Monitoring Students’ Perceptions in an App Inventor School Course

Giorgos Panselinas, Efi Fragkoulaki, Nikolaos Angelidakis, Stavros Papadakis, Eleytherios Tzagkarakis, and Vassileios Manassakis

Abstract—MIT App Inventor is a block-based programming environment for children and teenagers that sets a “low floor” for allowing creative app building while engaging students in complex computational thinking activities. The present study aims at (a) monitoring students’ perceptions on ‘popularity’ and ‘perceived difficulty’ of certain activities/lessons through the implementation of an App Inventor course in a Greek lower high school, (b) detecting any course design or activity/lesson plan and implementation factors that affected students’ perceptions and finally (c) evaluating their experience with App Inventor in contrast with their previous experience with MicroWorlds Pro and Scratch. Our study confirms students’ positive perceptions such as positive task value beliefs and self-efficacy, identifies features of successful “resources learning” in competence-based learning and finally offers a students’ comparison between App Inventor, MicroWorlds Pro and Scratch.

Index Terms—MIT App Inventor; Block-Based Programming; Students’ Perceptions; School Course

I. INTRODUCTION

According to Gartner, Inc., there will be nearly 20.8 billion devices on the Internet of things by 2020. Ubiquitous computing is here to stay [1]. Our world is not the world of Newton not even the world of Einstein. If Math and Science is considered essential for all children to understand how our world works, now in 21st century it is also essential for children to understand our digital world [2]. It is essential to understand computer science concepts and acquire computing competences so as not to be just consumers of digital products and services but also active participants in their future creation. Moreover, it is estimated that by 2020 90% of jobs in Europe will require high level digital skills. Commissioner Vassiliou argues that “It’s not enough to understand how to use an app or program; we need youngsters who can create their own programs” [3].

Despite this need, there may be a lack of up to 500,000 Information and Communication Technologies (ICT) professionals in 2020. Although supply of computing/IT skills is growing in Europe due to Vocational Education and Training, Higher Education is stagnating [4]. Globally, countries around the world are beginning to act on the rationale for introducing computing education as early as middle school in order to train young minds in this discipline and way of thinking. Greece, having the privilege to have more than 4000 computer science teachers holding a Computer Science (CS) degree, is one of them. Yet, in middle schools, students learn programming/coding along with simple ICT skills such as word processors etc. spending just one hour per week [5].

II. CHILDREN AND PROGRAMMING

A. Programming Environments for children

In an attempt to increase interest in CS, much effort has gone into developing tools and activities for young people [6].

The idea of “low floor, high ceiling”, as one of the guiding principles for the creation of programming environments for children, has been around since the days of Logo. It essentially means that though it should be easy for a beginner to cross the threshold to create working programs (low floor), the tool should also be powerful and extensive enough to satisfy the needs of advanced programmers (high ceiling) [2].

Yet, MicroWorlds Pro, a logo programming environment still used in a lot of Greek middle schools, though powerful, it uses text based instructions. Glezou & Griogriadou [7] assume that the Logo programming syntax could be a significant barrier for some students who get caught up in the details of the programming syntax. That could be the main reason that Scratch is replacing Logo in Greek schools too. As Berry [8] argues “…building programs, rather than writing them, means learners can focus on expressing ideas as code rather than mastering formal syntax.”

However, MIT App Inventor is also such a tool. It is a visual programming environment that uses Scratch-like graphical blocks of code for building Android mobile apps. It sets a “low floor” for allowing creative app building (something all teens are eager to do) while still engaging in complex computational thinking concepts including procedural and data abstraction, iterative and recursive thinking, structured task breakdown, conditional and logical thinking, and debugging [2].

B. Children’s Learning and Perceptions as far as block-based programming environments vs text-based environments is concerned

Weintrop [9] found that a majority of students (92%) viewed block-based programming as easier than text-based
programming. In the aforementioned study but also in Weintrop and Wilensky study in a high school [10], Weintrop reports on students citing a numerous factors that contribute to making block-based programming easy, including (a) the natural language description of blocks, (b) the drag-and-drop composition interaction, (c) the ease of browsing the language and (d) the lack of needing to memorize commands. Students also identify drawbacks to block-based programming compared to the conventional text-based approach, including (a) a perceived lack of authenticity, (b) a difficulty of building large complex programs with block-based tools and (c) being less powerful.

In another study, in novices in programming middle school students, Price & Barnes [11] found that while the interface did not seem to affect students’ attitudes or perceived difficulty, students using the block interface spent less time off-task and completed more of the activity’s goals in less time.

The aforementioned findings provide evidence that reinforces the intuition that the block-based modality is easier for novices and gives us insight into what features are seen as the sources of this ease-of-use through the eyes of a learner [9].

C. Children’s Learning and Perceptions on MIT App Inventor

As Utting and Roy and their colleagues argue [12], App Inventor and Scratch strive to engage the novice programming students by enabling them to act on projects, which hold more real life context in contrast to more conventional programming. Both of them are attractive programming environments and combine richness with simplicity.

However, MIT App Inventor appears to be a more popular programming environment for use with learners in the teenage group than Scratch because it fascinates them as they can create personalized software for their most precious device, the smartphone/tablet, and literally augment their own reality [12]. Such projects are not just about programming but engage the students with the world outside of the computer lab. The app’s effect is not enclosed within a virtual world, it has a meaning in the real one [13].

In another study, Nikou & Economides[14], studying both Scratch and App Inventor in a Greek senior high school, detected rise of motivation to engage in programming from intrinsic rewards (challenge, curiosity, mastery) for both groups. Yet, intrinsic goal orientation was stronger in App Inventor. In the same study, researchers detected stronger task value beliefs for the App Inventor group. Task value belief refers to the students’ perception about the material of the course in terms of interest, importance and usefulness. Finally, Nikou & Economides [14] detected an increase in self-efficacy for both groups. Self-efficacy refers to students’ confidence that they are able to perform well in the class.

III. FURTHER RATIONALE AND OBJECTIVES OF PRESENT STUDY

But what about course design? Resnick [15] underlines the importance of project-based learning and creative design process when children learn with Scratch. However, most recent research in computing education in K-12 has focused less on process, and more on tools. Thus, despite the flurry of recent research activity, many key questions still remain unanswered, and there is much that needs to be done on how to design the teaching and learning experience [2].

Moreover, students’ positive perceptions such as positive task value beliefs and self-efficacy are true promoters of students’ learning [16].

Meanwhile, the studies in the previous section that detect students’ perception on their learning programming/coding use a pre-test/post-test design, not monitoring the possible transition in perception depending on lessons designed. Thus, those studies cannot detect course design and implementation factors that affect students’ perceptions.

The present study aims at (a) monitoring students’ perceptions on ‘popularity’ and ‘perceived difficulty’ of certain activities/lessons through the implementation of an App Inventor course in a Greek lower high school, (b) detecting any course design or activity/lesson plan and implementation factors that affected students’ perceptions and finally (c) evaluating their experience with App Inventor in contrast with their previous experience with MicroWorlds Pro and Scratch.

IV. METHOD

A. Course Design and Participants

Course design was based on “integration pedagogy” that put first “resources learning” activities and then the implementation of projects emerged from students’ ideas. “Resources learning” activities are learning activities that provide the students with the knowledge, the skills for action and the life-skills which students mobilize to cope with problem situations [17]. Yet, students gave feedback to evaluate only the lesson plans and activities predesigned for them because that was the first goal of the monitoring process. Course design, lesson plans, educational resources (e.g. worksheets) and students’ monitoring tool were produced for the project, funded with the support of the European Commission, “Junior Code Academy – Enhancing coding skills in European Schools”.

App Inventor activities/lessons started in the first week of February 2017 in all classes of the 9th Grade (6 classes, 135 students) of a junior high school in Greece. The students were divided into groups of 3-4 persons. Activities/Lessons lasted 40 minutes per week. The students had already taken 10 lessons about algorithms and programming with Logo using MicroWorlds Pro. Some of them had a short period experience with Scratch when in Primary school. The teacher is also one of the researcher and author of the present paper.

1) Resources Learning Activities:

1st Activity/Lesson plan 1 (February 6-10): It concerned the basic concepts (algorithm, program, programming languages, etc.). There has also been a presentation video and children finished an activity in Code.org site: (Flappy bird) https://studio.code.org/flappy/1

2nd Activity/Lesson plan 2 (February 13-17): Children participated in activities from the Code.org site (Minecraft) https://studio.code.org/s/mc/stage/1/puzzle/1, and, later, they
were introduced to the notion of computational thinking. The activity concluded with students filling a computational thinking worksheet including unplugged activities.

3rd Activity/Lesson plan 3 (February 20-24): Introduction to App Inventor interface - features, examples, tutorials. The groups of students created accounts and interacted with the programming environment for the first time.

4th Activity/Lesson plan 4 (February 28-March 03): It concerned the building of the first application which was created by the students in a teacher guided step by step process. The students tested their first application on the school tablet.

5th Activity/Lesson plan 5 (March 06-March 10): Students created an app with the assistance of the activity worksheet. The app begins playback of music, plays a sound when the screen is tapped and makes the mobile phone vibrate.

6th Activity/Lesson plan 6 (March 13-March 17): Students, guided by the activity worksheet, created two applications: (a) School tablet plays what you type in a text box and (b) Minion answers your questions.

7th Activity/Lesson plan 7 (March 20-March 24): Students created an application using sensors and events, once again with the assistance of the activity worksheet.

2) Project-based Activities:
The remaining five (5) lessons engaged students in developing their own applications, working in groups and presenting them to the whole class.

B. Data Collection
At the end of each lesson (resources learning activities) students filled the lesson monitoring questionnaire:
The student monitoring questionnaire posed the following questions: (1) In a scale from 1 to 5, how much did you like JCA lesson today? (2) In a scale from 1 to 5, did you find the lesson activities difficult? (3) In a scale from 1 to 5, were you able to finish all the activities purposed for this lesson? (4) In a scale from 1 to 5, how do you globally evaluate today's lesson?

A focus group interview was conducted after the "Resources Learning Activities" in order (a) for the students to evaluate their experience with App Inventor in contrast with their experience with MicroWorlds Pro or even Scratch and (b) for the researchers to have a deeper insight to all individual answers in questionnaires.

C. Method of Analysis
In order to achieve the goals of the study:
1) Answers provided to the following research questions:
   As far as ‘Lesson Popularity’ is concerned, students answer the following questions in questionnaires, whereas “why” is discussed in focus group and is based also on teacher’s observation.
   i. In a scale from 1 to 5, how much did you like JCA lesson today?
   ii. In a scale from 1 to 5, how do you globally evaluate today’s lesson?

   As far as ‘Lesson perceived difficulty’ is concerned, students answer the following questions in questionnaires whereas “why” is discussed in focus group and is based also on teacher’s observation.
   iii. In a scale from 1 to 5, did you find the lesson activities difficult?
   iv. In a scale from 1 to 5, were you able to finish all the activities purposed for this lesson?

2) We compare the aforementioned answers in questionnaires as they change through the seven (7) activities/lessons to detect course design and implementation factors that possibly affect students’ perceptions. We also use teacher observation of lesson and students’ behavior to come up with some hypotheses.

3) We ask students in focus group to compare their experience with App Inventor with their experience with MicroWorlds Pro and Scratch.

Thus, we use triangulation of data (students’ questionnaires answers, teacher-researcher observation data and focus group data) to validate and discuss the results [18].

V. RESULTS
As far as ‘Lesson popularity’ and ‘Lesson perceived difficulty’ is concerned we present students’ answers ‘1’ and ‘2’ together and also ‘4’ and ‘5’ together. We have grouped positive (1+2) and negative (4+5) answers the way that enables for a more focused review of the results.

A. Lesson Popularity
In a scale from 1 to 5, how much did you like JCA lesson today?

In a scale from 1 to 5, how do you globally evaluate today’s lesson?
Generally, students liked the activities with App Inventor and they have given them a very good grade as the Fig. 1 and 2 present. Monitoring students answers through the implementation of the course we can see that Lesson Plan 1 and Lesson Plan 5 were the most popular and high evaluated whereas 3 & 4 were less popular but also 2 & 6 were evaluated less than the other lessons.

B. Lesson Perceived Difficulty

In a scale from 1 to 5, did you find the lesson activities difficult?

In a scale from 1 to 5, were you able to finish all the activities purposed for this lesson?

Generally, most of the students did not find the activities difficult and finished all of them in due time (Fig. 3 and 4). However, about 20% of the students estimated lesson plan 3 and lesson plan 5 more ‘difficult’, with a slightly higher percentage than the other lessons. Furthermore, in lesson plan 3 & 4 we had the greater percentage of students that did not finish all the activities purposed (Fig. 4).

C. Hypotheses on course design and implementation factors that possibly affected students’ perceptions

The first hypothesis we can make is that activities the kind of Code.org ‘Flappy bird’ are very popular to students. Students perceive them as easy and evaluate them with a very good grade.

The second hypothesis we can make is analyzing the case of activities of the lesson plan 5 through teacher-researcher’s observation. In Lesson plan 5, according to teacher-researcher the students developed autonomy. Although some students found the activities difficult (Fig. 3) they didn’t need the support of the teacher and finally finished the activities (Fig. 4). We can therefore assume that in Lesson plan 5, students provided some evidence of “scaffolded” groupwork learning in the ‘hand-over’ stage [19]. Furthermore, 76% of the students evaluated the activities of lesson plan 5 with a high grade (Fig. 4). Thus, we can assume that the activities in Lesson plan 5 were adjusted to students’ capacities and interest, and put in the right place in the course design and implementation after the “resources learning” activities of lesson plans 3 and 4 where teacher’s support was needed.

The third hypothesis that we can make is that the activities of the lesson plans 3&4 were not as successful as other lesson activities. Just half of the students liked the activities and more than 20% of them did not manage to finish them all. We cannot tell if the fact that they did not finish them made them respond that they did not like them or the reverse is true.

D. Students’ perceptions comparing App Inventor, MicroWorld Pro and Scratch

When students were asked in focus groups which programming environment they prefer and why, they agreed that they definitely prefer App Inventor.

The reasons they cited are that App Inventor is a) easy, b) more pleasant, c) it has a more attractive environment, d) you don’t have to memorize the list of the commands, e) you make more interesting apps, f) you can easily build an app that runs at your phone, g) you don’t have to remember how to write a command, h) it is not as boring as Logo, i) we had worksheets that helped us build the apps without the help of the teacher. They also liked that they worked in groups. In contrast with Scratch, students told the teacher that Scratch is not so interesting, it is for younger kids and App Inventor has friendlier human-computer interface.

The teacher confirms that even the students that were not interested in previous taught computer science lessons (theoretical curricula and MicroWorlds Pro based exercises) engaged with great enthusiasm in App Inventor activities. They asked the teacher to assist in providing opportunities to build apps at home and they also displayed an emerging interest in developing their own ideas.
Our study confirms the popularity of MIT App Inventor (MAI) activities in teenage group [13]. It also confirms that MAI indeed sets a “low floor” for 9th graders as students generally estimated it as easy [2].

Our observation alongside with students’ answers in questionnaires and focus group confirms an increase in motivation, self-efficacy and task value beliefs [14]. Students found it much easier than Logo’s text-based MicroWorlds Pro [9] but also easier and “with a friendlier environment” than Scratch, a statement on behalf of the students that we did not found in another study. MAI is said to be more interesting than Scratch and students asked for more projects maybe because in MAI “you can easily build an app that runs at your phone” students’ most precious device [13]. The aforementioned statements and the following: “Scratch is for younger kids”, “with MAI you make more interesting apps” perhaps confirm stronger task value beliefs in MAI activities than Scratch activities in terms of interest, importance and usefulness [14]. Furthermore, the reasons why students preferred MAI than a Logo text-based programming environment were a) easier and friendlier environment, b) the lack of needing to memorize the commands and c) stronger task value beliefs [9],[10]. It is also interesting that the drawback of ‘perceived lack of authenticity’ students identified for block-based programming in the previous studies [9], [10] does not apply in MAI as our study presents.

Moreover, what are the insights that our study offers as far as designing the teaching and learning experience? Actually project-based learning and creative design process is also the goal of working with MAI as it is in Scratch [15]. The last 5 lessons of the course design of the study were designed to be project-based, and also students were interested and proposed building their own apps based on their own ideas. However, worksheets that guided students in “resources learning” activities were evaluated positively by students and did not hinder their autonomy as we observe in Lesson plan 5. Yet, observing the transition of students’ perceptions on lesson 5, lesson 6 and lesson 7 we believe that project-based learning had to be initiated earlier, immediately after lesson 5. Teacher and worksheets scaffold groupwork learning to students’ autonomy or ‘hand-over’ stage [19]. When autonomy is accomplished, it is time for project-based learning and creative design process, where students develop their own ideas and build their own apps [15].

Students’ positive perceptions are true promoters of students’ learning [16] so it is important to identify them. However, our study draws some limitations as it is a case study of an implementation to one school. Furthermore, although it uses triangulation of data to validate results, it doesn’t use predictive statistical analysis to identify statistical significant correlations. For all these reasons, the discussion is carried out upon hypotheses and not upon conclusions that can be generalized.

ACKNOWLEDGEMENTS

With the support of the European Commission, EU Erasmus+ Programme, under the action type Strategic Partnerships for School Education: “Junior Code Academy – Enhancing Coding Skills in European Schools” (2015-1-PT01-KA201-013121), coordinated by Code for All, Lda.

REFERENCES


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
Nikolaos Angelidakis was born in Halkidiki, Greece, in 1972. He received a BSc degree in Computer Science in 1994, and a MSc degree in Computer Science (E-commerce and Information Systems) in 2008, both from the Computer Science Department of the University of Crete.

He teaches Informatics in Secondary Education. He is also Deputy Head Teacher of 5th Junior High School of Heraklion. He has written a book in Greek concerning introduction to computer programming with Python (2015). He is co-author of a school book concerning IT Applications (2014).

Stavros Papadakis was born in Crete in 1973. He received his B.Sc. in Computer Science from the Computer Science Department, University of Crete in 1996. He served in the Greek Army in 2000-2002 as a reserve officer. He worked as a SOFTWARE ENGINEER for the Computing Center of the University of Crete and as a contract DEVELOPER with emphasis on Web Applications Backend programming and Databases. He also worked as a TEACHER of Computing and ICT in various schools and vocational training centers.


Mr Tzagkarakis was awarded with the silver medal at the "Education Business Awards 2017 - Recognizing Best Practices in Managing Educational Institutions" at the category "Life-Long Learning – Public School"

Vassilios Manassakis holds a MSc in Computer Networks (1997), Computer Science Department of University of Edinburgh, and a B.Sc. in Computer Science (1995), Computer Science Department of University of Crete. He is currently a Promoter of European Projects of Secondary Education in the Regional Directorate of Education of Crete, monitoring already existing European projects and providing assistance to schools wishing to participate to European projects. He has previous working experience as an ICT teacher at the Ministry of Education and also as a Network Technician and Designer in various private companies.

He has participated in various European projects, such as: 1) "Cultural Mosaic of Europe – Past, Present and Future", Comenius School Partnerships 2) "Improving The Quality Vocational Learning Activities", Leonardo Da Vinci 3) "The use of information and communication technology (ICT) in tourism and hospitality management", Leonardo Da Vinci - IVT 4) "Transitions from childhood to youth", Comenius 5) "Training Adult Trainers: Train the Trainer", Leonardo Da Vinci – VETPRO 6) "Teaching Rubbish", Youth In Action, Turkey 7) "Social Media for Democracy “, Youth In Action, Georgia 8) "Environmental Tourism – Alternative Ways", Comenius-Regio 9) "Building Rapport to prevent early school leaving", centralised Comenius Multilateral Project 10) "Continuing Professional Development: a sustainable and effective tool for employability", Study Visit, Madritt-Spain 11) "Raising the Bar ”, Training Course, Valletta – Malta 12) "Work Based Learning in VET in Europe: Exchanging ideas, developing projects", Berlin – German

DOI: [http://dx.doi.org/10.24018/ejers.2018.0.CIE.633](http://dx.doi.org/10.24018/ejers.2018.0.CIE.633)