IADIS International Journal on WWW/Internet
Vol. 14, No. 1, pp. 101-124
ISSN: 1645-7641

EVALUATING THE EFFECT OF USING SCAFFOLDING TECHNIQUES TO SUPPORT JAVA PROGRAMMING ON A MOBILE PHONE

Chao Mbogo. *Kenya Methodist University*

Edwin Blake and Hussein Suleman. *University of Cape Town*

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 proposed that programming environments on mobile phones could include scaffolding techniques specifically designed for mobile phones, and designed based on learners' needs. Experiments were conducted with 142 learners from three universities in Kenya and South Africa in order to investigate the effect on learners of using the theoretically-derived scaffolding techniques to construct Java programs on a mobile phone. The results provided empirical evidence that scaffolding techniques specifically designed for mobile phones and designed based on learners' needs could effectively support the construction of programs on a mobile phone.

KEYWORDS

Mobile Phone, Java, Programming, Scaffolding, Effect

1. INTRODUCTION

Computer programming is a difficult subject for most learners of programming. Research indicates this to be a universal problem, especially among novice learners (Watson & Li 2014). The learning difficulties in the subject indicate that some programming skills are beyond the novice learners' efforts. Scaffolding refers to support provided so that the learner can engage in activities that would otherwise be beyond their abilities or their unassisted efforts (Wood et al. 1976). In order to contribute towards tackling learning difficulties in programming, novice learners can be supported to construct programs while they are outside the classroom. This makes any such support to be additional to the learner's classroom learning, and not a replacement. Support to learners outside the classroom can be provided using PC-based applications. Indeed, several studies have offered scaffolded environments on PC platforms targeting novice learners of programming, for example, 3D environments such as Alice (Dann et al. 2011).

While the availability of PC-based scaffolded programming environments is notable, most learners who are in resource-constrained environments, such as in parts of Africa, have limited access to PCs while they are outside the classroom. In fact, in many developing countries, people are much more likely to use computers at school or at work than to own them at home. For example, a recent study conducted among 36,619 people in 32 emerging and developing countries showed that computer ownership rates are lowest in sub-Saharan African nations where roughly a quarter or fewer have computers at home in every one of these countries, with the fewest in Uganda, where just 3% said that they have a computer (Pew Research Center 2015). The limited access to PCs outside the classroom aggravates the learning difficulties faced by learners. In fact, research conducted in Tanzania highlights that one of the contributors to learners struggling in programming is lack of adequate access to computers, which limits hands-on learning (Apiola & Tedre 2011).

In developing countries, the ubiquity of mobile devices hold enormous promise as the single ICT most likely to deliver education, and to do so in a sustainable, equitable and scalable basis (Traxler 2011). Mobile devices include laptops, tablets and mobile phones. Of these, mobile phones are the most widely used mobile devices among learners in developing countries (Kafyulilo 2012). Thus, the mobile phone was selected as the resource that could be used to learn programming outside the classroom. However, limitations of mobile phones, such as small screen size and small keypads, impede their use as typical programming environments. To deal with these limitations, and for handheld devices to become effective learning tools, the unique design challenges inherent in such a system must be understood (Luchini et al. 2002).

In addition to addressing limitations of mobile phones, the challenges faced by learners of programming should be considered. This is because addressing these challenges maximizes the potential of meeting learners' needs. In providing scaffolding, the needs of learners can be placed at the center of the design process. Such an approach was defined as learner-centered design, which claims that software can embody scaffolding that can address learners' needs (Soloway et al. 1996) - a design method that is emphasized as important in computing education (Guzdial 2015). Consequently, this research proposed that programming environments on mobile phones could include scaffolding techniques that are specifically designed for mobile phones and designed based on learners' needs.

1.1 Designed Scaffolding Techniques

In order to provide scaffolding techniques in a mobile programming environment, an Android application was designed based on a theoretical scaffolding framework (Quintana et al. 2004), challenges faced by learners of programming, and limitations of mobile phones (Mbogo et al. 2014). The application was designed to support construction of Java programs. 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. Three types of scaffolding techniques were designed: (i) static scaffolding that never fades; (ii) automatic scaffolding that is automatically provided at first but fades with time or can be cancelled by the user; and (iii) user-enabled scaffolding that is not automatically provided and the learner has to initiate its use.

Figure 1 shows the designed main interface with parts of a Java program. In this main interface the learner clicks on the button that relates to the part they need to work on. Figure 1 shows only the main class as enabled and can be constructed at this stage. Until the learner correctly creates the main class the other parts of the program remain disabled. Thereafter, the learner is guided to create the header comments part then the main method part and so on. The program layout is retained even when learners progress to an advanced interface, where the order of program creation is not restricted. Thus, the program layout is a static scaffolding technique since it does not change or fade away with time. On clicking each program part on the main interface another interface is opened with an editor that provides creation of only the selected program part. For example, Figure 2 shows creation of only the main method. The ability to work on one part of the program at a time could assist in working with the small screen. Because of the restriction of a small screen size, which remains unchanged, this scaffold is static and does not fade. While working on a program part (for example, while editing the main method in Figure 2), a learner could swipe to the full program interface and view the whole program at the state at which it was last saved (Figure 3). This is an example of user-initiated scaffolding. Another example of user-initiated scaffolding are examples and hints (Figure 4) that pop up when the example menu is selected. To compile the program, the learner presses the run button at the top-right of Figure 1 and the full program is sent to the ideone online compiler and debugging tool (Sphere Research Labs 2010). The results and output are sent back to the mobile interface.

Figure 1. Main interface

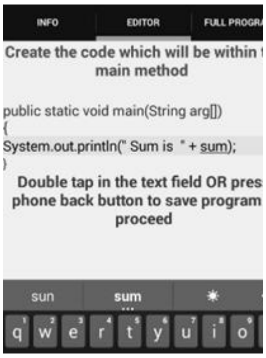

Figure 2. Editor interface

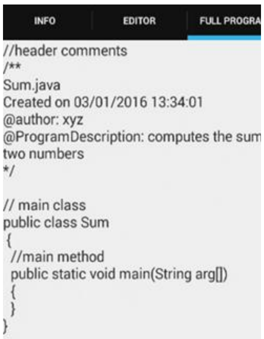

Figure 3. Full program as was last saved.

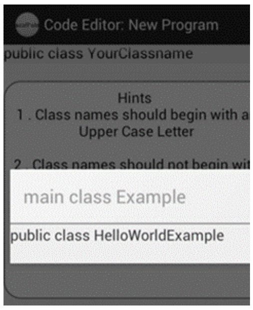

Figure 4. Main class example and hints.

Following the design of these scaffolding techniques the aim of this paper was to answer one research question: What is the effect on learners of using the theoretically-derived scaffolding techniques to construct Java programs on a mobile phone? By learners constructing programs using the derived scaffolding techniques in an experimental group and some constructing programs using a non-scaffolded environment in a control group, the study investigated the effects of the scaffolding techniques. The contribution of this paper is twofold: (i) to provide empirical evidence on the effect of scaffolding techniques to support Java programming on a mobile phone; and (ii) implications of the study.

1.2 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.

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 some 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 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 applications such as in TouchDevelop.

2. EVALUATION

2.1 Participants and Experiment Design

In order to evaluate the effect of using the scaffolding techniques, two experiments were conducted with a total of 142 learners of Java programming from 3 universities in South Africa and Kenya: University of Western Cape (UWC); Kenya Methodist University (KeMU); and Jomo Kenyatta University of Agriculture and Technology (JKUAT). Participants were randomly split into a control and an experimental group. The control group used a non-scaffolded environment and the experimental group used the scaffolded environment. Table 1 shows the distribution of learners in control and experimental groups across the 3 institutions. In the first and second experiments at KeMU and JKUAT learners took part in 2-hour experiment sessions. In the first experiment at UWC learners took part in a 1-hour experiment session. The difference in time depended on how long learners were available for.

Table 1. Distribution of learners in two experiments across experimental and control groups

Experiment	Institution	Total Number of learners	Number of learners in experimental group	Number of learners in control group
One	UWC	27	14	13
	KeMU	14	7	7
	JKUAT	29	13	16
Two	KeMU	24	13	11
	JKUAT	48	24	24

2.2 Programming Tasks

In the two experiments, four different sets of programming exercises were used: three different exercises for the first experiments at UWC, KeMU and JKUAT; and one set of similar exercises for the second experiments at KeMU and JKUAT. In the first experiment, the exercises were obtained from the different teachers of the courses in their respective institutions. In the second experiment, learners from both KeMU and JKUAT had covered similar topics in introduction to Java programming. Therefore, the exercises from the respective teachers were combined into one set. Despite the differences in the first and second sets of exercises, all the exercises covered introductory topics in Java. These tasks are presented in Figure 5 and Figure 6.

Programming Task for UWC group in Experiment 1

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 calculate and display the area using height x width.
6. Write a program that determines if a number input by a user is odd or even.

Programming Task for KeMU group in Experiment 1

1. Write a program that initialises x to 10 and prints out its double value.
2. Use the appropriate control structures to print out the first 10 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 number 34, and 12 and 24, respectively.
5. Write a program that creates a class that contains the constructor: *Item(int id, String title) { }*

Programming Task for JKUAT group in Experiment 1

1. Write a program that output 'Scaffolding at JKUAT'.
2. Write a program that computes the sum and average of the number 1-20.
3. Write a program that captures and displays the ages of two students.
4. Write a program that uses a method to capture two integers and outputs their sum.

Figure 5. Programming tasks attempted by learners in the first Experiment

1. Write a program that initialises x to 10 and prints out its double value.
2. Using a for-loop print the first 10 natural numbers.
3. Write a program that accepts input from the user and displays this as
   "Your input is " + input.
4. Write a program that uses a method **input()** to capture and display the names of two students.
5. Write a program that creates two classes. The second class contains the constructor below. Access this constructor from the main class. *Output() { System.out.println("Constructor called"); }*
6. Write a program that uses a for-loop within a method avg() to calculate the sum of the numbers 20-100 and displays the sum. Call this method from the main method.

Figure 6. Programming tasks attempted by learners in the second Experiment

2.3 Experiment Procedure

At each experiment session participants were first introduced to the purpose of the research and were guided through completion of a consent form. Participants were then randomly divided into control and experimental groups. Participants were issued with Android phones containing the application. 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. Due to the use of the Internet for collecting computer logs and use of the ideone online compiler, participants were issued with airtime to cover data costs where there was no Wi-Fi. Participants were then issued with printouts containing the programming tasks. During the experiment sessions Google Analytics was used to collect logs of the learners' interaction with the applications. After the experiment sessions, participants were asked to fill out an online questionnaire designed using LimeSurvey and consisted of two parts: (i) Demography section that was filled by all 142 learners; and (ii) reflections and perceptions on scaffolding techniques section that was filled by the 71 learners in the experimental group. At the end of the experiment sessions participants returned the phones that were issued.

3. EVALUATION CRITERIA

The data from this study was analyzed to measure: task success; time-on-task; and errors (Albert & Tullis 2008). Further, learnability was measured for only the experimental group in which learners used the scaffolded environment. In addition qualitative feedback was collected from learners in the experimental group. Considering these metrics, this research led to sub-questions related to each metric as discussed in the next sub-sections. The control and experimental groups were independent as each was subjected to one treatment (scaffolded or non-scaffolded environment). Therefore, the two-sample t-test was used to determine if the unknown means of the various metrics are different from each other (Elliott & Woodward 2007). t-tests are often used when only small samples are available ($n < 30$) (Harmon 2011). Since analysis was conducted per university, per experiment, the sample sizes in all the cases were less than 30.

3.1 Task Success

Task success was measured by analyzing the level of completion of tasks. A complete programming task is one that met all three criteria: (i) had all the required program parts completed; (ii) successfully compiled after completion of the required parts; and (iii) produced the required output. Incomplete tasks are tasks that failed to meet at least one of the criteria for completeness. To measure the effect of using the scaffolding techniques task success results from the control group and the experimental group were compared. This led to the first research sub-question: What is the effect of using the scaffolding techniques on task success?

The hypotheses derived for task success for attempted tasks were: (i) H_0: The mean number of attempted tasks in the experimental group is not larger than the mean number of attempted tasks in the control group; and (ii) H_1: The mean number of attempted tasks in the experimental group is larger than the mean number of attempted tasks in the control group.

Some tasks could be attempted and completed. Therefore, the hypotheses derived for task success for attempted and completed tasks were: (i) H_0: The mean number of completed tasks in the experimental group is not larger than the mean number of completed tasks in the control group; and (ii) H_1: The mean number of completed tasks in the experimental group is larger than the mean number of attempted tasks in the control group. A one-tailed t-test was used to test these hypotheses.

3.2 Time-on-task

Time-on-task was the duration between the start and end of a program for both complete and incomplete programs. The end-time for complete programs referred to the first time the program compiled successfully and produced the desired output. The end-time for incomplete programs referred to the time the user quit working on the program. Data for time-on-task was measured by considering three criteria (Sauro & Lewis 2012): (i) task completion time for completed tasks; (ii) time until failure for incomplete tasks; (iii) and total time per user for both incomplete and completed tasks. This led to the second sub-question: What is the effect of using the scaffolding techniques on time-on-task?

To address this sub-question, time-on-task results between the control group and the experimental group were compared. The hypotheses derived for time on completed tasks were: (i) H_0: The mean completion time in the experimental group is not less than the mean time on complete tasks in the control group; and (ii) H_1: The mean completion time in experimental group is less than the mean time on complete tasks in control group.

The hypotheses derived for time on incomplete tasks were: (i) H_0: The mean time on incomplete tasks in the experimental group is not less than the mean time on incomplete tasks in the control group; and (ii) H_1: The mean time on incomplete tasks in the experimental group is less than the mean time on incomplete tasks in the control group. A one-tailed t-test was used to test these hypotheses.

3.3 Errors

Two types of errors were evaluated: (i) the number of run-time errors for all the programs in the control and experimental groups; and (ii) errors that triggered scaffolding techniques that offered support for error detection, only for the experimental group. This led to the third sub-question: What is the effect of using the scaffolding techniques on the number of errors?

To address this sub-question, the number of run-time errors between the control groups and the experimental groups were compared. The hypotheses that were derived for errors were: (i) H_0: The mean number of run-time errors encountered in the experimental group is not lower than the number of run-time errors encountered in the control group; and (ii) H_1: The mean number of run-time errors encountered in the experimental group is lower than the number of run-time errors encountered in the control group. A one-tailed t-test was used to test these hypotheses.

3.4 Learnability

The data from time-on task was used to evaluate learnability. A comparison was made between time-on-task from one task to the next. This analysis considered only the experimental group because the aim was to investigate the learnability of the scaffolded environment. This led to the fourth sub-question: What is the effect of using the scaffolding techniques on time-on-task over time?

3.5 Qualitative Feedback

Self-reported data was collected by learners reflectively indicating their perceptions on using the scaffolding techniques.

4. RESULTS AND DISCUSSION

This section reports results from the two experiments. Due to a technical challenge, the logs from KeMU's first experiment session were not recorded. However, the number of tasks that were completed was recorded manually and the learners completed the questionnaire at the end of the session. For this reason, KeMU's data in the first experiment was analyzed to measure only task success and qualitative feedback.

4.1 Task Success – First Experiment

Table 2 shows the statistical results for attempted and completed tasks in the first experiment. At KeMU, there was no significant difference between mean number of attempted tasks in the experimental group and the mean number of attempted tasks in the control group. With a p-value of 0.16, the first null hypothesis cannot be rejected. Therefore, the mean number of attempted tasks in the experimental group is not larger than the mean number of attempted tasks in the control group. However, there was a significant difference between the mean number of completed tasks in the experimental group and the mean number of completed tasks in the control group at KeMU. With a p-value of 0.02, the second null hypothesis is rejected in favour of the second alternate hypothesis. Therefore, the mean number of completed tasks in the experimental group is larger than the mean number of completed tasks in the control group. The learners at KeMU were not able to attempt the last two tasks and they indicated that they struggled with topics of methods, classes and constructors in the classroom, considering that for most of them this was the first time to learn programming using Java.

Table 2. Statistical task success results for attempted and completed tasks in the first Experiment

		Attempted Tasks		Completed Tasks	
Institution	**Statistical Metric**	**Experimental Group**	**Control Group**	**Experimental Group**	**Control Group**
KeMU	***M***	2.57	2.29	1.57	0.71
	SD	0.53	0.49	0.53	0.75
	t	t (12) = 1.04		t (11) = 2.44	
	p	**0.16**		**0.02**	
UWC	***M***	2.29	1.54	1.57	0.69
	SD	1.07	0.88	1.02	0.63
	t	t (25) = 1.99		t (22) = 2.72	
	p	**0.03**		**0.006**	
JKUAT	***M***	2.38	2.50	1.23	1.44
	SD	1.04	-0.33	1.17	0.89
	t	t (22) = 1.04		t (22) = 0.52	
	p	**0.37**		**0.30**	

At UWC, there was a significant difference between the mean number of attempted tasks in the experimental group and the mean number of attempted tasks in the control group. With a p-value of 0.03, the first null hypothesis is rejected in favour of the first alternate hypothesis. Therefore, the mean number of attempted tasks in the experimental group is larger than the mean number of attempted tasks in the control group. Similarly, there was a significant difference between the mean number of completed tasks in the experimental group at UWC and the mean number of completed tasks in the control group. With a p-value of 0.006, the second null hypothesis is rejected in favour of the second alternate hypothesis. Therefore, the mean number of completed tasks in the experimental group is larger than the mean number of completed tasks in the control group. Further, some learners in the experimental group at UWC were able to complete the third task, while no learner in the control group was able to complete this task. Lastly, no learner was able to attempt the last program, perhaps due to the time constraint of the experiment session being in just 1 hour.

At JKUAT, there was no significant difference between the mean number of attempted tasks in the experimental group and the mean number of attempted tasks in the control group. With a p-value of 0.37, the first null hypothesis cannot be rejected. Therefore, the mean number of attempted tasks in the experimental group is not larger than the mean number of attempted tasks in the control group. Similarly, there was no significant difference between the mean number of completed tasks in the experimental group and the mean number of completed tasks in the control group. With a p-value of 0.30, the second null hypothesis cannot be rejected. Therefore, the mean number of completed tasks in the experimental group is not larger than the mean number of completed tasks in the control group.

4.1.1 Discussion: Task Success in the First Experiment

Of the three experiment sessions at UWC, KeMU and JKUAT, one resulted in a significantly higher number of attempted tasks in the experimental group than in the control group, and two resulted in significantly higher number of completed tasks in the experimental groups than in the control groups. Further, some learners at UWC's experimental group were able to complete the third task while no learner in the control group completed the same task. These results indicate that the scaffolding techniques enabled completion of more programming tasks than the non-scaffolded environment. The qualitative feedback by learners at KeMU on the difficulty of some programming aspects affecting their ability to use the scaffolded environment indicated that, even with a scaffolded environment, there needs to be a connection between what learners have learnt in the classroom and the use of the environment.

A further analysis was conducted to understand the results at JKUAT. It was noted that learners in the control group accessed previously attempted programs that were stored on the mobile phone, and reloaded them to the interface to edit them. This could be because learners found it cumbersome to type each program from scratch on the small interface of the mobile phone. It could also be attributed to how leaners construct programs on a PC by copying old programs to the programming environment and editing them to suit a new program.

These results warranted further study where learners in both the control and experimental groups could write the programming tasks from scratch, and hence provide a uniform baseline for both groups. Further, in order to understand why learners were not able to attempt all tasks, the post-experiment questionnaire was redesigned to include a relevant question. In addition, since the results from KeMU were not used for the entire analysis, there was a need to conduct additional experiments in order to strengthen the conclusions. Consequently, a second experiment was conducted.

4.2 Task Success – Second Experiment

Table 3 presents the statistical results for attempted and completed tasks in the second experiment. At KeMU, there was a significant difference between the mean number of attempted tasks in the experimental group and the mean number of attempted tasks in the control group. With a p-value of 0.03, the first null hypothesis is rejected in favor of the first alternate hypothesis. Therefore, the mean number of attempted tasks in the experimental group is larger than the mean number of attempted tasks in the control group. Similarly, there was a significant difference between the mean number of completed tasks in the experimental group at KeMU and the mean number of completed tasks in the control group. With a p-value of 0.0003, the second null hypothesis is rejected in favor of the second alternate hypothesis. Therefore, the mean number of completed tasks in the experimental group is larger than the mean number of completed tasks in the control group.

At JKUAT, there was a significant difference between the mean number of attempted tasks in the experimental group and the mean number of attempted tasks in the control group. With a p-value of 0.004, the first null hypothesis is rejected in favor of the first alternate hypothesis. Therefore, the mean number of attempted tasks in the experimental group is larger than the mean number of attempted tasks in the control group. Similarly, there was a significant difference between the mean number of completed tasks in the experimental group than in the control group. With a p-value of 0.0004, the second null hypothesis is rejected in favor of the second alternate hypothesis. Therefore, the mean number of completed tasks in the experimental group is larger than the mean number of attempted tasks in the control group.

Table 3. Statistical task success results for attempted and completed tasks in the second Experiment

		Attempted Tasks		**Completed tasks**	
Institution	**Statistical Metric**	**Experimental Group**	**Control Group**	**Experimental Group**	**Control Group**
KeMU	***M***	2.46	1.82	1.69	0.36
	SD	0.97	0.60	1.03	0.50
	t	*t* (20) = 1.8		*t* (18) = 4.10	
	p	**0.03**		**0.0003**	
JKUAT	***M***	3.58	2.36	2.50	0.86
	SD	1.56	1.41	1.87	1.19
	t	*t* (46) = 2.82		*t* (39) = 3.59	
	p	**0.004**		**0.0004**	

4.2.1 Discussion: Task Success in the Second Experiment

The experiment sessions at KeMU and JKUAT both resulted in a significantly higher number of attempted tasks in the experimental group than in the control group. Similarly, both experiment sessions resulted in a significantly higher number of completed tasks in the experimental groups than in the control groups. The results from both KeMU and JKUAT indicate that the scaffolding techniques enabled learners to attempt and complete more programming tasks than the non-scaffolded environment.

At KeMU, only one learner from both groups was able to attempt any of the last three tasks. At JKUAT, fewer learners were able to attempt the last three tasks than the first three. At the end of the experiment session, learners were asked to indicate reasons why they could not attempt all the tasks. Collectively, the reasons that the learners gave are: ‘time could not allow’, ‘the tasks were a bit challenging for me’, ‘I have very limited Java knowledge’, and ‘I came late to the session so I had limited time to attempt all.’

These reasons indicate that with more time and with sufficient programming background, learners may be able to attempt, and perhaps complete, more programming tasks using the scaffolding techniques.

4.3 Time-on-task – First Experiment

Time-on-task was measured in four ways: (i) time on incomplete tasks; (ii) time on complete tasks; (iii) total time on tasks; and (iv) comparison of times on complete tasks from one task to another. Table 4 shows the statistical results for all complete and incomplete tasks in the first experiment at UWC and JKUAT.

There was no significant difference in mean completion time between the experimental group and the control group at UWC. With a p-value of 0.34, the first null hypothesis cannot be rejected. Therefore, the mean completion time in the experimental group is not less than the mean completion time in the control group. In contrast, there was a significant difference between the mean time on incomplete tasks in the experimental group at UWC and the mean time on incomplete tasks in the control group. With a p-value of 0.003, the second null hypothesis is rejected in favor of the second alternate hypothesis. Therefore, the mean time on incomplete tasks in the experimental group is less than the mean time on incomplete tasks in the control group.

There was no significant difference in mean completion time between the experimental group and the mean completion time in control group at JKUAT. With a p-value of 0.49, the first null hypothesis cannot be rejected. Therefore, the mean completion time in the experimental group is not less than the mean completion time in the control group. Similarly, there was no significant difference between the mean time on incomplete tasks in the experimental group at JKUAT and the mean time on incomplete tasks in the control group. With a p-value of 0.37, the second null hypothesis cannot be rejected. Therefore, the mean time on incomplete tasks in the experimental group is not less than the mean time on incomplete tasks in the control group.

Figure 7 shows the time-on-task for each of the completed tasks in the experimental and control groups at UWC. These three tasks are considered because they were the ones completed by more than one learner in either of the groups. Figure 8 shows the time distributions for the first two completed tasks in the experimental and control groups at JKUAT. These two tasks are considered because they were the ones completed by more than one learner in both groups. Table 5 shows the statistical results for the first two tasks at UWC and JKUAT. The first two tasks are considered because they were the ones completed in both the control and experimental groups at UWC and JKUAT.

Table 4. Statistical time-on-task results for all complete and incomplete tasks in the first Experiment

		Completed tasks		**Incomplete tasks**	
Institution	**Statistical Metric**	**Experimental**	**Control**	**Experimental**	**Control**
UWC	***M***	20.76	22.18	7.51	21.70
	SD	9.99	8.05	6.34	12.74
	t	$t(18) = 0.41$		$t(15) = -3.27$	
	p	**0.34**		**0.003**	
JKUAT	***M***	22.46	22.44	34.00	30.86
	SD	17.77	13.00	28.27	21.74
	t	$t(26) = 0.004$		$t(26) = 0.34$	
	p	**0.49**		**0.37**	

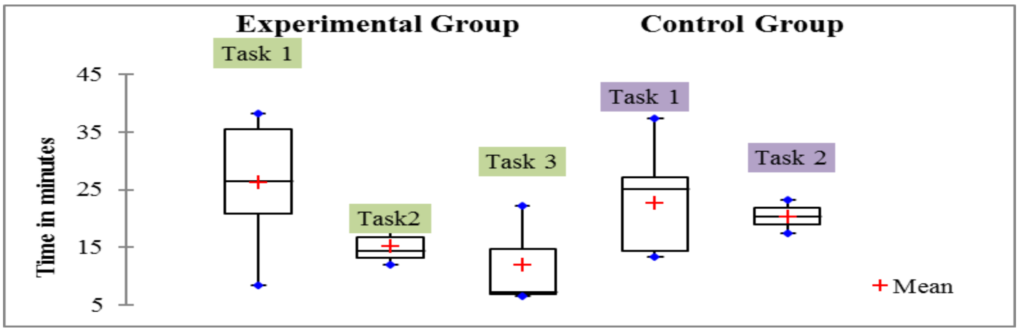

Figure 7. Box plot showing time on completed tasks per-task in the Experimental and Control group at UWC, Experiment 1

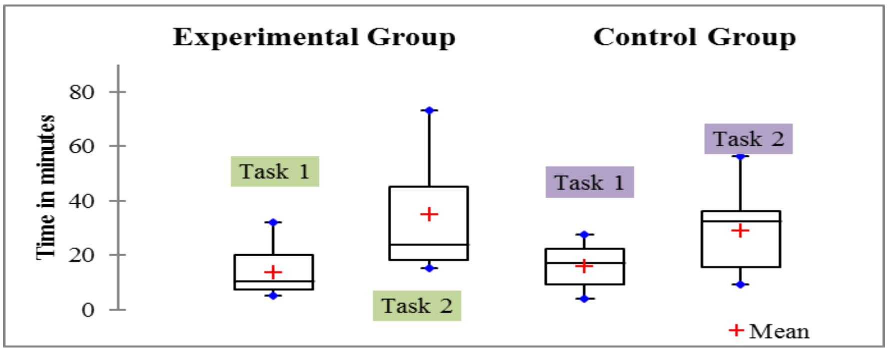

Figure 8. Box plot showing time on completed tasks per-task for Experimental and Control group at JKUAT, Experiment 1

Table 5. Statistical time-on-task results for the first two tasks in the first Experiment

		Task 1		**Task 2**	
Institution	**Statistical Metric**	**Experimental**	**Control**	**Experimental**	**Control**
UWC	***M***	26.2	22.71	15.61	20.33
	SD	9.90	9.07	2.99	4.12
	t	$t(14) = 0.78$		$t(1) = -1.63$	
	p	**0.22**		**0.17**	
JKUAT	***M***	13.92	15.86	35.11	28.96
	SD	8.90	8.25	24.31	14.89
	t	$t(16) = -0.48$		$t(5) = 0.52$	
	p	**0.32**		**0.31**	

At UWC, there was no significant difference in mean completion time for the first task in the experimental group and the first task in the control group. Similarly, there was no significant difference in mean completion time for the second task in the experimental group and the second task in the control group. With both p-values > 0.05, the first null hypothesis cannot be rejected. Therefore, the mean completion time per task in the experimental group is not less than the mean completion time per task in the control group.

At UWC, learners in the experimental group spent a significantly shorter time on the second task than the first task. For example, there was a significant difference in mean completion time on the second task (M = 15.61, SD = 2.99) in comparison to the first task (M = 26.2, SD = 9.90), t (14) = 3.57, p = 0.002. On the other hand, the control group showed a non-significant difference in mean completion time in the second task (M = 20.33, SD = 4.12) in comparison to the first task (M = 22.71, SD = 9.07), t (4) = 0.53, p = 0.31. Therefore, the mean completion time for subsequent tasks after the first in the experimental group is less than the mean completion time for subsequent tasks after the first in the control group.

At JKUAT, there was no significant difference in mean completion time for the first task between the experimental group and the first task in the control group. Similarly, there was no significant difference in mean completion time for the second task in the experimental group and the second task in the control group. With both p-values > 0.05, the first null hypothesis cannot be rejected. Therefore, the mean completion time per task in the experimental group is not less than the mean completion time per task in the control group.

At JKUAT, there was no significant difference between the mean completion time in the first task in the experimental group (M = 13.92, SD = 8.90) and the mean completion time in the second task in the experimental group (M = 35.11, SD = 24.31), t (5) = -1.88, p = 0.06. On the other hand, there was a significant difference between the mean completion time in the first task in the control group (M = 15.86, SD = 8.25) and the mean completion time in the second task in the control group (M = 28.96, SD = 14.89) t (16) = -2.49, p = 0.01. Therefore, the mean completion time for subsequent tasks after the first in the experimental group is not less than the mean completion time for subsequent tasks after the first in the control group.

4.3.1 Discussion: Time-on-task in the First Experiment

Results from UWC and JKUAT indicate that the mean completion time in the experimental group is not less than the mean completion time in the control group. This is supported by results that indicate that the mean completion time per task in the experimental group is not less than the mean completion time per task in the control group. This shows that the scaffolding techniques did not enable faster completion times than the non-scaffolded environment. Further, as reported in the results for task success for JKUAT, the learners in the control group edited previously completed programs as opposed to starting programs from scratch. This shows that for the second experiment, learners in the control group had an advantage over learners in the experimental group.

Results from UWC indicate that the mean time on incomplete tasks in the experimental group is less than the mean time on incomplete tasks in the control group. This shows that learners using the scaffolding techniques were able to reach failure states quicker and could move on to other tasks, as opposed to learners in the control group who spent longer on unsuccessful tasks. However, results from JKUAT indicate that the mean time on incomplete tasks in the experimental group is not less than the mean time on incomplete tasks in the control group. This shows that the scaffolding techniques did not enable learners to reach failure states quicker than the non-scaffolded environment.

Lastly, results from UWC indicate that learners in the experimental group spent significantly shorter times in subsequent tasks after the first task. In comparison, learners in the control group did not show this trend. This indicates the learnability of the scaffolded environment. However, results from JKUAT indicate that there was no significant difference between the mean completion time in the first task in the experimental group and subsequent tasks. On the other hand, learners in the control group took a significantly longer time on the second task than on the first task. This shows that the scaffolding techniques did not enable faster completion times in subsequent tasks after the first.

4.4 Time-on-Task – Second Experiment

Table 6 shows the statistical results for all complete and incomplete tasks in the third experiment. There was no significant difference in mean completion time between the experimental group at KeMU and the control group. With a p-value of 0.22, the first null hypothesis cannot be rejected. Therefore, the mean completion time in the experimental group is not less than the mean completion time in the control group. There was no significant difference between the mean time on incomplete tasks in the experimental group at KeMU and the mean time on incomplete tasks in the control group. With a p-value of 0.37, the second null hypothesis cannot be rejected. Therefore, the mean time on incomplete tasks in the experimental group is not faster than the mean time on incomplete tasks in the control group.

There was no significant difference in mean completion time at JKUAT between the experimental group and the control group. With a p-value of 0.09, the first null hypothesis cannot be rejected. Therefore, the mean completion time in the experimental group is not faster than the mean completion time in the control group. There was a significant difference between the mean time on all incomplete tasks in the experimental group at JKUAT and the mean time on all incomplete tasks in the control group. With a p-value of 0.04, the second null hypothesis is rejected in favor of the second alternate hypothesis. Therefore, the mean time on incomplete tasks in the experimental group is faster than the mean time on incomplete tasks in the control group.

Table 6. Statistical time-on-task results for all complete and incomplete tasks in Experiment 2

		Completed tasks		**Incomplete tasks**	
Institution	**Statistical Metric**	**Experimental**	**Control**	**Experimental**	**Control**
KeMU	***M***	20.88	27.36	30.65	33.39
	SD	15.01	13.59	21.41	16.77
	t	t (4) = 0.86		t (16) = -3.44	
	p	**0.22**		**0.37**	
JKUAT	***M***	15.82	18.75	22.84	31.39
	SD	11.15	7.51	17.66	19.92
	t	t (52) = 1.34		t (57) = -1.78	
	p	**0.09**		**0.04**	

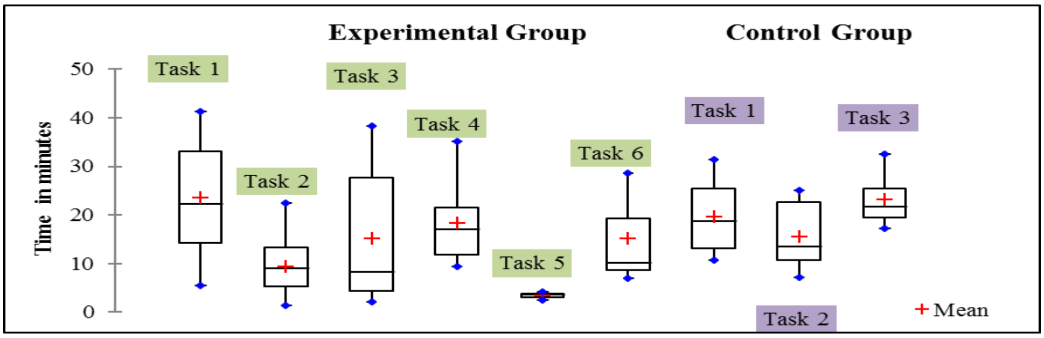

Figure 9. Box plot showing task completion rates across completed tasks for Experimental and Control groups at JKUAT, Experiment 2

Table 7. Statistical time-on-task results per completed task in the second Experiment at JKUAT

		Task 1		**Task 2**		**Task 3**	
Institution	**Statistical Metric**	**Experimental**	**Control**	**Experimental**	**Control**	**Experimental**	**Control**
JKUAT	***M***	23.53	19.56	9.42	15.56	15.16	24.25
	SD	10.69	7.79	5.59	6.99	13.00	5.66
	t	t (21) = 1.09		t (11) = -2.18		t (12) = -1.93	
	p	**0.14**		**0.03**		**0.04**	

Figure 9 shows the time-on-task for each of the completed tasks in the experimental and control groups at JKUAT. JKUAT's data is used to show the time-on-task because they contained the group that completed the most number of tasks. Table 7 shows the statistical results per completed task at KeMU and JKUAT in the second experiment.

There was no significant difference in the mean completion time in the first task in the experimental group and the mean completion time in the first task in control group. With a p- value of 0.14, the first null hypothesis cannot be rejected. Therefore, the mean completion time for the first task in the experimental group is less than the mean completion time for the first task in the control group. However, there was a significant difference in the mean completion time in the second task in the experimental group and the mean completion time in the second task in the control group. Similarly, there was a significant difference in the mean completion time in the third task in the experimental group and the third task in the control group. With both p-values < 0.05 in the second and third tasks, the first null hypothesis is rejected for these tasks in favor of the alternate hypothesis. Therefore, the mean completion time for the second task in the experimental group is less than the mean completion time for the second task in the control group. Similarly, the mean completion time for the third task in the experimental group is less than the mean completion time for the third task in the control group.

4.4.1 Discussion: Time-on-task in the Second Experiment

Results from KeMU and JKUAT indicate that the mean completion time in the experimental group is not less than the mean completion time in the control group. This is supported by results from KeMU that indicate that the mean completion time per task in the experimental group is not less than the mean completion time per task in the control group. This shows that the scaffolding techniques did not enable faster mean completion times than the non-scaffolded environment.

However, results from JKUAT indicate that the mean completion times for the second and third tasks in the experimental group are less than the mean completion time for the second and third tasks in the control group. These results indicate that after the initial familiarization with a new environment, learners using the scaffolding techniques were able to complete tasks significantly faster than learners using the non-scaffolded environment. This indicates the learnability of the scaffolded environment.

Lastly, results indicate that the mean time on incomplete tasks in the experimental group is not less than the mean time on incomplete tasks in the control group. This shows that the scaffolding techniques did not enable learners to reach failure states quicker than the non-scaffolded environment.

4.5 Errors – First Experiment

Errors were measured by investigating the number of run-time errors for all the programs in the control and experimental group and the errors that triggered scaffolding techniques that offered support for error detection, only for the experimental group.

Table 8 shows the statistical results on the mean number of errors for all tasks, first task and second tasks in the first experiment. The first analysis was conducted on the mean number of errors for all the tasks. There was a significant difference between the mean number of run-time errors encountered on all the tasks in the experimental group at UWC and the mean number of run-time errors encountered on all the tasks in the control group. With a p-value of 0.0004, the null hypothesis is rejected in favor of the alternate hypothesis. Therefore, the mean number of run-time errors encountered in the experimental group at UWC is lower than the mean number of run-time errors encountered in the control group. On the contrary, there was no significant difference between the mean number of run-time errors encountered on all the tasks in the experimental group at JKUAT and the mean number of run-time errors encountered on all the tasks in the control group. With a p-value of 0.41, the null hypothesis cannot be rejected. Therefore, the mean number of run-time errors encountered in the experimental group at JKUAT is not lower than the mean number of run-time errors encountered in the control group.

Table 8. Statistical results on the mean number of errors for all tasks, first task, and second task at UWC and JKUAT in the first Experiment

		All tasks		**Task 1**		**Task2**		**Task 3**	
Institution	**Statistical Metric**	**Experi mental**	**Control**	**Experi mental**	**Control**	**Expei mental**	**Control**	**Experi mental**	**Control**
UWC	***M***	1.93	6.41	1	7.61	3	3		
	SD	1.43	4.38	0	4.33	1.41	2.64		
	t	$t(20) = -3.97$		$t\ (12) = -5.50$		$t\ (3) = -5.50$			
	p	**0.0004**		***p* = 0.00006**		***p* = 0.05**			
JKUAT	***M***	5.5	5.11	4	3.55	7.57	5.66	3	5.66
	SD	5.70	3.61	3.60	2.00	7.36	4.37	2.82	3,91
	t	$t(17) = 0.23$		$t\ (2) = 0.20$		$t\ (9) = 0.62$		$t\ (2) = -1.16$	
	p	**0.41**		**0.42**		**0.27**		**0.18**	

A second analysis was conducted on the mean number of run-time errors per task, as shown in Table 8. There was a significant difference between the mean number of run-time errors encountered on the first task in the experimental group at UWC and the mean number of run-time errors encountered on the first task in the control group. With a p-value of 0.00006, the null hypothesis is rejected in favor of the alternate hypothesis. Therefore, the mean number of run-time errors encountered in the experimental group at UWC is lower than the mean number of run-time errors encountered in the control group. However, there was no significant difference between the mean number of run-time errors encountered in the second task in the experimental group at UWC and the mean number of run-time errors encountered on the second task in the control group. With a p-value of 0.05, the null hypothesis cannot be rejected. Therefore, the mean number of run-time errors encountered in the experimental group at UWC is not lower than the mean number of run-time errors encountered in the control group. Statistical analysis was not performed on the third and fourth tasks because these had only one learner each attempting these tasks in the control group.

At JKUAT, there was no significant difference between the mean number of run-time errors encountered in the first three tasks in the experimental group and the mean number of run-time errors encountered in the first three tasks in the control group. With all p-values > 0.05, the null hypothesis cannot be rejected for these tasks. Therefore, the mean number of run-time errors encountered in the experimental group is not lower than the mean number of run-time errors encountered in the control group.

A further analysis was conducted on UWC's and JKUAT's experimental group data to investigate which parts of the programs that the error prompts occurred. The results revealed that most of the error prompts occurred in the main class chunk. Examples of the error prompts displayed to the learners are when the main class does not begin with an upper case letter (Figure 10 in italics) and some in the main method where a learner did not correctly complete the for-loop declaration (Figure 11 in italics). Further analysis on the data from the first experiment revealed that learners in the control group had syntactical errors that could be reduced by scaffolding techniques found in the scaffolded environment. For example, Figure

12 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 the scaffolded environment, a scaffolding technique that provides default statements such as ‘System.out.println()’ reduces the occurrence of such syntax errors. It was noted that none of the programs written by learners in the control group contained header comments (as can be seen from Figure 12); this is as opposed to the scaffolded environment that guides the learner to create header comments.

Main Class Button Pre
Main Class Child
Started at Basic Interface
Editor
Main class Error classname does not begin with an upper ca

Figure 10. Error prompt showing incorrect creation of the main class

Main Method Button Pre
Main Method Child
Editor
System.out.println selected from statement dialog
for-loop selected from statement dialog
Main Method Error: A for loop syntax doesnt have two commas within the declaration

Figure 11. Error prompt showing incorrect completion of the for-loop

```
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 12. A program showing the Keywords ‘String’ and ‘System’ written in lower case ‘s’ (in bold)

4.6 Errors- Second Experiment

For the second experiment, JKUAT is used to illustrate the results on errors since it had the highest number of participants in both the control and the experimental groups. Table 9 shows the statistical results on the mean number of errors for all tasks, first, second and third tasks in the second experiment at JKUAT. There was a significant difference between the mean number of run-time errors encountered in all the tasks in the experimental group at JKUAT and the mean number of run-time errors encountered on all the tasks in the control group. With a p-value of 0.0003, the null hypothesis is rejected in favor of the alternate hypothesis. Therefore, the mean number of run-time errors encountered in the experimental group at is lower than the mean number of run-time errors encountered in the control group. Further, there was a significant difference between the mean number of run-time errors encountered in the first, second and third tasks in the experimental group and the mean number of run-time errors encountered in these tasks in the control group. With p-values < 0.05, the null hypothesis is rejected in favor of the alternate hypothesis. Therefore, the mean number of run-time errors encountered in the experimental group for these tasks is lower than the mean number of run-time errors encountered in the control group.

A further analysis was conducted on JKUAT’s experimental group data to investigate where most of the error prompts occurred. Most of the error prompts were encountered in the first program, at two error prompts on average per learner. The additional analysis revealed that most of the error prompts were encountered within the main class chunk. Examples of the error prompts displayed to the learners are the main class containing special characters (Figure 13 in italics) and some in the main method where a learner wrote public, void or return statement within the main method (Figure 14 in italics).

Table 9. Statistical results on the mean number of errors for all tasks, first, second and third tasks at JKUAT in the second experiment

		All tasks		**Task 1**		**Task2**		**Task 3**	
	Statistical metric	**Experimental**	**Control**	**Experimental**	**Control**	**Experimental**	**Control**	**Experimental**	**Control**
JKUAT	***M***	1.78	5.02	2.05	5.83	1.6	3.83	1.75	7
	SD	1.08	5.39	1.16	7.03	0.91	3.15	1.30	3.42
	t	t (40) = -3.64		t (18) = -2.24		t (14) = -2.28		t (4) = -3.97	
	p	**0.0003**		**0.018**		**0.019**		**0.008**	

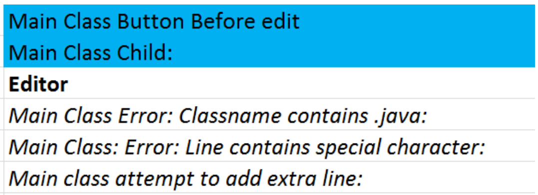

Figure 13. Error prompts encountered within the main class in italics

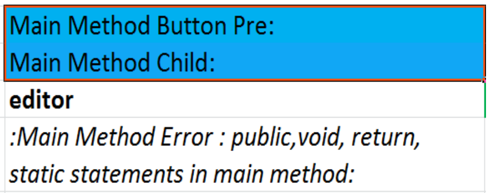

Figure 14. Error prompts encountered within the main method in italics

4.6.1 Discussion: Error Results from First and Second Experiments

Of the three experiment sessions, two resulted in a significantly lower mean number of errors across all the tasks in the experimental group than in the control group. Further, the first task at UWC (first experiment) and the first three tasks at JKUAT (second experiment) resulted in a significantly lower mean number of errors in the experimental group than in the control group. The results indicate that scaffolding techniques may lead to fewer run-time errors. Further, additional analyses indicate that the scaffolding techniques may capture some syntactical errors that a non-scaffolded environment may not.

4.7 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.’

5. CONCLUSION

This paper has reported on results of an evaluation with 142 learners of a Java programming course in 3 universities in experimental and control groups. The proposition of this research was that programming environments on mobile phones could include scaffolding techniques that are specifically designed for mobile phones, and designed based on learners’ needs. To address this proposition, one research question was posed: What is the effect on learners of using the theoretically-derived scaffolding techniques to construct Java programs on a mobile phone? This section presents answers, based on the results, to the sub-questions that were posed to address the research question.

i. Scaffolding techniques enable learners to attempt and complete more programming tasks than a non-scaffolded environment.
ii. The scaffolding techniques do not enable faster average task completion times than a non-scaffolded environment. However, after the initial familiarization with the scaffolded environment, the scaffolding techniques may enable faster completion of tasks than a non-scaffolded environment..
iii. The scaffolding techniques may lead to fewer run-time errors. Further, the scaffolding techniques capture some syntactical errors that a non-scaffolded environment may not.
iv. Learners using the scaffolding techniques spend shorter times in subsequent tasks after the previous tasks.

These results indicate that specifically designed scaffolding techniques for mobile phones can enable learners to construct programs on a mobile phone and meet learners’ needs. Further, learners indicated that they found the scaffolding techniques useful in supporting construction of programs on a mobile phone.

5.1 Limitations of the Study and Directions for Future Work

In this study, the emphasis was on providing scaffolding techniques intended to be used by learners who were just beginning to learn programming using Java. Therefore, they were not used to create complex or high-level programs. Hence, the simplicity of the programs used in the study may be limiting. Future work will provide students with more complex tasks such as those that have multiple methods or controlled loops. In relation to this, future work will study the limitations of the interface encountered when tackling more complex and larger programs and if and how these might influence the design of additional scaffolding techniques.

The choice of Android as an implementation platform means that only specific phones could be used during the experiments. Further, this means that users of other platforms cannot use the application. Further, there are other limitations of mobile phones, such as limited memory, that were not considered. This study focused on the limitations of small keypads and small screens.

There is usually a variance in the ability of learners in an introductory programming class. This study did not test the knowledge of the learners prior to conducting the experiments. Future work will conduct a pre-experiment test to determine learners' programming knowledge in order to better place them in appropriate groups with similar abilities.

The t-test was the only statistical metric used to compare data between the control and experimental groups. This might not have been adequate. Hence, future work will look at alternative statistical tests such as the Wilcoxon test. Further, future work will also measure compile-time errors encountered during the program creation on the mobile phone.

Learners in the control group were not asked on their reflections on the use of a mobile phone to construct programs. Hence, 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. Another direction for future work is to conduct a comparative study between the use of the scaffolded environment on the mobile phone and a desktop IDE. Also, a further study will investigate how the learners' experience with the scaffolded environment on the mobile phone transfer to desktop IDEs.

Finally, this research did not evaluate the long-term learning impact of the use of the scaffolding techniques on the eventual performance of students in their programming course, say at the end of the term. This was not evaluated because learners were already exposed to other learning resources and tools for programming and it would have been difficult to determine whether the use of the scaffolding techniques is what directly influenced their eventual success or failure in programming. Nevertheless, given more time and resources, such a long-term study is possible.

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.

Apiola, M. & Tedre, M., 2011. Towards a contextualized pedagogy for programming education in Tanzania. In *IEEE Africon '11*. IEEE, pp. 1–6. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6072010 [Accessed May 31, 2014].

Dann, W.P., Cooper, S. & Pausch, R., 2011. Learning to Program with Alice. Available at: http://dl.acm.org/citation.cfm?id=2011893 [Accessed January 6, 2015].

Elliott, A.C. & Woodward, W.A., 2007. *Statistical Analysis Quick Reference Guidebook: With SPSS Examples*, SAGE Publications. Available at: http://books.google.com/books?hl=en&lr=&id=SOsX0IbNxeIC&pgis=1 [Accessed December 9, 2014].

Guzdial, M., 2015. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*, Morgan & Claypool Publishers. Available at: http://www.morganclaypool.com/doi/abs/10.2200/S00684ED1V01Y201511HCI033 [Accessed March 28, 2016].

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].

Harmon, M., 2011. *t-Tests in Excel - The Excel Statistical Master*, Mark Harmon. Available at: http://books.google.com/books?hl=en&lr=&id=C1OHSbQUvAsC&pgis=1 [Accessed November 25, 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].

Luchini, K. et al., 2002. Scaffolding in the small. In *CHI '02 extended abstracts on Human factors in computing systems - CHI '02*. New York, New York, USA: ACM Press, p. 792. Available at: http://dl.acm.org/citation.cfm?id=506443.506600 [Accessed February 1, 2015].

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].

Pew Research Center, 2015. Internet Seen as Positive Influence on Education but Negative on Morality in Emerging and Developing Nations. Available at: http://www.pewglobal.org/2015/03/19/internet-seen-as-positive-influence-on-education-but-negative-influence-on-morality-in-emerging-and-developing-nations/ [Accessed June 5, 2016].

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].

Quintana, C. et al., 2004. A Scaffolding Design Framework for Software to Support Science Inquiry. *Journal of the Learning Sciences*, 13(3), pp.337–386. Available at: http://dx.doi.org/10.1207/s15327809jls1303_4 [Accessed February 4, 2014].

Sauro, J. & Lewis, J.R., 2012. *Quantifying the User Experience*, Elsevier. Available at: http://www.sciencedirect.com/science/article/pii/B9780123849687000023 [Accessed October 26, 2014].

Soloway, E. et al., 1996. Learning theory in practice: case studies of learner-centered design. In *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*. New York, New York, USA: ACM Press, pp. 189–196. Available at: http://dl.acm.org/ft_gateway.cfm?id=238476&type=html [Accessed May 31, 2014].

Sphere Research Labs, 2010. Ideone TM API. , pp.1–11.

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].

Traxler, J., 2011. Learning with Mobile Devices Somewhere Near the Bottom of the Pyramid « Educational Technology Debate. Available at: http://edutechdebate.org/affordable-technology/learning-with-mobile-devices-somewhere-near-the-bottom-of-the-pyramid/ [Accessed March 19, 2015].

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].

Watson, C. & Li, F.W.B., 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14*. New York, New York, USA: ACM Press, pp. 39–44. Available at: http://dl.acm.org/citation.cfm?id=2591708.2591749 [Accessed January 14, 2015].

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].