INITIAL EVALUATION OF A MOBILE SCAFFOLDING APPLICATION THAT SEEKS TO SUPPORT NOVICE LEARNERS OF PROGRAMMING

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 aim of this paper is to explore the use of an application that scaffolds the constructions of programs on a mobile device. The application was developed to support novice learners of programming outside the classroom. This paper reports on results of a first experiment conducted to evaluate the mobile application. The main research questions are: (i) whether the use of the application is effective in supporting construction of programs on a mobile device; and (ii) how the learners experienced the use of the mobile application. Data was collected by task completion, video and audio recording, and a questionnaire. A total of 18 first-year learners of programming from two African universities took part in the experiment by participating in focus groups. Almost two thirds of the learners completed two out of three programming exercises using the mobile application, with all the learners completing the first program. The results of the study suggest that the students found the mobile application useful, as evident from high rating of its features. The results also consisted of feedback from the learners on features that would make the application more usable. The findings suggest that the use of a mobile scaffolding application may support novice learners of programming outside the classroom. The outcomes of these results lead to a clearer understanding of how to design a mobile application that scaffolds the construction of programs on a mobile device.

KEYWORDS

Mobile, Evaluation, Programming, Support, Scaffolding, Novice Learner.

1. INTRODUCTION

Difficulties encountered while learning computer programming are a universal problem. There have been numerous attempts to tackle these difficulties (for example, (Apiola et al., 2011) (Lahtinen et al., 2005)); but these challenges still remain. Hence, effective instructional strategies and optimal learner support mechanisms should be developed, to provide learners of computer programming with the optimal learning environment that they need (Mow, 2008).

These difficulties show that some programming skills that novice learners need are beyond their abilities. Scaffolding refers to support provided so that the learner can engage in activities that would otherwise be beyond their abilities (Jackson et al., 1998). Providing this support outside the classroom could contribute to tackling learning difficulties. Support is emphasized upon because any such support should be additional to the learners' classroom learning, and not a replacement. Providing support outside the classroom aims to make the most of the resources available to support learning, by making their provision more flexible, open and responsive to the needs of individual learners (Bentley, 2012).

The ubiquity of mobile phones provides an opportunity to use them as a resource to support learners beyond the classroom. In addition, mobile phones can be used by learners in resource-constrained environments who own a mobile device but cannot access PCs outside the classroom. Mobile phones can also be used when and where using a PC would be inconvenient.

In order to exploit the ubiquity of mobile phones to provide support to novice learners of programming, an application was developed that scaffolds the construction of Java programs on a mobile device (Mbogo et al., 2013). This application was developed based on design decisions drawn from learning theories,

requirements gathered and a resulting scaffolding framework. For the sake of illustration in this paper, this mobile application will be referred to as *ScafFold*.

ScafFold is a mobile application developed for the Android platform, aimed at supporting learning of programming by scaffolding the construction of Java programs on the mobile device. Java was chosen as the language for constructing the programs because it is commonly taught in undergraduate courses. Android was selected as the platform of implementation because it is open source, and it also has an 80% market share among smartphone users (Dara, 2013). ScafFold runs on Android version 2.2 upward. The current version is for evaluation purposes only and it is not hosted on a site for download.

ScafFold provides scaffolding by: representing a program in five chunks that represent parts of a Java program; restricting a learner to complete a program in a certain order; enabling construction of a program one chunk at a time; providing instructions, default code, steps, hints, examples and error prompts where appropriate; and fading the scaffolds as the learner progresses from one successfully completed and compiled program, to the next. Figure 1 shows the main interface of Scaffold, which shows the five chunks and a step provided. Figure 2 shows the main class chunk complete (in green), and the header expanded to reveal some automatically generated code. Figure 3 shows an example of an error prompt displayed if the class name is completed inappropriately.

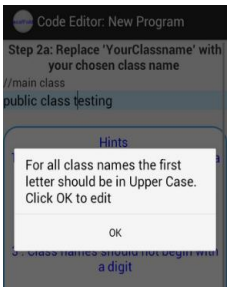

Figure 1. Main interface **Figure 2. Main class created** **Figure 3. Error prompt**

This paper reports on results of a first experiment conducted to evaluate ScafFold. The evaluation was based on a three-level framework (Vavoula & Sharples, 2009), and sought to answer the research questions:

1. Is the use of the application effective in supporting construction of programs on a mobile device?
2. What were the learners' experiences in using the mobile application?

In attempting to answer the research questions above, the following parameters were evaluated: usability of the application; desirability of the features of the application as rated by the learners; and user experience while constructing programs using the mobile application.

In summary, the contribution of this paper is twofold: Presentation of findings from evaluation of a mobile scaffolding application; and Illustration that a development environment with adaptable scaffolding can improve the effectiveness of constructing programs on a mobile device.

The rest of the paper is organized as follows: Section 2 describes related work; Section 3 describes the study methodology, including the e-survey methodology, focus groups and task list; Section 4 briefly discusses the evaluation framework and presents the findings from the survey; and Section 5 concludes the paper.

2. RELATED WORK

The most relevant study in terms of a mobile intervention is TouchDevelop, an application that provides a programming environment, language and a code editor (Tillmann et al., 2012). This mobile application was built for the Windows platform, and employs a semi-structured code editor, which allows for pre-selection of statement kinds and expression tokens. Semi-structured code reduces the cognitive load on learners in memorizing statements and expressions that they would need. TouchDevelop also shows a listing of a complete program. Showing a full program would enable a learner to relate the program part they are working on with the whole view. TouchDevelop does not employ the use of chunks to display a program in parts, an approach that the work described in this paper takes. The TouchDevelop study evaluated the

application with high school and 8th grade learners. The results of the study indicate that a programing environment that runs on a mobile device has the potential to dramatically reduce the technical learning overhead. This paper identifies with these results as a motivation to provide mobile support outside the classroom.

The most relevant study in terms of application of the scaffolding theory is a PC-based application, known as Pseudo-code Development Environment (PDE) (Costelloe, 2004). It allows learners to select the level of problems they wish to work on. The aim is to facilitate guided-learning by doing, and lead the learner to a solution by offering feedback and subdividing the tasks into more manageable sub-tasks. Use of feedback and subdivision of tasks is an approach that the work described in this paper identifies with. PDE also provides a problem category selection such as sequence problems or iteration problems. One way that PDE subdivides a task is by having a learner indicating input data by answering a question on how many inputs a problem needs. PDE is suited for creation of pseudo-code where specification of input could be necessary. However, in creation of a program, a learner does not need to specify input prior to writing the program. Further, since PDE was tailored for pseudo-code creation, it did not make provision for running and compilation of a program.

To summarize, the study in this paper is related to a number of findings and results that have appeared in related contexts. The contribution is not only to report the findings from an initial evaluation of the mobile application, but also to show that a development environment with adaptable scaffolding can improve the effectiveness of constructing programs on a mobile device.

3. STUDY METHODOLOGY

3.1 Study design

The study was conducted in two Universities: University of Cape Town (UCT) and University of Western Cape (UWC). The two universities were selected for this study due to their convenience in terms of having established contacts. A total of 18 first-year learners of programming from the two universities participated in the experiment: 8 from UCT and 10 from UWC. 17 of the learners studied Computer Science, and 1 student studied Electrical and Computer Engineering; all were at Bachelors level.

Prior to starting the experiments, ethical clearance was obtained from the participating universities. Prior to participating in the experiment, all the learners were briefed on the purpose of the research and were requested to complete a consent form that declared: not having used the mobile application before; voluntary participation; freedom to withdraw from the experiment at any time; consent to use audio/video/image recording; and use of their anonymised feedback. The learners at UCT were given incentives of R50, while the learners at UWC were provided with lunch.

A multi-method approach was adopted in the study in order to collect as much data as possible. The study methodology included the following: the learners first participated in focus groups, which consisted of 2 to 10 participants at a time; in the focus group, a task list was issued ,which consisted of programming exercises to be completed using the mobile application; during the focus group, video and image recordings were taken; after the focus group, an electronic survey (e-survey) was issued, which consisted of a questionnaire that collected demographic information and user feedback about the application. All the learners were issued with Android phones to use during the experiment.

3.2 The e-survey methodology

The e-survey methodology has the advantages of decreased cost, faster response times and increased response rates (Lazar & Preece, 1999). The e-survey methodology was used for these reasons and was designed using Limesurvey[1]. The intent of the survey was clearly outlined in the introduction of the survey, enabling well-informed participation and consent. The participants' privacy and confidentiality was ensured by not asking for personal information such as names or identification numbers. A computer was provided at

[1] http://www.limesurvey.org/

the venue of the focus groups for completion of the questionnaire at the end of the session. A total of 16 learners completed the questionnaire. 2 of the learners could not complete the questionnaire due to time constraints as they were attending a class session immediately after the focus group. The questionnaire had three sections: demographic information; interface usability; and user experience.

3.3 Focus groups methodology

Focus group participants were recruited through random sampling after announcements in first-year programming classes. The participants volunteered for the experiment. The focus group approach is defined as a research technique that collects data through group interaction on a topic determined by the researcher, and involves a group of participants and one or more moderators (Colm et al., 2011). For the evaluation of an artifact design, an exploratory focus group studies the artifact to propose improvements in the design (Tremblay et al., 2010). The artifact in this study is the mobile application. At UCT, the learners participated in 3 1-hour long focus groups in groups of 3, 2 and 3 learners respectively. At UWC, all the 10 learners participated in a single focus group during a 1-hour lunch break session. In both locations, all the learners were provided with mobile phones to use for completing a programming task, which is described in the next section.

3.4 Task completion

The learners were required to complete a task that consisted of three Java programming exercises. A task list was used so that when a user goes through an interface, they are goal-oriented by completing specific tasks. (Lazar et al., 2010). The programming questions that the learners were required to complete are:

1. Write a program called **Testing** that prints the words 'This works!'.
2. a.) Write a program called **Odd** that uses a for-loop to print the odd numbers from 1 to 20.
   b.) Write a program that calculates the area of a room that is 10 meters wide and 25 meters long.
3. Write a program that requires input of your name and outputs it in the format 'Your name is X'.
   Hint: Use the BufferedReader class on the mobile application to accept user input.

Some of the learners completed 2 a.), while others completed 2 b.). This was by choice of the learners on instruction that they could complete either.

3.5 Video and image recording

3 students were recorded while they were completing one of the programming exercises. The video recordings give insight to some tacit information and interaction with the application. The video camera was close enough to capture the learners' interaction with the application, but not too close to interfere with the interaction. From time to time, still images of the learners' mobile screenshots were captured. The video and image recordings were analyzed alongside the feedback from the questionnaire, as described in the evaluation section.

4 EVALUATION AND FINDINGS

A three-level evaluation framework was used (Vavoula & Sharples, 2009) to evaluate the mobile application. These levels are Micro, Meso and Macro levels. The *micro* level evaluates the usability of the application, and seeks to find out if the application is designed in such a way that it is usable while constructing programs on the mobile device. A questionnaire was used as the data collection mechanism for the micro level. The *meso* level evaluates the user experience and seeks to find out: if the use of the application is effective in supporting the construction of programs on a mobile device; and learners' cited experiences while using the application. Task completion in the focus group, video and image recordings, and a questionnaire were used as data collection methods for the meso level. The *macro* level evaluates the impact of the application on learning practices, and also finds out the learners' experiences while using the application, and if the application improves their learning processes. This study will evaluate findings at the micro and meso levels.

The macro level is left for future evaluations, which will involve experiments over a longer period of time. Using this framework, the findings are grouped into three sections: usability; desirability of the features of the application; and user experience.

4.2 Usability

Evaluation for usability was measured using the Nielsen heuristics model (Nielsen, 1994). The learners filled a questionnaire at the end of the focus group session that required them to indicate the usability of the application. Table 1 shows 9 heuristic features used to measure usability. Each question was measured using a five-point likert scale that ranged from 'strongly disagree' to 'strongly agree'. The table shows the percentage of the 16 learners who responded to each category.

Generally, the application was cited as mostly useful in terms of interface usability. This is evidenced by 6 of the 9 features having at least 80% combined positive feedback of 'agree' and 'strongly agree'. Clarity of labels, consistency in words and actions, availability of guidance and assistance, and legibility of textual aspects were indicated as the best usability features in the application. However, lack of sufficient feedback and presence of too much information on the editing interface were cited to be the most limiting factors and need improvement. Some of the additional comments (cited verbatim) that the learners' gave in terms of improvement of usability include:

- The instructions were hidden and I didn't know where to look to get the next one. I suggest using a tabbed interface and not a list view.
- Some of the options, such as compile and run, should be made more accessible.
- Sometimes undoing an action was not clear. Doesn't give you the option of undoing fully.
- There is too much information on the coding screen.
- Textual aspects should be distinguished by different colours to be more recognizable.

Table 1. Responses on usability of the mobile application

Heuristic\Scale	Strongly disagree	Disagree	Neither disagree nor agree	Agree	Strongly agree	Combination of Agree & Strongly Agree
Attractiveness	0%	6%	25%	50%	19%	69%
Easy to follow and use	0%	6%	0%	50%	44%	88%
Sufficient feedback	6%	6%	0%	50%	38%	88%
Clear labels	0%	0%	6%	38%	56%	94%
Clear ways of undoing/redoing	0%	6%	25%	25%	44%	69%
Consistent words/actions	0%	0%	6%	69%	25%	94%
Guidance/assistance	6%	0%	0%	44%	50%	94%
No irrelevant information	0%	19%	25%	31%	25%	56%
Legible textual aspects	0%	6%	0%	50%	44%	94%

4.2 Desirability of the features of the application

The learners were asked to indicate the extent to which they agreed that the scaffolding features of the application support the construction of programs on the mobile device. Table 2 shows how the learners rated the different scaffolding features of the application in terms of agreeing and strongly agreeing. The last column gives the combination of these two sets of responses per feature. The features with the highest percentages in the last column are the most perceived to effectively support constructions of programs on a mobile device. Availability of hints, presentation of programs in chunks and provision of steps that enable the user to interact with the application, were indicated by the learners as most desirable. Error prompts and provision of default code were the least desirable features as rated by the learners.

Table 2. How the learners rated the different features of the application in terms of effectiveness in supporting construction of programs on a mobile device

Scaffolding features	Strongly Disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree	Combination of Agree & Strongly Agree
Presentation in chunks	0%	12%	0%	63%	25%	88%
Completion part at a time	0%	13%	6%	38%	43%	81%
Steps to interact with application	0%	0%	12%	63%	25%	88%
Availability of hints	0%	0%	6%	38%	56%	94%
Error prompts	13%	12%	6%	44%	25%	69%
Dialog prompt of options e.g 'System.out.println()',	13%	0%	0%	31%	56%	87%
Provision of default code	0%	19%	19%	31%	31%	62%
Provision of examples	0%	6%	18%	38%	38%	76%
View of full program at any time	0%	0%	19%	38%	43%	81%

4.3 User experience

4.3.1 Task Analysis

All the 18 learners who participated in the survey completed the first programming task. Two-thirds of the participants completed the first two programming exercises. Only two learners managed to complete the 3rd exercises that required usage of the BufferedReader class to accept user input, as shown in Figure 4. Learners indicated that they had not learnt the use of the BufferedReader in class, but had been taught use of the Scanner class for input. This posed a challenge since the Ideone compiler that the application uses only accepts input through the BufferedReader class.

4.3.2 Challenges encountered

All the learners indicated that the application can fit in a novice's learning environment to effectively support construction of programs on a mobile device. However, during the task completion, the following challenges were experienced by some of the learners and also observed:

1. The video recordings showed that the learners hardly scrolled to view information that is not readily visible on the screen. In several instances, learners kept clicking on a non-active button, while the instructions on what to do next were at the bottom of the screen, which would have been visible upon scrolling. This gives an indication to try out different screen interactions like: usage of tabs, as suggested by some learners; and also use of hierarchical views that can be expanded (Churchill & Hedberg, 2008).
2. In some instances, the learners completed a full program all within the class declaration (Figure 4). Upon pressing the back button to go back to the main interface, the learner got a prompt informing them that the class declaration required only one line. This indicates that the application could be improved to provide immediate prohibition of unrequired code, and not wait until the learner attempts to proceed.
3. Despite provision of a dialog box to select some statements to use, some learners opted to ignore the prompt and type the statements on their own. A commonly occurring instance was the provision of a dialog box that provided a choice of preselecting 'System.out.println()', and therefore just required users to fill the required output inside the brackets. But some learners opted out of the dialog box and typed the statement from scratch. Some provision of alerting the learner that they can preselect the statements even if they opt out would be helpful, especially because these pre-selection dialog boxes were some of the highly rated features of the application.

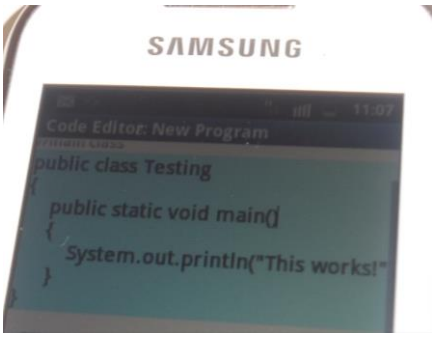

Figure 4. Task 1 created within class declaration

4. A major challenge was the soft key pad covering the nearly three-quarter of the screen while typing. This blocked some of the instructions and hints, and some learners missed these. This was especially so in the code editing screen, but not a major problem in the main interface. One approach to handle this in the code editor would be to use tabs at the top of the screen as opposed to the bottom for hints, examples and instructions. Indeed, a study suggested that scrolling can be reduced by placing navigational features in the fixed place near the top of presented resource, and by placing key information at the top (Jones et al., 1999).
5. In some situations, the wireless network was very slow and hence slowed down the compilation phase. Some of the learners switched to the data network, which turned out to be more reliable than the school wireless networks.

5. CONCLUSION

This paper has reported upon results of an evaluation of a mobile scaffolding application. A majority of the learners completed most of the programming tasks. They indicated that the features of the application effectively support construction of programs on a mobile device. The highly rated features are: hints; steps; presentation of programs in chunks; pre-structured dialogs; completion of one chunk at a time; and view of the full program. Further, the learners indicated that the application is usable due to having clear labels and textual representation, consistency in words and availability of guidance and steps.

However, as literature reveals (Churchill & Hedberg, 2008) and also as pointed out by the learners, mobile phones present several challenges in their use to support learning. Thus, if mobile phones are to be effectively used to support construction of programs on a mobile device, the design and presentation has to be optimized for mobile phones. According to the feedback from the learners and also literature (Churchill & Hedberg, 2008) (Jones et al., 1999), this can be done in several ways: minimize the amount of text on the screen by providing tabs to navigate through the information; make important links such as compile and view of full program more visible, provide hierarchical views of information; create tabs at the top instead of the bottom of the screen to take care of the soft-keypad of touch screens; and enable clear ways of undoing and redoing a task.

Although the findings are encouraging and useful, the study in this paper has certain limitations that require further research. Firstly, the application was developed for the Android platform and this presents a limitation in the number and type of mobile platforms it can be tested on. The application is developed to be used with the Java programming language due to its popularity in first-year undergraduate courses. However, it is acknowledged that not testing it using the other programming languages could be limiting. In addition, the programming exercises used in this initial evaluation are simple and therefore present a limitation in the extent to which the application can be used to support more difficult tasks. A future evaluation will include exercises that cover more advanced tasks. The number of participants in the evaluation was also limited and some key feedback could have been missed. Nevertheless, the principles and feedback obtained from this evaluation can be applied across interventions that target other mobile platforms and programming languages.

The results of the evaluation suggest that the use of a scaffolding application may support construction of programs on a mobile device; this warrants further study. The experience of learners using the application can be made more effective based on user feedback, supported by underlying theory and existing research. These results lead to a clearer understanding of how to effectively design a mobile scaffolding application to support learners outside the classroom.

Current work seeks to implement the feedback obtained from the learners into a second iteration of the mobile application. Future work involves testing and evaluation of the mobile application with a larger number of undergraduate programming learners over a longer period of time.

ACKNOWLEDGEMENT

This study is funded by the Hasso Plattner Institute, and supported by the ICTD laboratory at University of Cape Town. We thank Bill Tucker of UWC and Audrey Mbogho of UCT for allowing access to their learners of programming. We thank all the 18 learners who participated in the experiments.

REFERENCES

Apiola, M., Tedre, M. & Oroma, J.O., 2011. Improving Programming Education in Tanzania: Teachers' and Students' Perceptions. In *Proceedings 41st ASEE/IEEE Frontiers in Education Conference*. Rapid City, 2011.

Bentley, T., 2012. *Learning Beyond the Classroom: Education for a Changing World*. Taylor & Francis.

Churchill, D. & Hedberg, J., 2008. Learning object design considerations for small-screen handheld devices. *Computers and Education* , 50, pp.881-93.

Colm, O., Xiaofeng, W. & Kieran, C., 2011. The use of focus groups in complex and pressurised IS studies and evaluation using Klein & Myers principles for interpretive research. *Information Systems Journal*, 22(3), pp.235-56.

Costelloe, E., 2004. *The Use of a Software Enabled Scaffolding Environmnet to Aid Novice Programmers*. Thesis. University of Dublin.

Dara, K., 2013. *Android dominates 81 percent of world smartphone market*. [Online] (1) Available at: http://news.cnet.com/8301-1035_3-57612057-94/android-dominates-81-percent-of-world-smartphone-market/ [Accessed 28 November 2013].

Jackson, S., Krajcik, J. & Soloway, E., 1998. The design of guided learner-adaptable scaffolding in interactive learning environments. In *CHI '98 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Los Angeles, CA USA, 1998.

Jones, M. et al., 1999. Improving Web interaction on small displays. *Computer Networks*, 31(11-16), pp.1129-37.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H., 2005. A study of the difficulties of novice programmers. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05)*. New York, 2005.

Lazar, J., Feng, J.H. & & Hochheiser, H., 2010. *Research Methods in Human-Computer Interaction*. John Wiley & Sons.

Lazar, J. & Preece, J., 1999. Designing and implementing Web-based surveys. *Journal of Computer Information Systems*, 39(4), pp.63-68.

Mbogo, C., Blake, E. & Suleman, H., 2013. A Mobile Scaffolding Application to Support Learners of Computer Programming. In *ICTD '13 Proceedings of the Sixth International Conference on Information and Communication Technologies and Development*. Cape Town, 2013.

Mow, I.T.C., 2008. Issues and Difficulties in Teaching Novice Computer Programming. In M. Iskander, ed. *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*. pp.119-204.

Nielsen, J., 1994. Heuristic Evaluation. In Nielsen, J. & Mack, R.L. *Usability Inspection Methods*. New York: John Wiley & Sons.

Tillmann, N. et al., 2012. The Future of Teaching Programming is on Mobile Devices. In *ITiCSE'12 Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. Haifa, Israel, 2012.

Tremblay, M.C., Hevner, A.R. & Berndt, D.J., 2010. The Use of Focus Groups in Design Science Research. In *Design Research in Information Systems*. pp.121-43.

Vavoula, G. & Sharples, M., 2009. Meeting the Challenges in Evaluating Mobile Learning: A3-level Evaluation Framework. *International Journal of Mobile and Blended Learning*, pp.54-75.