas between the affordances of technology and their pedagogical content knowledge (Angeli & Valanides, 2005). TM can engage learners in a process of developing technological solutions to pedagogical problems by aligning teachers' PCK with knowledge about the affordances and constraints of various computer-based technologies. TM is the iterative process of determining relations or linkages among the affordances of a technological tool, content, and pedagogy taking into account learners' content-related learning difficulties. Mapping refers to the process of establishing such connections or linkages.

3 Purpose of the Study

The last few years, many researchers at the domain of Computer Science are trying to define Computational Thinking. Simultaneously, they have conducted several studies aiming students' development Computational and Algorithmic Thinking, through the exploitation of various tools and pedagogical strategies. In this study, the theoretical framework of TPCK and the instructional design model of TM were adopted, in order to redesign the lessons related to the teaching of the development of Computational and Algorithmic Thinking. The main purpose was to examine whether the framework of TPCK and TM was sufficient to eliminate students' misconceptions about the concept of algorithms and branching structures (Ioannou & Angeli, 2013), as well as, to effectively develop their Computational Thinking skills.

4 Methodology

4.1 Participants

Two hundred and forty 8th graders, participated in the study, one hundred and twenty-seven (52.9%) constituted the Experimental Group, and one hundred and thirteen (47.1%) the Control Group. One hundred and twenty-one were males (50.4%) and one hundred and nineteen were females (49.6%). Students were taught by eight (four females and four males) secondary education Computer Science teachers with more than ten years of teaching experience each, who were trained and prepared accordingly to meet the needs of this study.

4.2 Teaching Intervention

4.2.1 Lessons 1 and 2

The main learning objective for the first two lessons was that students understand the concept of the Algorithms, the methods of representation Algorithms (Verbal Description, Pseudocode and Logical Diagram), the characteristics that designate an Algorithm and the steps for the development of a program. At the beginning of the first lesson all students of the experimental group and control group answered a pretest which had a duration of 10 minutes. For the understanding of the concept of Algorithm, experimental group students utilized Robomind. They executed a comprehensive program and they had to observe how the program worked, and verbally describe in the correct order all the steps. Throughout this activity, students were puzzled concerning on how they should describe the steps taken by the virtual robot in the program they executed. This activity assisted them to understand the concept of Algorithm (i.e. a series of commands for solving a problem).
Afterwards, students were divided in three groups and worked with different activities utilizing a word processing file, in order to discover the three different methods of representation algorithms. Group A, dealt with activities related to verbal description of Algorithms, Group B dealt with activities related to Pseudocode and Group C dealt with activities related to Logical Diagrams. Through these activities, students had the opportunity to reflect and discover the three methods of representation Algorithms. Finally, each group presented and explained to the whole class the method of representation of Algorithms they dealt with.
During the second lesson, students continued to working in three groups (the same groups as in the previous lesson). Each group utilized Robomind and executed a different program where they had to observe whether it consists of any probable programming mistakes (for example, it repeats by mistake the same commands and never terminates). Then, their Computer Science teacher executed these different programs and explained step by step to the whole class the different types of programming mistakes these programs had. Through this activity, students had the opportunity to comprehend the characteristics that must designate an Algorithm (i.e. clarity, effectiveness and permeability).

4.2.2 Lesson 3

The main learning objective of the third lesson was to learn the four phases (Phase 1: Define the Problem, Phase 2: Define the Steps for Solving the Problem, Phase 3: Conversion of the Steps in Program and Phase 4: Check the program for mistakes) of the development of an Algorithmic Implementation. Students must be able to analyze a simple sequential structure problem, to formulate the algorithm in verbal form and program it, utilizing a simulating software (Robomind and Lego NXT Mindstorm – Experimental Group/Alice - Control Group).
Initially, a small scenario-problem was given to students, where they had to program it following the four phases of the development of an Algorithmic Implementation. This scenario, omit from students: "Apply the four phases of the development of an Algorithmic Implementation and program the virtual robot utilizing the simulation software of Robomind 4.3 in order to draw on the floor of the virtual map a white rectangle 4X4."
For purposes of consolidation and evaluation, students worked with Robomind 4.3 and implemented a number of relative activities. Ending these activities, students were separated in four groups and exported their code into Lego NXT Mindstorm. Then, they activated Lego NXT Mindstorm and executed the program utilizing the field which was especially designed for the purpose of these lessons. Through these activities students had the opportunity to understand the additional value of programming.

4.2.3 Lesson 4

The objective of this lesson was to enable students to analyze a simple branch structure problem, to formulate the algorithm in verbal form and then program it utilizing a simulating software (Robomind

4.3 and Lego NXT Mindstorm – Experimental Group/Alice 3.1 - Control Group). In this fourth lesson, the same tools were used and the same methodology was followed. Students worked individually using a worksheet that contained several exercises. The basic activity they worked on asked the students to program (following the four phases of the Algorithmic Implementation) the virtual robot using a branch structure to move through the map and stop when it meets the beacon using two different maps. Throughout these simple exercise, students understood the use and potentials of the branch structure in programming, because they had the opportunity to observe that the code they wrote behaved differently according to the map which was uploaded in their program. When students finished their exercises, they were separated in four groups and exported their code into Lego NXT Mindstorm which was executed using the designed field.

4.2.4 Lesson 5

The learning objective of the last lesson, was to enable students to analyze a simple problem of repeated structure, to formulate the algorithm in verbal form and program it utilizing a simulating software (Robomind 4.3 and Lego NXT Mindstorm – Experimental Group/Alice 3.1 - Control Group). Again in this lesson, the same tools were used (as in the previous lessons) and the same methodology was followed.
The basic activities for this lesson asked students to program (following the four phases of the Algorithmic Implementation) the virtual robot using a repeated structure (Figure 2, 3). Through these exercises students understood the use and potentials of the repeated structure in programming, because they had the opportunity to observe that they can program the virtual robot using much more less commands than in a sequential structure. When students finished their exercises, they were separated in four groups and exported their code into Lego NXT Mindstorm which was executed using the designed field. In the last ten minutes of the lesson, students answered the postest in order to check the degree of understanding of the teaching concepts.

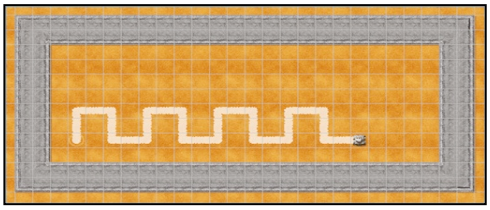

Figure 2: Repeated Structure Zik-Zak program - Robomind 4.3.

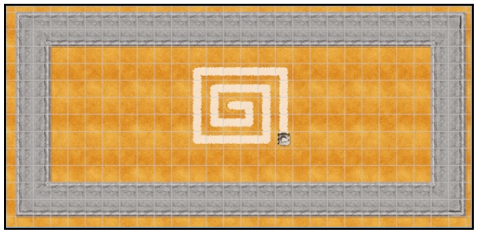

Figure 3: Repeated Structure Meandros program - Robomind 4.3.

Control Group

The basic technological tool utilized for the development of the Computational and Algorithmic Thinking in the control group was the software called Alice (Figure 4). Alice (v3.1), was used mainly to teach the three basic algorithmic structures (Sequential Structure, Branch Structure and Repeated Structure). The teaching methodology adopted in the control group was the same as with the experimental group and the exercises (Figures 5, 6, 7) were almost the same as with experimental group.

Figure 4: Simulation Program Alice 3.1

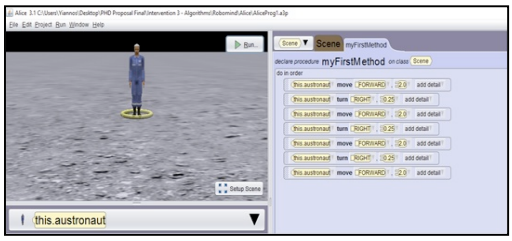

Figure 5: Sequential Structure exercise – Alice 3.1

Figure 6: Branchig Structure Exercise – Alice 3.1.

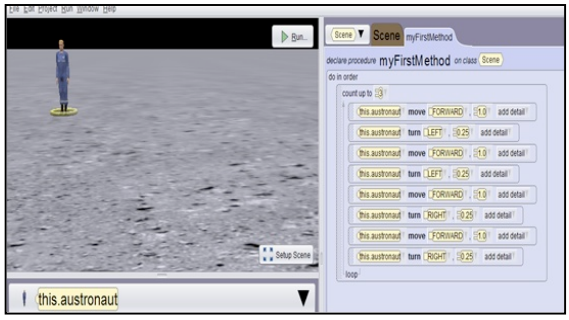

Figure 7: Repeated Structure Exercise – Alice 3.1.

4.3 Instruments

At the beginning of the first lesson a pretest was administered. The pretest was comprised of ten questions related to participants' conceptions about algorithms and computational thinking. Each correct answer to a question was awarded ten points. The same test was also administered as a postest at the end of the study. The duration of both the pretest and postest was ten minutes each. The formats of the questions were multiple choice, right/wrong choice and put the commands/instructions in the right order. Three questions examined the understanding of the concept of algorithms and the methods of representation (Verbal Description, Pseudocode and Logical Diagram). Two questions examined the understanding of the Sequential Structure and two questions the understanding of the Branch Structure. The last three questions examined the understanding and characteristics that designate an Algorithm and the steps for the development of a program.

4.4 Design Procedure

The principles of the theoretical frameworks of TPCK and TM were chosen to redesign these lessons. According to the topic selected, the learning objectives were defined relatively. A significant parameter that was taken into account in the design process was that the students had not been taught before anything on the development of algorithmic and computational thinking. According to the repeated procedure of the Technology Mapping, the first key action was to find and decide on an appropriate and effective technological tool exploited in the design of the teaching activities. During the last years, many software applications have been developed for teaching the development of computational and algorithmic thinking. The plethora of software was a parameter that made difficult the selection of suitable tools. For the selection of Robomind 4.3 various factors were taken into account such as, the ease of installation and use on computers. Robomind 4.3 gives the opportunity to novice programmers to learn the basic algorithmic structures and programming techniques and develop their computational thinking competencies. Furthermore, the fact that Robomind 4.3 can communicate and export the code to Lego NXT Mindstorm was an additional reason for its selection.

After selecting the basic technological tools, the appropriate teaching methodology was decided. An effective way for developing Algorithmic and Computational thinking is by practising in problem solving. Students worked individually, as well as in groups practising in problem solving, following the phases of the development of the Algorithmic Implementation. The learning activities were classified as having the same or similar difficulty in both groups (i.e. experimental and control).

5 Results

To check whether the instructional intervention facilitated students' development and understanding of Computational and Algorithmic Thinking competencies and concepts respectively, an analysis of covariance (ANCOVA) was conducted. For students' performance in postest the analysis showed that there was a statistically significant difference (p=.05) between Experimental Group (M=76.21, SD=18.11) and Control Group (M=69.41, SD=15.41). The analysis showed that the covariate wasn't statistically significant $F(1, 235) = 2007.08$, $P < 0.00$, $\eta^2 = 0.90$, showing that students with higher performance in the pretest didn't have higher performance in postest compared to the students with lower performance in the pretest. The results also showed that after the removal of the effect of the covariate "performance in pretest" into the dependent variable "performance in postest" the differences in "performance in postest" between groups were statistically significant, $F(1, 235) = 62.52$, $P < 0.00$, $\eta^2 = 0.21$ with the performance of Experimental Group to be higher than the Control Group.

6 Discussion

The results indicate that the theoretical framework of TPCK and the instructional design model of TM facilitated the design teaching for the development of Computational and Algorithmic Thinking of students. The development of Computational and Algorithmic Thinking is considered very important in the field of Computer Science in Education. Students face several difficulties in understanding algorithmic concepts and algorithmic structures (Ioannou & Angeli, 2013). The software Robomind 4.3 was selected due to its strong educational affordances and hence utilized teaching of novice programmers. Participant students in the Experimental Group had higher performance in postest from the students of the Control Group. Clearly, the significant results can be attributed to the intervention and to the instructional materials that included in Robomind 4.3 activities.

References

Angeli, C., & Valanides, N. (2005). Pre-service teachers as ICT designers: An instructional design model based on an expanded view of pedagogical content knowledge. *Journal of Computer-Assisted Learning*, *21*(4), 292-302.

Computer Science Teachers Association Task Force. (2011). *K–12 Computer Science Standards,* New York, ACM.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs.cmu.edu/~ CompThink/resources/TheLinkWing.pdf*.

Dagdilelis, V., Satratzemi, M., & Evangelidis, G. (2004). Introducing secondary education to algorithms and programming. *Education and Information Technologies*, *9*(2), 159–173.

Henderson, P. B., Cortina, T. J. & Wing, J. M. (2007). Computational thinking. *Proceedings of the 38th SIGCSE*

Ioannou, I., & Angeli, C. (2013). Teaching computer science in secondary education: A technological pedagogical content knowledge perspective. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 1-7). ACM.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, *108*(6), 1017-1054.

National Research Council. (2010). Report of a Workshop on the Scope and Nature of Computational Thinking. Retrieved May 15, 2016 from http://www.nap.edu/catalog.php?record_id=12840

Wing, J. (2007). *Computational Thinking*. Retrieved May 15, 2016 from http://www.cs.cmu.edu/afs/cs/usr/wing/www/Computational_Thinking.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.