SCAFFOLDING JAVA PROGRAMMING ON A MOBILE PHONE FOR NOVICE LEARNERS

Chao Mbogo[1], Edwin Blake[2], Hussein Suleman[3]
University of Cape Town, Department of Computer Science[1, 2, 3]
Cape Town, South Africa
chao.mbogo@uct.ac.za[1]; hussein@cs.uct.ac.za[2]; edwin@cs.uct.ac.za[3]

ABSTRACT

The ubiquity of mobile phones provides an opportunity to use them as a resource for construction of programs beyond the classroom. However, limitations of mobile phones impede their use as typical programming environments. This research proposes that programming environments on mobile phones should include scaffolding techniques specifically designed for mobile phones, and designed based on learners' needs. This paper discusses the effectiveness of theoretically-derived scaffolding techniques to construct Java programs on a mobile phone. The results indicate that even though scaffolding techniques could support learners to program on a mobile phone, further modifications of the designed scaffolding techniques may be necessary in order to more effectively support programming on a mobile phone.

KEYWORDS

Computer Programming, Scaffolding, Mobile, Evaluation.

1. INTRODUCTION

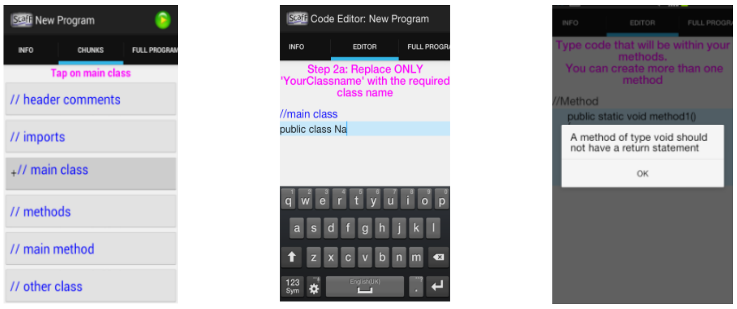
Computer programming is a difficult subject for most novice learners (Piteira & Costa 2012). In order to contribute towards tackling learning difficulties in the subject, learners can be supported to construct programs while they are outside the classroom. *Support* is emphasized because any such support should be additional to the learners' classroom learning and not a replacement. *Scaffolding* refers to support provided so that a novice learner can carry out a task, solve a problem or achieve a goal that would otherwise be beyond the learner's unassisted efforts (Wood et al. 1976).

Most learners at institutions in Africa are in resource-constrained environments where they have limited access to PCs and laptops while they are outside the classroom. The ubiquity of mobile devices provides an opportunity to use them as a resource for construction of programs beyond the classroom. A study in a developing country shows that mobile phones are the most available technological tools among learners (Kafyulilo 2012). Therefore, the mobile phone was selected as the resource that can be used for construction of programs outside the classroom. However, limitations of mobile phones, such as small screen size and small keypads, impede their use as typical programming environments.

In addition to addressing limitations of mobile phones, the challenges faced by learners of programming should be considered in order to maximize the potential of meeting learners' needs. Consequently, this research proposes that programming environments on mobile phones should include scaffolding techniques that are specifically designed for mobile phones and designed based on learners' needs.

In order to provide scaffolding techniques in a mobile programming environment, an Android application was designed based on a theoretical scaffolding framework, challenges faced by learners of programming, and limitations of mobile phones (Mbogo et al. 2014). The application supports construction of Java programs. In this paper, this application will be referred to as ScafFold. ScafFold provides 5 types of scaffolding techniques: (i) representation of a program in parts; (ii) restricting a learner to complete a program in a certain order; (iii) enabling construction of programs one part at a time; (iv) error correction; and (v) supportive scaffolding such as steps to guide the learner on how to interact with the application, provision of default code, and examples and hints related to the different program chunks.

Figure 1-left shows representation of a Java program in parts that indicates the structure of a program before a learner interacts with the different parts. At this screen a full program tab is also provided, which can be selected in order to view the full program at any time. The figure also shows the main class as the only one currently active; the learner has to successfully complete this before the next part is activated. An illustration of completion of a program one part at a time is shown in Figure 1-centre where only the main class is being constructed. The full program can also be viewed at this editor. Figure 1-right shows an error prompt if a void method is completed with a return statement. An example of supportive scaffolding is illustrated with the step instruction ‘Tap on main class’ at Figure 1-left. To compile the program, the learner presses the green run button at the top of Figure 1-left and the full program is sent to the ideone[1] online compiler and debugging tool. The results and output are sent back to the mobile interface.



Tap on main class

Step 2a: Replace ONLY 'YourClassname' with the required class name

```
//main class
public class Na
```

Type code that will be within your methods.
You can create more than one method

```
//Method
public static void method1()
```

A method of type void should not have a return statement

Figure 1. Interfaces of ScafFold showing examples of scaffolding techniques

To evaluate effectiveness of the scaffolding techniques, an empirical evaluation was conducted where some learners attempted Java programming tasks using ScafFold (the experimental group), and other learners attempted similar programming tasks using a mobile programming environment that did not offer any scaffolding (the control group). Java was selected as the language for construction of programs because it was the common language taught across the universities that participated in the study. ScafFold is not yet available for public download since it is still an experimental prototype. The data from the two groups of learners were analyzed to measure: (i) the number of tasks completed; (ii) the amount of time spent on the tasks; and (iii) the errors from the tasks. In addition, learners’ reflections were collected on the usefulness of the scaffolding techniques that they had interacted with to complete programs on the mobile phone. The contribution of this study is threefold: (i) to demonstrate the effectiveness of scaffolding techniques to support programming on a mobile phone; (ii) to highlight further modifications on the already designed scaffolding techniques; and (iii) to share learners’ reflections on the usefulness of the scaffolding techniques.

2. RELATED WORK

Test My Code (TMC) (Vihavainen et al. 2013) is a PC programming environment that enables learners to submit code created on IDEs to a remote server, from which instructors can perform manual code reviews. TMC offers scaffolding in the form of exercises with code snippets to be completed by the learner, and feedback that is displayed on the IDE once an instructor reviews the code. Similarly, PETCHA (Queirós & Leal 2012) a teacher-learner learning management system, works with Eclipse to scaffold a learner’s programming process by automatically creating a project on the IDE and performing code validation.

Indeed, significant work has been done on scaffolding learners while they use PCs to program, starting from earlier work by Gudzial (Guzdial et al. 1998). Yet, little work has been done to extend the implementation of scaffolding in order to support construction of programs on mobile phones. However, there are several mobile IDEs for Java programming available on the Google Play store, such as Sand IDE. However, the interfaces of these IDEs mostly mimic PC-based IDEs and do not offer scaffolds that would support a novice learner or address the limitations of mobile phones. Recent work by Microsoft (Tillmann et al. 2011) enables development of mobile apps using a new language - TouchDevelop - on the TouchDevelop programming environment where much of the code is created by tapping through menus. TouchDevelop also offers a guide to a user using an interactive tutorial that fades at some point. Even though TouchDevelop employs some scaffolding, it is a specialized language that was designed for a visual programming environment. Therefore, the techniques cannot be applied trivially to Object Oriented languages such as Java. In contrast, the aim of this research is to support construction of programs that are typically taught in an introductory course taught using Java, as opposed to creating mobile apps such as in TouchDevelop.

[1] http://ideone.com/

3. STUDY METHODOLOGY

In order to evaluate the effectiveness of scaffolding techniques, experiments were conducted with a total of 70 learners of programming from 3 universities in South Africa and Kenya: 27 learners at University of Western Cape (UWC); 14 learners at Kenya Methodist University (KeMU); and 29 learners at Jomo Kenyatta University of Agriculture and Technology (JKUAT). Despite the geographical and background differences between South Africa and Kenya, all learners were taking an introductory course in programming using Java. The teachers were asked for a set of 5 Java exercises relating to introductory topics that they had already taught in the course. At the time of the evaluation, all 3 groups had covered the topics of Java syntax, input-output, loops, methods, and classes.

Programming tasks for UWC group

1. Write a program that calculates the total cost of an item that is R159.72 and incurs a VAT of 14%.
2. Write a program that uses a for-loop to calculate the sum of the numbers from 1 to 50 and displays the sum and average.
3. Write a program that uses a method name() to print out your name.
4. Write a program that uses the Scanner input to ask for the user's name and age, and prints "Hello" + name " your age is" + age;
5. Write a program that uses a method input() to ask for height and width of a rectangle, and calculates and displays the area using height x width.

Programming tasks for KeMU group

1. Write a program that initializes x to 10 and prints out its double value.
2. Use the appropriate control structure to print the first 10 natural numbers.
3. Write a program that accepts two numbers as input and calculates the average.
4. Overload a method to print one and two integer values. Call these methods from the main method to output the integers 34, and 12 and 23, respectively.
5. Write a program that creates a class that contains the constructor below
   Item (int id, String title) { }.

Programming tasks for JKUAT group

1. Write a program that outputs "Scaffolding at JKUAT."
2. Write a program that computes the sum and average of the numbers 1-20.
3. Write a program that captures and displays the age of two students.
4. Write a program that uses a method to capture two integers and outputs their sum.
5. Write a program that initializes default values of name and age of a person in a constructor and outputs these in a main class.

The learners were issued with Android phones. All the phones used during the experiments were touchscreen smartphones with popup keyboards. A majority of the phones were Samsung Galaxy Pockets (S5300) and a few others were Samsung SIIs. The learners were randomly split into control groups and experimental groups and were then guided on how to download the respective application; the applications had been stored on Google drive. The learners were issued with printed copies of the exercises. Table 1 shows the distribution of the number of learners in the control and experimental groups in the three participating institutions. At JKUAT, the number of learners in the control group was higher than that in the experimental group because 2 learners used the download link for the non-scaffolded application.

Google Analytics was used to collect logs of the learners' interaction with the applications. At the end of the experiment the learners filled an online questionnaire. The questionnaire was designed using LimeSurvey and consisted of two parts: (i) Demography; this section was filled by all 70 learners; and (ii) reflections and perceptions on scaffolding techniques; this section was filled by the 37 learners in the experimental group.

Table 1: Distribution of learners across control and experimental groups.

Institution	Control	Experimental
UWC	13	14
KeMU	7	7
JKUAT	16	13

4. EVALUATION CRITERIA

Evaluation was done while learners interacted with the scaffolding techniques to construct programs on a mobile phone. Therefore, data was collected from learners' interaction with the application. In fact, evaluation models such as the CIAO model (Jones et al. 1999) have outlined that while evaluating educational technology one should consider data about learners' interaction with the software. This points to measuring performance because performance is all about what the user actually does in interacting with the product (Albert & Tullis 2008) and consists of five types of metrics: task success; time-on-task; errors; efficiency; and learnability. In this paper, task success, time-on-task, and errors will be discussed. Learnability and efficiency will be discussed in future work. In addition, qualitative feedback was obtained from learners on their reflections on the scaffolding techniques.

4.1Task Success

Task success measured if learners were able to complete a task. A *complete* task is a programming task that met three criteria: (i) a program that had at least the main class, the header, and the main method completed; (ii) a program that successfully compiled after completion of at least the parts in (i); and (iii) a program that produced the required output. To evaluate overall task success, measurement was done of: (i) number of attempted tasks; (ii) number of incomplete tasks; and (iii) number of completed tasks.

4.2 Time-on-task

Time-on-task is the time elapsed between the start of a task and end of a task. Time-on-task was measured between the start and end of a program for both complete and incomplete programs. For complete tasks, the end time refers to the first time the program was successfully compiled and produced the desired output.

4.3 Errors

Errors were evaluated by measuring: (i) the number of errors encountered after compiling and running the programs; and (ii) errors that triggered scaffolding techniques that offered support for error correction, for the experimental group.

4.4 Qualitative Feedback

Self-reported data was collected by learners reflectively indicating which scaffolding techniques they felt supported the construction of programs on the mobile phone. These subjective measures gave an indication of which scaffolding techniques learners felt could be useful.

5. RESULTS AND DISCUSSION

In KeMU and JKUAT, learners took part in 2-hour experiment sessions. At UWC, learners took part in a 1-hour experiment session. The difference in time was dependent on how long the groups of learners were available. Due to a technical challenge, the logs from the KeMU session were not obtained. However, the number of tasks that were completed was manually logged during the session, and the learners completed the online questionnaire at the end of the session. For this reason, KeMU's data was analyzed to measure only task success and subjective feedback.

5.1 Task Success

At KeMU, all the 7 learners in both experimental and control groups attempted the first and second tasks, with more learners in the experimental group completing these tasks (Table 2); a total of 71% in the

Table 2: Number of learners who attempted (A) and completed (C) each task in control and experimental groups.

	KeMU				UWC				JKUAT			
	Control		Experimental		Control		Experimental		Control		Experimental	
	A	C	A	C	A	C	A	C	A	C	A	C
Program 1	7	3	7	6	12	7	14	12	11	9	13	9
Program 2	7	1	7	4	6	2	10	6	14	11	10	5
Program 3	2	1	4	1	1	0	6	2	12	2	5	1
Program 4	0	0	0	0	1	0	2	0	2	1	2	1
Program 5	0	0	0	0	0	0	0	0	1	0	1	0

experimental group compared to 21% in the control group. This indicates that the scaffolded environment enabled the learners to attempt more tasks than in the un-scaffolded environment. The learners at KeMU were not able to attempt the last two tasks and they indicated that they also struggled with topics of methods, classes and constructors in the classroom, considering that for most of them this was the first time they were learning programming using Java. This indicates that even though scaffolding could be useful in programming on a mobile environment, the learner has to be able to link what they learn in the classroom with the programs they need to write in a mobile programming environment.

At UWC, 12 learners in the control group attempted the first programming task, with 7 of them successfully completing it. The 14 learners in the experiment group attempted the first programming task with 12 learners completing it. In addition, more learners in the experimental group were able to attempt tasks than those in the control group. This indicates that the scaffolded environment offered the support the learners needed to attempt more programming tasks. No learner was able to attempt the last program, perhaps due to the time constraint of the experiment session being conducted in just 1 hour.

Table 2 indicates that more learners in the control group were able to complete the second task at JKUAT. Further, more learners in the control group were able to attempt the third task. However, it was observed that the learners in the control group accessed previously attempted programs that were stored on the mobile phone, and reloaded them to the interface in order to edit them, as opposed to constructing them from scratch. This could be attributed to how the learners construct programs on a PC by copying old programs to the programming environment and editing to suit a new program. Learners reloading and editing previously completed programs could have contributed to them attempting more tasks than the experimental group, where the interface was more restrictive.

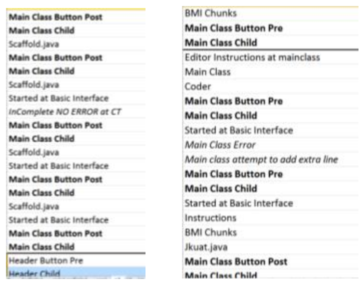
5.2 Time-on-task

Table 3 shows the average time taken on each program at UWC and JKUAT, in minutes and seconds, for attempted and completed tasks. Learners in the experimental group at UWC took longer to complete the first program than those in the control group, perhaps due to familiarizing themselves with the application. However, the learners in the experimental group took a shorter time in the subsequent programs. This indicates two things: (i) after the initial familiarization with a scaffolded environment, the time taken by learners to complete a program reduces in subsequent programs; and (ii) learners using scaffolded environment complete programs quicker than learners using a non-scaffolded environment.

Table 3: Average time on each task for attempted (A) and completed (C) tasks in control and experimental groups.

	UWC				JKUAT			
	Control		Experimental		Control		Experimental	
	A	C	A	C	A	C	A	C
Program 1	22:54	20:19	14:12	25:42	21:21	15:15	29:52	14:10
Program 2	19:19	20:33	10:47	15:30	43:11	29:10	56:03	35:12
Program 3	5:20	-	6:35	14:47	49:51	19:30	16:13	39:05
Program 4	1:18	-	7:07	-	28:01	16:00	4:13	19:42
Program 5	-	-	-	-	6:39	-	47:38	-

At JKUAT, learners in the experimental group took longer on tasks than in the control group for the first two tasks. This also explains why there were more attempts at tasks in the control group at JKUAT, as previously indicated in Table 2. Further analysis shows that learners in the experimental group went back and forth between the chunks, therefore spending more time on the program. For example, Figure 3-left shows a program in which a learner went back to the created ‘main class’ a total of 5 times (in bold), did a test of the program once (in italics), before finally starting on creating the header. In Figure 3-right, the learner attempted to create the main class three times (in bold), including encountering two errors (in italics), before finally succeeding (at ‘Main Class Button Post’). In both cases the learners did not view the full program while going back and forth in the chunk. This indicates that learners may need to be more effectively supported to view the current full program while they are moving between the chunks.



Main Class Button Post
Main Class Child
Scaffold.java
Main Class Button Post
Main Class Child
Scaffold.java
Started at Basic Interface
InComplete NO ERROR at CT
Main Class Button Post
Main Class Child
Scaffold.java
Started at Basic Interface
Main Class Button Post
Main Class Child
Scaffold.java
Started at Basic Interface
Main Class Button Post
Main Class Child
Header Button Pre
Header Child

BMI Chunks
Main Class Button Pre
Main Class Child
Editor Instructions at mainclass
Main Class
Coder
Main Class Button Pre
Main Class Child
Started at Basic Interface
Main Class Error
Main class attempt to add extra line
Main Class Button Pre
Main Class Child
Started at Basic Interface
Instructions
BMI Chunks
Jkuat.java
Main Class Button Post
Main Class Child

Figure 3. Demonstration of learners going back and forth on the same chunk of a program (in bold).

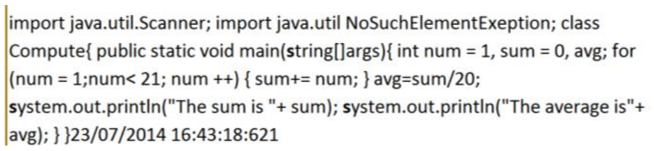
5.3 Errors

Table 4 shows the number of learners who encountered multiple compile/run (C/R) errors and errors that were captured as a scaffolding technique (Scaffolded errors) for each attempted task, for both groups at UWC and JKUAT. At UWC, a higher number of learners in the control group than in the experimental group encountered multiple compile/run errors. At JKUAT, 12 learners in the experimental group encountered errors that generated an error prompt as a scaffolding technique in the first programming task, and then 10 learners had no compile/run errors in the same task. This indicates that if errors are captured earlier in the program creation as a scaffolding technique, the learners are likely to encounter fewer errors during compiling and running the program.

Table 4: Number of learners who encountered multiple compile/run(C/R) errors and scaffolded errors in attempted tasks in control and experimental groups.

	UWC			**JKUAT**		
	Multiple C/R errors		**Scaffolded errors**	**Multiple C/R errors**		**Scaffolded errors**
	Experimental	**Control**	**Experimental**	**Experimental**	**Control**	**Experimental**
Program 1	0	4	2	3	7	12
Program 2	3	4	0	5	10	6
Program 3	2	2	0	3	11	2
Program 4	1	2	0	0	2	0
Program 5	-	-	-	1	1	0

In addition, it was observed that learners in the control group had syntactical errors that could be reduced by scaffolding techniques found in ScafFold. For example, Figure 4 shows a program of a learner in the control group in which the keywords ‘String’ and ‘System’ were written with a lower case ‘s’ (in bold). In ScafFold, a scaffolding technique that provides such default statements as ‘System.out.println()’ that a learner can reuse, reduces the occurrence of such syntax errors. Lastly, it was noted that none of the programs written by learners in the control group contained header comments (as can be seen from Figure 4); this, as opposed to ScafFold which guides the learner to create header comments.



```
import java.util.Scanner; import java.util NoSuchElementExeption; class
Compute{ public static void main(string[]args){ int num = 1, sum = 0, avg; for
(num = 1;num< 21; num ++) { sum+= num; } avg=sum/20;
system.out.println("The sum is "+ sum); system.out.println("The average is"+
avg); } }23/07/2014 16:43:18:621
```

Figure 4. A program showing syntactical errors ('s' in bold).

5.4 Qualitative feedback

Excerpts of some of the learners from the experimental group are cited verbatim, on their reflections on the use of scaffolding.

'I really enjoyed the program. It is structured, there's a tab for methods, a tab for main, a tab for classes. And it allows you to go through them by order. It highlights where you made a mistake and allows you to go back and fix errors.'

'The application divides the program or code into sections then one can the track and write the code properly by following the sections.'

'Preset statement helped in typing. The sections are well laid out. The hints helped in where to type. The error handling is accurate in pinpointing errors.'

Learners indicated that the following scaffolding techniques could further support programming on a mobile phone:

'I didn't see the part that creates a "constructor as simple as creating the main method…., the double clicks makes one lose patience....at this i can only recommend it to a friend if they are writing a very short program.'

'Would be great if there were a few imports (packages) that are commonly used that are in the preset menu.'

6. CONCLUSION

This paper has reported on results of an evaluation with 70 learners of a programming course taught using Java, in 3 universities. To attempt programming tasks using Java, some learners used a scaffolded mobile programming environment, while others used a non-scaffolded mobile programming environment. The data from these experiments were analyzed for each group to measure the number of tasks attempted and completed, time-on-task, and the errors encountered.

The results indicate that specifically designed scaffolding techniques for mobile phones can enable learners to construct programs on a mobile phone and meet learners' needs. For example, learners were able to interact with the restricted interface and created complete programs, once chunk at a time. In addition, the results show that learners using a scaffolded environment can be supported to avoid errors that would otherwise be encountered in a non-scaffolded environment. Further, learners indicated that they found the scaffolding techniques useful in supporting construction of programs on a mobile phone. Specifically, learners indicated that scaffolding techniques such as: restricting the order of program completion; guiding the learner to complete a program; enabling construction of programs one part at a time; provision of default code; and provision of examples and hints, as most useful to support construction of programs on a mobile phone.

However, the results indicate that even though creation of the program one part at a time, as a scaffolding technique, could support learners of programming to construct programs on a mobile phone, learners may need to be more effectively supported to view the full program at all times while working on the different chunks. Also, the results show that scaffolding techniques may not be useful if learners cannot link programming knowledge from the classroom to the program they are attempting in a mobile environment.

Although the findings are encouraging and useful, the study in this paper has certain limitations. Firstly, the study did not evaluate if learners who used ScafFold eventually performed better in programming than those who used the non-scaffolded environment. Secondly, a future experiment needs to ensure that learners in the control group write programs from scratch (instead of reloading previously completed programs) in order to ensure an equal baseline with the experimental group. Third, further work will conduct a finegrained analysis of the specific errors encountered in both the scaffolded and non-scaffolded environment. In addition, further work will conduct analysis on learnability and efficiency. Fourth, learners in the control group will be asked on their reflections on the use of a mobile phone to construct programs. Lastly, additional experiments will be conducted in order to increase the confidence of the conclusions and to capture feedback from learners in the control group on perceptions of writing programs on a mobile phone.

ACKNOWLEDGEMENT

This work was supported by funding from the Hasso Plattner Institute, Google Anita Borg, and resources in ICT for Development laboratory at University of Cape Town.

REFERENCES

Albert, W. & Tullis, T., 2008. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*, Morgan Kaufmann.

Guzdial, M. et al., 1998. Supporting Programming and Learning-to-Program with an Integrated CAD and Scaffolding Workbench. *Interactive Learning Environments*, 6(1-2), pp.143–179. Available at: http://www.tandfonline.com/doi/abs/10.1076/ilee.6.1.143.3609 [Accessed February 19, 2014].

Jones, A. et al., 1999. Contexts for evaluating educational software. *Interacting with Computers*, 11(5), pp.499–516. Available at: http://iwc.oxfordjournals.org.ezproxy.uct.ac.za/content/11/5/499.short [Accessed March 31, 2014].

Kafyulilo, A., 2012. Access, use and perceptions of teachers and students towards mobile phones as a tool for teaching and learning in Tanzania. *Education and Information Technologies*, 19(1), pp.115–127. Available at: http://link.springer.com/10.1007/s10639-012-9207-y [Accessed July 30, 2014].

Mbogo, C., Blake, E. & Suleman, H., 2014. Supporting the Construction of Programs on a Mobile Device: A Scaffolding Framework. In *Proceedings of 4th International Conference on M4D Mobile Communication for Development*. Dakar, Senegal, p. 155. Available at: http://people.cs.uct.ac.za/~edwin/MyBib/2014-m4d.pdf [Accessed March 11, 2014].

Piteira, M. & Costa, C., 2012. Computer programming and novice programmers. In *Proceedings of the Workshop on Information Systems and Design of Communication - ISDOC '12*. New York, New York, USA: ACM Press, pp. 51–53. Available at: http://dl.acm.org/citation.cfm?id=2311917.2311927 [Accessed May 31, 2014].

Queirós, R.A.P. & Leal, J.P., 2012. PETCHA. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE '12*. Haifa, Israel: ACM Press, p. 192. Available at: http://dl.acm.org/citation.cfm?id=2325296.2325344 [Accessed February 19, 2014].

Tillmann, N. et al., 2011. TouchDevelop. In *Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software - ONWARD '11*. New York, New York, USA: ACM Press, p. 49. Available at: http://dl.acm.org/citation.cfm?id=2048237.2048245 [Accessed February 8, 2014].

Vihavainen, A. et al., 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*. Canterbury, England: ACM Press, p. 117. Available at: http://dl.acm.org/citation.cfm?id=2462476.2462501 [Accessed February 19, 2014].

Wood, D., Bruner, J.S. & Ross, G., 1976. The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, 17(2), pp.89–100. Available at: http://doi.wiley.com/10.1111/j.1469-7610.1976.tb00381.x [Accessed March 11, 2014].