PROGRAMMING EDUCATION WITH A BLOCKS-BASED VISUAL LANGUAGE FOR MOBILE APPLICATION DEVELOPMENT

Can Mihci* and Assoc. Prof. Nesrin Ozdener**
*Marmara University Institute of Educational Sciences
**Marmara University Ataturk Faculty of Education Department of CEIT

ABSTRACT

The aim of this study is to assess the impact upon academic success of the use of a reference block-based visual programming tool, namely the MIT App Inventor for Android, as an educational instrument for teaching object-oriented GUI-application development (CS2) concepts to students; who have previously completed a fundamental programming course that involved education of structured programming concepts using C# (CS1). It has also been studied whether impacts upon CS2 success of factors such as previous success in CS1 and prior high school experience in programming, have a relationship with the impact upon CS2 success from the use of the blocks-based instructional tool. The research; which has a post-test only quasi-experimental design; has a sample comprised of 101 Undergraduate students who are taking up the CS2 course at a Department of Computer Education and Instructional Technologies (CEIT).

KEYWORDS

Programming education, app inventor, programming paradigms, mobile application development, programming blocks

1. INTRODUCTION

Although programming education is vital to the development of many industries and a highly popular subject in many curriculums of today's academic institutions, it is a known fact that it poses great difficulties to novice students who most often fail to achieve desired results at introductory levels of it (Kinnunen and Malmi,2008;Özdener,2008; Mccracken et al., 2001; Soloway, Ehrlich, Bonar and Greenspan 1983). The fact that average academic success in worldwide programming education is below desired levels, shows us that programming education has its own exclusive challenges. According to Robins, Rountree and Rountree (2003), students need to possess high-levels of cognitive skills for overcoming the said challenges and also, lecturers need to use appropriate teaching strategies.

Papert, who has extensively studied on how to teach programming more effectively; has created the Constructionist learning theory, which is based on Piaget's idea of "learners intrinsically constructing knowledge" and which suggests that the most effective method for learning is through the construction of personally meaningful concrete artifacts with a social aspect to them (Papert and Harel 1991). This theory has led to creation of the blocks-based visual programming languages to be used in programming education in order to make things simpler and more personal for students; although the initial idea was to be employ these for offering easier-to-use tools to end-user developers (Mohamad, Patel et al. 2011).

As previously mentioned, although block-based visual programming languages are thought to designed for the benefit of end-user developers (Mohamad, Patel et al. 2011), several studies also propose similar approaches, for example, with physical real-world blocks, as programming education tools for small children (Wyeth and Purchase 2000). Other studies have discussed the effect of education with tangible blocks increasing interest and success in programming in children (McNerney 2004, Horn and Jacob 2006), and although positive results with the use of block-based tools in programming education have occasionally been reported (Horn and Jacob 2007, Wang, Zhang et al. 2011), it is thought that there is a need for further experimental studies that display concrete results.

Another approach that has seen recent use for increasing the motivation of students towards programming education is the use of smart mobile devices in programming education. Smart mobile devices are becoming vastly popular; in that, as of the year 2011, there is at least one software application in the mobile phone of every other American citizen (Purcell 2011). Considering the high usage ratios and the popularity among the young generation of smart mobile devices, it can be expected that educational applications that target these platforms should gain interest from students (Rau, Gao et al. 2008).

According to Dvorak and Buchanan, students allocate more time to study for lessons that are taught with new tools (Dvorak and Buchanan 2002). And perhaps for this reason, Mahmoud and Dyer have proposed in their teaching model -which has been fashioned in parallel with their study conducted at Guelph University- to give programming education to students by using BlackBerry branded smartphone devices (Mahmoud and Dyer 2007). However, there is no experimental data that shows how using this smart device as an educational tool for teaching programming would affect student success. Tillmann et al. Have also proposed a system in which students would get all the programming education entirely over smart mobile devices, and although they have not tested the use of this system, they have claimed that the future of programming education lies in "mobile programming" (Tillmann, Moskal et al. 2012). Again, in this context, Kurkovsky and Bhagi have each run separate studies that aimed to teach students programming through game development for mobile platforms and they have both emphasized the positive student motivation that has been observed. However, both studies have not made it clear the level of success achieved by students in programming education, especially as compared to conventional syntax-based approaches (Kurkovsky 2009, Bhagi 2012). Another unanswered question is whether the use of mobile devices would affect all students the same way. All in all, a review of current literature shows that, although use of smart mobile devices as educational tools for increasing student motivation in programming classes has already been discussed, there is need for further experimental research in this matter.

A project developed by Google Inc. for enabling end-users to develop their own apps on the Android mobile operating system (although later discontinued and adopted by the MIT Media Lab), the App Inventor for Android (AIA) is a blocks-based visual programming language (BBVPL) that has achieved popularity in the recent years. With its potentially promising source for student motivation by enabling development of personally meaningful mobile apps on student-owned devices; and an easy to understand blocks-based visual programming approach, App Inventor soon made researchers think that it could prove to be a valuable tool in programming education. As such, it has been proposed that this blocks-based app development tool for mobile platforms could be used in programming education by Karakus and his colleagues (Karakus, Uludag et al. 2012) and it has been also claimed by Wolber that it is a useful tool for increasing student motivation towards programming classes (Wolber 2011). It could therefore be beneficial to determine experimentally whether the use of MIT App Inventor for Android would have a positive impact upon student success in programming education. Particularly, this study proposes the use of MIT App Inventor for Android as a transitional tool from the console based procedural approach stage in programming education to the object-oriented GUI-application development stage. As opposed to most previous research with App Inventor, all the participants in the study have prior experience in programming both at a CS1 course where they received the console based procedural programming education, with some of the participants having even more, pre-university programming experience whom they mostly received at vocational high schools.

2. PURPOSE

The aim of this study is to investigate the effects of using a blocks-based visual programming language for app development in mobile platforms as an educational tool for teaching programming at the object oriented GUI-application development (will be referred to as CS2 from now on) phase upon the academic success of undergraduate students. Additionally, it has been investigated whether there is an interaction between effects from factors that influence object oriented GUI-application development related academic success, such as previous success level at a prior university programming course that involves fundamental education in structured programming concepts (will be referred to as CS1 from now on) and high-school experience in programming prior to university education (if any) upon student success; and the effect from the tool that has been used. In this context, the following hypotheses have been tested:

- The average post-test results of the experimental group, who used the Blocks-Based Visual Programming Language (will be referred to as BBVPL from now on) will be significantly greater than that of the control group who used a conventional syntax-based object-oriented programming language (will be referred to as SBL from now on) for desktop application development (**Hypothesis 1**).
- A factorial analysis of variance shall reveal a significant three-way interaction between the effects of the following factors upon academic success of students in object oriented GUI-application development: the factor of educational tool employed (BBVPL / SBL), the factor of high school experience in programming education, (experienced/inexperienced) and the factor of previous success at the CS1 university course (successful/unsuccessful) (**Hypothesis 2**).
- A comparison of average academic success between the so-called "novice" sub-groups, which have been determined as students who lack high-school programming experience and who have been previously unsuccessful at CS1 course, shall reveal a significant difference in favor of the sub-group that has used the BBVPL against the sub-group that used SBL (**Hypothesis 3**).

3. METHODOLOGY

3.1 Research Design and Sample

The study follows a posttest only quasi experimental design. Pre-tests have been used however, to control for groups equivalency. The study sample is comprised of 101 2nd year undergraduate students at the Department of Computer Education and Instructional Technology, who have taken up at the previous semester the CS1 course, which involves teaching programming basics through the structured /procedural programming paradigm and by the development of console applications for the desktop PC using .NET (C#) language. The students belong to two groups, namely the daytime education and evening education; which have been randomly assigned as the experimental and control groups for the research. The software application development projects undertaken throughout the CS2 course have been taught by using a reference Blocks-Based Visual Programming Language for Mobile Application Development (BBVPL), namely the MIT App Inventor for Android, in the experimental group; whereas projects in the control group have been developed as Windows-Form applications on Microsoft Visual Studio 2010, using .NET (C#). The course was continued for 5 weeks and it was ensured that projects, learning outputs and weekly course content for both groups are identical, with the only difference in both groups being the educational tool (the programming language) used.

3.2 Data Collection Instruments

Pretest 1: Developed by Osman Ay (Ay, 2011), this is a test that measures knowledge of console-based procedural programming concepts in the C# language, which has an item internal consistency coefficient (Cronbach's Alpha) of 0,67. Using this test, the knowledge of students in the structured/procedural programming concepts, which they have learned throughout the CS1 course, has been measured and it was made sure that the experimental and control groups are equivalent in this sense.
Prior Programming Experience Form: This form was used in order to gather data from students pertaining to the type of high schools they have graduated from and the context of programming education they have received in high school (if any).
Pretest 2: Developed by the researchers, this test measures knowledge in Object-Oriented GUI Application Development concepts of programming education and has been used to verify the consistency of student declarations in "Prior Programming Experience Form". The test, which contains 13 items; has an item internal consistency coefficient (Cronbach's Alpha) of 0,94. As a result of this test, students have been allocated to "Prior experience in GUI-application development" nominal factor groups as experienced/inexperienced.
Post-test: This test has been used to at the end of the research application, for the purpose of comparing academic success of students in experimental and control groups in regards to education on object-oriented GUI-application development. The test, which consists of 20 items, has been implemented as an applied

examination in a computer laboratory. Students in the experimental and control groups have been asked to use relevant tools, with which they have received the CS2 education (MIT App Inventor for the experimental group and Microsoft Visual Studio 2010 for the control group), for answering the test questions. Cronbach's Alpha value for the test, which was developed by the researchers, has been found to be 0,865.

4. FINDINGS

4.1 Testing for Equivalency in Groups

In order to determine whether study groups are equivalent in terms of resuls from Pretest 1 and Pretest 2; it was initially made sure that results from all sub-groups for each test are normally distributed. Following this, an independent samples T-test (p=0,900) for equivalency in Pretest 1, where results are normally distributed; and a Mann-Whitney U test (p = 0,737) for equivalency in Pretest 2, where results are not normally distributed; have been carried out. Both tests have shown that there is no significant difference between experimental and control groups in terms of Pretest 1 and Pretest 2 scores.

4.2 CS2 Success of Groups Depending on the Instructional Tool Used (Hypothesis 1)

The post-test scores of the experimental and control groups have been compared with an independent samples T-test and the results have been given in Table 1. According to statistical test results, average scores in the post-test, which measures object-oriented GUI-application development skills, are significantly different in the experimental and control groups ($t(74)=2,201$, $p<0,05$), with the control group having the higher average. Hypothesis 1 is therefore rejected.

Table 1. T-test results showing difference in score averages between groups depending on the instructional tool used. (Experimental group: MIT App Inventor; Control group: Visual Studio 2010)

Group	N	X	S	df	t	P
Control	34	47,50	25,71	74	2,201	.031
Experimental	42	36,66	17,02			

4.3 Creation of Factor Groups

Students who fall below average in Pretest 1 scores at a range of standard deviation multiplied by 0.3 have been grouped as "unsuccessful", whereas students who fall above average in Pretest 1 scores at a range of standard deviation multiplied by 0.3 have been grouped as "successful" to determine the factor groups for CS1 success

Prior programming experience that students have earned during high-school education has been uncovered with the "prior programming experience form". According to data acquired through this form, it was found that not only did some of the students in the research sample have had programming experience in high school, but also a portion of this experience involved education in object-oriented GUI-application development. Therefore, Pretest 2 has been administered to students in both experimental and control groups to measure knowledge in object-oriented GUI-application development concepts. Students who were below average were allocated into the "inexperienced" and students above average wer allocated into the "experienced" groups for the Factor of Prior High School Education in GUI-App Development. In order to ensure the consistency of this allocation to groups, a chi-square test has been carried out between this nominal grouping and the declarations of students at the prior programming experience form on receiving GUI Application Development course during high school. The chi-square test yielded a result (sd=1, p=0.00) showing the grouping was consistent with student declarations. In this context, the data in Table 2 pertaining to programming education at high school have been gathered.

Table 2. Students' prior programming education from high school

		Type of High School (Received Programming Edu. ?)		Received GUI App Dev. Education?		Was successful in Pretest 2?	
Group	**Total Students**	**Vocational (Yes)**	**Regular (No)**	**Yes**	**No**	**Yes**	**No**
Control	47	38	9	28	10	17	11
Experimental	54	41	13	31	10	23	8
Total	101	79	22	59	20	40	19

4.4 Relationship between Effects of Factors upon CS2 Success (Hypothesis 2)

A 2x2x2 three-way analysis of variance statistical test has been carried out to determine the main factor effects upon CS2 success and the nature of relationships between these effects. The factors that have been taken into account are: a) the instructional tool (BBVPL / SBL), b) success at the CS1 course (Successful/Unsuccessful) and c) High-School Experience in GUI-Application Development (Experienced/inexperienced). The chart detailing tests of between-subject effects for the analysis has given in Table 3. According to analysis results, no three-way interaction between effects of the factors upon CS2 success has been found ($F(1,65) = .440$, $p = .509$). On the other hand, there is a significant two-way relationship between factors of instructional tool and high-school experience in GUI-application development ($F(1,65) = 7{,}010$, $p = .010$). Additionally, main effects upon CS2 success for each factor, namely the instructional tool ($F(1,65) = 6{,}349$, $p = .014$), high school experience in GUI-application development ($F(1,65) = 10{,}435$ $p = .02$) and success at the CS1 course ($F(1,65) = 8{,}147$ $p = .006$) have been found to be statistically significant.

Table 3. 3-way Analysis of Variance Tests of Between Subject Effects detailing the effects upon CS2 success the factors of instructional tool, CS1 success and high-school experience in GUI-App development

Source of Variance	Sum of Squares	df	Mean Square	F	(p)
Tool	2026,120	1	2026,120	6,349	,014
CS1 Success	2600,003	1	2600,003	8,147	,006
High Sch. Exp. in GUI App Dev.	3330,078	1	3330,078	10,435	,020
TxC	48,951	1	48,951	,153	,697
TxH	2236,920	1	2236,920	7,010	,010
CxH	27,800	1	27,800	,087	,769
TxCxH	140,536	1	140,536	,440	,509
Error	20743,155	65	319,125		
Total	141375,000	73			

4.5 CS2 Success of Novice Students Groups Depending on Instructional Tool Used (Hypothesis 3)

The CS2 post-test scores from subgroups comprised of students who lack high-school experience in GUI-application development and who have been unsuccessful in CS1 –namely, the "novices"- have been compared with an independent samples T-test; the independent variable being the instructional tool used. The results have been given in Table 4. According to the test, there is no significant difference between post-test scores of novice groups depending on the instructional tool used $t(27)=-.977$, $p>.005$. Although, it is interesting that the average scores for novices using BBVPL is relatively higher than novices using SBL

5. DISCUSSION

A review of literature shows that a blocks-based visual programming tool can be used in introductory programming courses (Gestwicki and Ahmad 2011), that doing so may increase student motivation (Wolber 2011) and that while the said tool increases interest in programming in novice students, it may also help maintain the motivation of experienced students (Karakus, Uludag et al. 2012). However, the scenario that has been investigated with this study has yielded results that are partially on the contrary with these claims, showing that as far as academic success is concerned, overall scores of students using a conventional syntax-based language (SBL) for learning object oriented GUI-Application development have been found to be significantly higher than those who learned by using the blocks-based visual programming language (BBVPL). It should be noted that all students in the scenario had previously completed a CS1 course where they were educated in structured programming concepts by using a syntax-based language (C#) for console-application development. This result may show that students who got accustomed to developing applications using a certain programming language/environment had difficulties adapting to a new and radical language/environment. It is believed that it may be useful to investigate the reasons of such a probable problem in adaptation. On the other hand, one reason that may have negatively affected the success of students who used the reference BBVPL, namely the MIT App Inventor for Android, is the fact that the said programming environment is still in a beta development stage, which suggests usability problems. Informal feedback from students received after the research application has finished; revealed that students found App Inventor to be rather slow and problematic compared to an IDE they previously used. Whereas, other issues such as requirement for constant Internet connection, lack of simple features such as copy/paste at components level and various bugs have all led several students to think that working with App Inventor was a "waste of time". In his study, Bhagi had also underlined several shortcomings of the App Inventor platform in terms of mobile game development (Bhagi 2012). It should therefore always be kept in mind in further research that MIT App Inventor is still indeed at beta development stage.

Another research finding indicates that; in addition to the programming language of choice, prior success in a structured programming undergraduate course (CS1), and high-school experience in object-oriented GUI application development are all factors that have significant main-effects upon object oriented GUI application development (CS2) success at undergraduate level. This result complies with studies in the current literature which claim that high school programming education positively effects success in undergraduate programming education (Hagan and Markham 2000, Wilson and Shrock 2001, Holden and Weeden 2003) and studies that claim existing knowledge in structured/procedural programming approaches positively effect success in object-oriented programming education (Sharp and Griffyth 1999). A two way interaction exists between effects upon CS2 success of ,the programming language of choice and high-school experience in object-oriented GUI application development. While students with high-school experience in GUI-app development do better in CS2 using the SBL; those without high-school experience do better in CS2 using the BBVPL. This clearly shows that, the effect of the instructional tool (programming language of choice) upon students' CS2 success is influenced by long term (high school) experience; which brings an explanation to the result observed at the testing of Hypothesis 1. In that, the fact that CS2 success among the experimental and control groups seems to be higher in the control group who used the SBL, may be specifically due to students in possession of long-term, high school experience in GUI-app development have failed to adapt to the use of BBVPL.

From factors that impact CS2 success, the lack of significant relationship between success in the previous undergraduate CS1 course, which involved education in structured programming concepts, and high-school experience in object-oriented GUI-application development; could be associated with the content of the object-oriented GUI application development courses received by students during high-school. At this point, it could be possible that the vocational high schools that taught object-oriented GUI application development may have followed an objects-first approach in teaching programming to their novice students; which may have resulted in the students being captivated with the graphics nature of applications and failing to improve their more basic and abstract programming skills associated with structured programming, namely, simple algorithm development, working with conditionals, loops, etc. This is supported by several studies in the literature stating that as programming education shifts from procedural to visual and object-oriented, students' skills in algorithm development and code generation get weaker and their academic success therefore drops (Beaubouef and Mason 2005, Reges 2006).

The fact that there is no role of the CS1 success factor in the impact of programming language of choice upon CS2 success, clearly shows how important a CS1 course that prioritizes teaching structured programming concepts is for programming education. Since, it can be inferred that, students who have been successful at a course that encompasses structured programming concepts (CS1) are prone to be successful in an object-oriented GUI-application development course (CS2) regardless of which programming language is used as an instructional medium. It can be said that, no matter how GUI-application development is found to be more interesting and relevant to real-world by students; a structured programming course dealing with more abstract concepts as a first-step in programming education could be more viable.

The research also showed that, the so-called "novice" students, who did not have prior programming experience from high school and who were found to be relatively unsuccessful at the structured programming education with a syntax-based language at the CS1 course; have done better in CS2 using the BBVPL (MIT App Inventor for Android) than using the SBL, although the difference was not statistically significant. MIT App Inventor for Android therefore still has a potential to be used as a tool for introducing novices to programming or reclaiming students who perhaps failed to adjust themselves to programming due to abstraction concerns or difficulties in coding and syntax-use. This result complies with studies in the literature that claim MIT App Inventor could be a useful tool for introducing novices to programming (Spertus, Chang et al. 2010, Uludag, Karakus et al. 2011).

6. CONCLUSION

Although it is claimed by researchers that developing applications for use on smart mobile devices could positively influence academic success in programming courses due to an increase in student motivation; it is suggested by this research that this claim can be influenced by various factors. It is also suggested that success in a previous course that teaches structured programming concepts (CS1) is an important factor in achieving success in an object-oriented GUI application development course (CS2), to the point that; even if the IDE and programming language used in CS1 is changed at CS2. With the motivation effect it brings about and the blocks-based concept, MIT App Inventor may have a potential to be used as a tool for introducing novices -who have no prior experience in the field- to computer programming. Additionally, it can also be used for reclaiming students who could not adjust to conventional syntax-based programming education. On the other hand, students who possess a long-term experience in object-oriented GUI-application development –possibly acquired at a vocational high school- prefer to continue education in the syntax-based language for developing applications targeting the desktop PC-that which they are accustomed to-, rather than in MIT App Inventor. One reason for this, could be the the usability concerns involving MIT App Inventor, which is still at a beta development stage.

ACKNOWLEDGEMENTS

The study was carried out as part of a Master's Degree in the Computer Education and Instructional Technologies Department at Marmara University Institute of Educational Sciences,. It has been funded by Marmara University Research Projects Board as research project number EGT-C-YLP-150513-0211.

REFERENCES

Ay, O.(2011). Mantıksal Hata Örneklerinin Kullanıldığı Programlama Eğitiminde Uygulanan Öğretim Yöntemleri Ve Öğrenci Deneyimlerinin Akademik Başarıya Etkisi, Marmara Üniversitesi Eğitim Bilimleri Enstitüsü Yüksek lisans Tezi.

Beaubouef, T. and J. Mason (2005). "Why the high attrition rate for computer science students: some thoughts and observations." SIGCSE Bull. 37(2): 103-106.

Bhagi, A. (2012). Android Game Development with AppInventor. Department of Electrical Engineering and Computer Science. Massachusetts, USA, Massachusetts Institute of Technology. Master of Engineering in Electrical Engineering and Computer Science: 94.

Dvorak, J. D. and K. Buchanan (2002). Using Technology To Create and Enhance Collaborative Learning, Association for the Advancement of Computing in Education (AACE), P.O. Box 3728, Norfolk, VA 23514. Tel: 757-623-7588; e-mail: info@aace.org; Web site: http://www.aace.org/DL/.

Gestwicki, P. and K. Ahmad (2011). "App inventor for Android with studio-based learning." J. Comput. Sci. Coll. 27(1): 55-63.

Hagan, D. and S. Markham (2000). "Does it help to have some programming experience before beginning a computing degree program?" SIGCSE Bull. 32(3): 25-28.

Holden, E. and E. Weeden (2003). The impact of prior experience in an information technology programming course sequence. Proceedings of the 4th conference on Information technology curriculum. Lafayette, Indiana, USA, ACM: 41-46.

Horn, M. S. and R. J. K. Jacob (2006). Tangible programming in the classroom: a practical approach. CHI '06 Extended Abstracts on Human Factors in Computing Systems. Montreal, Quebec, Canada, ACM: 869-874.

Horn, M. S. and R. J. K. Jacob (2007). Designing tangible programming languages for classroom use. Proceedings of the 1st international conference on Tangible and embedded interaction. Baton Rouge, Louisiana, ACM: 159-162.

Karakus, M., et al. (2012). Teaching computing and programming fundamentals via App Inventor for Android. Information Technology Based Higher Education and Training (ITHET), 2012 International Conference on.

Kinnunen, P. and Malmi, L. (2008). CS minors in a CS1 course. In *Proceeding of the Fourth international Workshop on Computing Education Research* (Sydney, Australia, September 06 - 07, 2008). ICER '08. ACM, New York, NY, 79-90.

Kurkovsky, S. (2009). "Engaging students through mobile game development." SIGCSE Bull. 41(1): 44-48.

Mahmoud, Q. H. and A. Dyer (2007). Integrating BlackBerry wireless devices into computer programming and literacy courses. Proceedings of the 45th annual southeast regional conference. Winston-Salem, North Carolina, ACM: 495-500.

Mccracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I. ve Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In *ITiCSE-WGR '01: Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 125-180, New York, NY, USA. ACM Press.

McNerney, T. (2004). "From turtles to Tangible Programming Bricks: explorations in physical language design." Personal and Ubiquitous Computing 8(5): 326-337.

Mohamad, S. N. H., et al. (2011). Principles and dynamics of block-based programming approach. Computers & Informatics (ISCI), 2011 IEEE Symposium on.

Papert, S. and I. Harel (1991). "Situating constructionism." Constructionism: 1-11.

Purcell, K. (2011). "Half of adult cell phone owners have apps on their phones." Pew Research Center's Internet & American Life Project. Accessed January 9: 2012.

Rau, P.-L. P., et al. (2008). "Using mobile communication technology in high school education: Motivation, pressure, and learning performance." Computers & Education 50(1): 1-22.

Reges, S. (2006). "Back to basics in CS1 and CS2." SIGCSE Bull. 38(1): 293-297.

Robins, A., Rountree, J., ve Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137-172.

Sharp, H. and J. Griffyth (1999). "The Effect of Previous Software Development Experience on Understanding the Object-Oriented Paradigm." Journal of Computers in Mathematics and Science Teaching 18(3): 245-265.

Soloway, E., Ehrlich, K., Bonar, J. and Greenspan, J. (1983). What do novice know about programming? In B. Shneiderman and A. Badre (eds), *Directions in Human-Computer Interactions*, Ablex, Norwood, NJ, 27-54.

Spertus, E., et al. (2010). Novel approaches to CS 0 with app inventor for android. Proceedings of the 41st ACM technical symposium on Computer science education. Milwaukee, Wisconsin, USA, ACM: 325-326.

Tillmann, N., et al. (2012). The future of teaching programming is on mobile devices. Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education. Haifa, Israel, ACM: 156-161.

Uludag, S., et al. (2011). Implementing IT0/CS0 with scratch, app inventor forandroid, and lego mindstorms. Proceedings of the 2011 conference on Information technology education. West Point, New York, USA, ACM: 183-190.

Wang, D., et al. (2011). T-Maze: a tangible programming tool for children. Proceedings of the 10th International Conference on Interaction Design and Children. Ann Arbor, Michigan, ACM: 127-135.

Wilson, B. C. and S. Shrock (2001). "Contributing to success in an introductory computer science course: a study of twelve factors." SIGCSE Bull. 33(1): 184-188.

Wolber, D. (2011). App inventor and real-world motivation. Proceedings of the 42nd ACM technical symposium on Computer science education. Dallas, TX, USA, ACM: 601-606.

Wyeth, P. and H. C. Purchase (2000). Programming without a computer: a new interface for children under eight. User Interface Conference, 2000. AUIC 2000. First Australasian.