The Impact of STEM-based Mobile Apps on learners’ Self-efficacy for Computational Thinking Dimensions

Sarantos Psycharis

1School of Pedagogical and Technological Education, ASPETE, Athens, Greece

Abstract—Research studies provide evidence for the effectiveness of the use of smart mobile devices and their educational applications for Science and Engineering learning. STEM content Education epistemology is related to the so called “Integrated Curriculum”. Computational STEM Pedagogy is a didactic model that applies the Computational Science experiment in alignment with the practices of Computational Thinking. A study was conducted on the pedagogical use of App Inventor for the conceptual understanding of cross-cutting ideas when the Computational Pedagogy STEM content education model is implemented during the instruction. A total of thirty (30) prospective Engineering educators from a Higher Education Institute in Athens-Greece participated in this research study. Students participated during a course for the “Pedagogical applications of Computers” and programmed using the App Inventor software/platform in order to solve real/authentic problems related to Science and Engineering. Students’ measured outcomes in the form of pre- and post tests showed a statistically significant increase for their self-efficacy for the Computational Thinking (CT) practices. Semi-structured interviews -at a preliminary level-also indicated the effectiveness of the functionality and design features of this software for learning crosscutting concepts.

Index terms—STEM, App Inventor, computational thinking, physical computing, mobile learning, self-efficacy

I. INTRODUCTION

Science and Engineering Education should focus not only on core/threshold concepts but also on the so called crosscutting concepts. These ideas/concepts should be taught through education scenario that includes the practices of Scientists and Engineers. For learners’ engagement in teaching and learning sequences, three major dimensions are proposed [1]. These dimensions are:

• Scientific and Engineering practices

• Crosscutting concepts that unify the study of Science and Engineering through their common application across fields

• Core ideas in four disciplinary areas: Physical Sciences; Life sciences; Earth and Space Sciences; and Engineering, Technology, and applications

Computational Pedagogy -as a model for teaching and learning-provides the medium to effective implement constructivist theories of learning where students can take and analyze data and create-through levels of abstraction-their models that will be simulated. The term “practices” is used instead of a term such as “skills” to emphasize that engaging in scientific investigation requires not only skill but also knowledge that is specific to each practice [1].

II. COMPUTATIONAL THINKING

Computational Thinking (CT) is defined as “the thought process 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” [2]. NGSS [1] connects CT with “computational practices” by considering that Science and Mathematics are becoming computational endeavors and suggests CT as a core scientific practice. This connection raises another issue about the provision of the theoretical background for what form it should take in school science and mathematics classrooms.

According to [3] Computational Thinking in Mathematics and Science can take the form of a taxonomy consisting of four main categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. In Education Literature there are many definitions about CT and there is no consensus about the set of core concepts/ dimensions and practices that should be included in CT.

We present the most common practices. These include: abstraction (problems need to be solved using abstraction levels and the inductive process as well as by applying mathematical reasoning and design-based thinking [4]. Modeling establishment of mathematical relations between selected variables), algorithmic thinking, automation, decomposition of a problem as a set of smaller problems, debugging, pattern recognition (for example patterns of data), metacognition and generalization [5] as well as design-based thinking.

In education literature CT is related to engineering design thinking and this connection is one of the pathways we need to follow for the implementation of the STEM content epistemology.
III. COMPUTATIONAL SCIENCE

Computational Science is a cognitive area that deals with the construction of models for solving real life problems using a specific methodology that combines Computer Science, Numerical Analysis and the specific scientific area where we need to develop a model in order to solve a problem[6-10]. One of the fundamental components of Computational Science is the process of abstraction of a phenomenon and the simulation of the model developed. Data received by the model will be compared with the data from the real world [6-10].

Computational Science can be integrated with many concepts of Computational Thinking. For example, reference [6-11] stated that “the ability to think computationally is essential to conceptual understanding in every field, through the processes of problem solving and algorithmic thinking”.

In Computational Science Education (CSE) the following spaces are included in order to implement the Computational experiment in Education (CE)[11].

1. “The hypotheses space, where the instructor guides the students to create their hypotheses according to prior knowledge and decide about the model that should be used. Misconceptions and cognitive conflicts should be discovered explained by the instructor during this phase.

2. The experimental space, which includes the method and the simulation of the model and crosscutting ideas/concepts are explored through the running of the model. Data are collected and analyzed in order to: find e.g. patterns in them, to verify a well known law in Science etc Programming is useful at this stage in order to simulate the model and control the variables. Decomposition and algorithmic thinking are fundamental processes as well as evaluation of data, all in alignment with the practices of CT.

3. The prediction space, where the results, solutions or conclusions formulated in the experimental space, are generalized in other situations and phenomena governed by the same law or mathematical relations. In this space metacognitive awareness finds its application”.

Computational Science Educations needs a pedagogical strategy to be implemented in Education and the Inquiry based teaching and learning approach is a suitable candidate. According to [12][13], Inquiry based teaching and learning approaches includes the following features: Question (learner engages in scientifically oriented questions), Evidence (learner gives priority to evidence), Analyze (learner analyses evidence), Explain (learner formulates explanations from evidence), Connect (learner connects explanations to scientific knowledge), Communicate (learner communicates and justifies explanations), Reflect (learner reflects on the inquiry process, respond to his/her work, develops metacognitive experience) and these are implemented through the following Inquiry tools: orienting and asking questions; generating hypotheses; planning; investigating; analyzing and interpreting; exploring and creating models; evaluating and concluding; communicating; and predicting.

In Table 1 the connection between the three spaces of the Computational Science Experiment with the dimensions of Computational Thinking and the essential features of Inquiry dimensions is presented[6][11].

IV. ENGINEERING PEDAGOGY- STEM EPistemology- Computational Pedagogy

The discipline of Engineering includes the Engineering content and the Engineering design [14]. “Engineering content arises from the integration of science, mathematics, embracing a collection of tools and practices, which engineers can use to design solutions to specific problems based on scientific laws and certain constraints” Engineering education is defined as “the development of knowledge of scientific ideas alongside with the acquisition of skills and attitudes”.

According to reference [15] “when students are engaged in the design process, they can integrate various skills and types of thinking—analytical and synthetic- and detailed understanding”. Computational Thinking practices are related to engineering design when this design is implemented in a context through for example the development of prototype model/pattern recognition, use of abstraction and the construction of design-based computational artifacts, e.g. through programming using optical languages and environments like the App Inventor software.

Engineering Pedagogy, is based on the integration of the engineering epistemology (justification of knowledge), the engineering design process and the application of proper instructional strategies that encourage student-centered learning of mathematical and scientific core and crosscutting concepts [6][11][15][16].

Engineering Pedagogy is a fundamental component of STEM content education. Integrated approaches of STEM Education are based on interdisciplinary or /and trans-disciplinary methods and have been suggested for applying integrated STEM as an holistic approach to the curriculum [10]. Such approaches are expected to enhance students’ capacity to “solve real-life problems by applying concepts that cut across disciplines” [16].

Integrated STEM content education is “an effort to combine some or all of the four disciplines of science, technology, engineering, and mathematics into one class, unit or lesson that is based on connections between the subjects and real world problems” [15].

STEM content approach follows the so-called transdisciplinary approach and it “focuses on the merging of the content fields into a single curricular activity or unit to highlight “big ideas” from multiple content areas” [16-20].

The term Computational Pedagogy was introduced in reference [21] as an extension of TPACK model, and was called Computational Pedagogical Content Knowledge. In the same reference, authors state that Computational Pedagogy is an inherent outcome of Computing, Mathematics, Science and technology integration and “Computing” is related to programming and the concepts of Computational Thinking.

In this article it is adopted the aforementioned model [21] with the addition of the Computational spaces [9][11] and the practices related to the practices of engineering design and
the (CT) practices as an integrated interdisciplinary STEM content Computational model (Fig. 1).

V. MOBILE LEARNING
In educational literature, apps are rapidly emerging as a new medium for providing educational framework for conceptual understanding in Science and Engineering [22][23][24]. Research suggests that the use of mobile learning enhances the engagement of students in commitment and productivity as well as to motivate students to learn “computing”. [24]. The use of App Inventor for introducing learners in Computational Thinking practices by engaging them to create apps for mobile phones is suggested by NSF[25] and its use for teachers’ training for event-based programming and leveraging the sensors and actuators of mobile devices in [26]. The concepts of Computational Thinking (CT) and the practice of programming are difficult to delineate in the literature because many CT studies consider the use of programming as their context [27][28].

Authors in [29] state that “computational thinking is an approach that does not necessarily need programming of computers, but rather is an approach to problem solving that uses strategies such as algorithms and abstraction”. Despite the fact that programming is not necessarily connected to Computational Thinking, it is considered programming supports engagement in CT practices [3][4]. Construction of artifacts in the STEM content Pedagogical model is an essential feature and programming through App Inventor can facilitate this construction [23].

VI. METHODOLOGY
The main purpose of this research study is to examine the impact of teaching using the Computational Pedagogy model by the implementation of App Inventor - on learners’ self-efficacy for Computational Thinking practices when crosscutting ideas are included in a STEM content course.
The perception of students on their own competence is defined as self-efficacy. Based on the discussions reported in literature, measuring self-efficacy is certainly necessary in order to develop new pedagogical methods to address the problems related to computer programming[30]. CPES questionnaire[30] for self-efficacy was used adjusting questions for C++ to questions relative to the use of App Inventor for programming while questionnaire for CT self-efficacy in [5] was used. Thirty prospective Electrical Engineering students participated in this research during the second year of their studies through a course for “Pedagogical Applications of Computers”. The objective of this research is to examine the impact of using optical programming with App Inventor on students’ self-efficacy for CT and programming when cross-cutting and Engineering concepts and are taught. Questions in CPES questionnaire were modified to be in alignment with the use of optical programming. For example, instead of the question “I can write syntactically correct C/C++ statements” we used “I can write syntactically correct App Inventor statements” and instead of “I can write logically correct blocks of code using C/C++”, we used “I can write logically correct blocks of code using App Inventor”. We present the interface of the application in Fig.2 and the code for the calculation of the total capacitance in series in Fig.3. Other examples used the above mentioned software for exploring crosscutting concepts, like the conservation of energy, the exponential decay, pattern recognition in data etc.

### TABLE 1. Model for the connection between Computational Thinking and Computational Experiment spaces

<table>
<thead>
<tr>
<th>Spaces of the Computational Experiment</th>
<th>Essential Features of Inquiry and Practices of CT in the Computational Experiment in Education</th>
<th>Inquiry tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypotheses space</td>
<td>Essential Features of Inquiry: Question, Dimensions of CT: Abstraction, Decomposition</td>
<td>Orienting and asking questions, Generating hypotheses</td>
</tr>
<tr>
<td>Experimental space</td>
<td>Essential Features of Inquiry: Evidence, Analyze, Explain, Creation or Remixing of Code, Dimensions of CT, Abstraction, Algorithmic thinking</td>
<td>Planning, Investigating, Analysis and interpretation, Modelling</td>
</tr>
<tr>
<td>Prediction Space</td>
<td>Essential Features of Inquiry: Connect, Communicate, Reflect, Dimensions of CT, Debugging, Generalization</td>
<td>Conclusion, Evaluation, Prediction</td>
</tr>
</tbody>
</table>

![Fig. 1. The Computational Science Experiment (CE experiment) – Computational Pedagogy](image)
after the instruction and Cronbach alpha internal consistency reliability was 0.79

VII. RESULTS

Results show a significant change in students’ responses for self-efficacy for Computational Thinking before and after the instruction as well as for self-efficacy in programming (CPES questionnaire). Results also indicate a significant relation between the responses between the self-efficacy for the two questionnaires, something which was expected (Pearson \( r=0.7 \)). Students were completed the self-efficacy in Computational Thinking concepts [5] which contains 23 questions related to the different practices of Computational Thinking. Examples of questions were:

- “When solving a problem I look how information can be collected, stored, and analyzed to help solve the problem” (practice of CT (collection, representation, and analysis of data)).
- “I can write a computer program which runs a step-by-step sequence of commands (practice of CT-Algorithms)).
- “When creating a computer program I run my program frequently to make sure it does what I want and fix any problems I find”(practice of CT (Testing and Debugging)).
- “When creating a computer program I break my program into multiple parts to carry out different actions”(concept of CT (Problem Decomposition)).

A paired t-test was applied for the 23 questions (\( p=0.002 \)), indicating a significant difference in self-efficacy for CT concepts (we consider the average score for all the questions, while the value was less that 0.05 in each question).

Similar results were received for the questions of CPES.

A paired t-test was applied for the 28 questions of CPES (\( p=0.003 \)), indicating a significant difference in self-efficacy for programming. The use of this questionnaire is justified by its content validity to measure Computer programming in relation to self-efficacy. The inclusion of the questionnaire for the Computer Programming Self-Efficacy is justified by the relation of programming with CT, computing and engineering design [31].

According to [31], “computing” includes “computation”. Our initiative to use the CPES questionnaire was justified by the fact that if CT practices are related to self-efficacy and programming through App Inventor is a “good medium” to implement correlation between the scores of the two questionnaires is expected.

VIII. CONCLUSIONS

Our results are in alignment with research findings that indicate that the use of App Inventor enhances students capacity for problem solving.

As stated in [32] “the positive statistically significant differences in problem-solving skills and self-efficacy
indicate that this study could be utilized as a basis for building a teaching-learning program using App Inventor and creating an educational plan for teaching computational thinking”.

The findings above are in agreement with those of [33][34] and indicate that students’ perceptions change using App Inventor as they feel as producers and not only consumers of digital technology. According to [34] we should focus on the means to implement computational thinking and “move beyond computational thinking as the goal of computing education to a perspective of computational action. Computational action emphasizes learners ability to develop computational artifacts. App Inventor can impact the movement to computational action through the development of computational artifacts in mobile phones which are connected to Science and Engineering education.

Interaction through buttons triggered students’ interest - according to a preliminary quantitative analysis- for exploration of algorithms(e.g. to find the equivalent capacitance) as well as to decompose the problem in simpler ones according to the practices of CT and the Computational Science epistemology.

The results of the current study indicate a strong effect of mobile learning in introductory computer concepts and crosscutting ideas for teaching.

Mobile computing can serve as a platform to reveal Computational Thinking practices when the Computational Pedagogy model is implemented. App inventor seems to be a good “medium” to implement the “computational action”, and the practices of Computational Thinking, i.e. to create a computational model using concepts from the computer science and test the model according to the engineering design cycle.

Before the instruction, students were presented with some ready examples from App inventor tutorials and they were taught about this type of optical programming, with focus on concepts they had met during their courses in Science and Engineering. Courses. They had already attended a course in C++, so they already knew the basic programming concepts (control structures, loop control etc.).

Some students expressed initially negativity towards App Inventor when they made errors but as the course ran for a second period they stated that they would use App Inventor as a programming tool.

Most of the students, after some presentations by the instructor-considered that the interface and the blocks of App Inventor were easily understood and operational

At the beginning of the course, during the hypotheses step of the computational experiment, they were asked to describe phenomena from their studies in Engineering that exhibit algorithmic thinking and if they know the concept of modeling as restriction of the theory. Another issue not related to computing but to the capacity to recognize the concept of modeling, is the difficulty of students to recognize which is the “system” and which is the surroundings of the system, so they could easily create buttons to deploy this interactivity between the system and its environment.

Despite the fact that CT is something beyond computer programming, our results indicate that for the self-efficacy structure, CT and computer programming are strong related.

During the course, we applied the Computational Pedagogy model as a theoretical framework for model development and collection and analysis of data for system designan engineering design process).

During the intervention, a series of didactic scenario was presented. These scenarios were based on the constructivism theories that suggest the developing of artifacts in a context, i.e. to solve a problem.

At the last space of the computational experiment( the generalization/metacognitive phase) students started to think in a more abstract ways in order to find common factors that govern a phenomenon and for example, they should apply the same techniques of App Inventor for the connection of resistors or springs.

This work does not deal with simple “tutorial” application like the development of apps that control a screen, or the transformation of text to speech etc. Instead we wanted to develop apps that are related to the curriculum of prospective engineering educators.

The objectives of future research include the development of Computational Pedagogy teaching and learning scenario with activities for improving students’ self-efficacy in Computational Thinking and programming with focus on more cross-cutting ideas.

REFERENCES


[6] Author


[9] Author


The International Conference on E-Learning in the Workplace 2020, www.icelw.org


[20] T. J. Moore, “STEM integration: Crossing disciplinary borders to promote learning and engagement”. Invited presentation to the faculty and graduate students of the UTeachEngineering, UTeachNatural Sciences, and STEM Education program area at University of Texas at Austin, December 15, 2008.


direc-tions in computer science education summit meeting, Orlando, FL., 2014


AUTHORS

Sarantos Psycharis is with the School of Pedagogical and Technological Education, ASPETE, Athens, Greece (e-mail: spsyscharis@gmail.com).

Manuscript received 14 April 2020. Published as submitted by the author(s).