Teaching Mobile Development with Pair Programming

Mohammed Seyam
Department of Computer Science, Virginia Tech
KnowledgeWorks II, Room 1122
2202 Kraft Dr, Blacksburg, VA 24061
+1 540 449 9415
seyam@vt.edu

D. Scott McCrickard
Department of Computer Science, Virginia Tech
KnowledgeWorks II, Room 1118
2202 Kraft Dr, Blacksburg, VA 24061
+1 540 231 6698
mccricks@cs.vt.edu

ABSTRACT
Pair Programming has demonstrated benefits for education, but unique concerns of mobile software design raise questions about the effectiveness of Pair Programming in this evolving field. This paper probes unique challenges for Pair Programming when used in mobile software design classes, focusing on five mobile design topics: dealing with interface and data management, using camera, handling multi-device connectivity, using sensors and collecting GPS data, and using microphones and speakers. The paper highlights successes and challenges for Pair Programming and mobile applications, concluding with recommendations on building assignments, managing student interaction, and implementing Pair Programming for instructors considering using it in their mobile development classes.

Keywords
Computer Science Education; Pair Programming; Mobile development.

1. INTRODUCTION
Agile methods provide techniques, tools, and practices to help software practitioners develop software quickly and efficiently, valuing working software and face-to-face interactions over plans, processes, and documentation [1; 2]. One of the Agile practices that have been used in educational contexts is Pair Programming (PP), which is an approach that features two developers working on the same development task [3; 4]. One developer (usually called the driver) actually writes code, while the other (the navigator) watches the driver, provides advice, and seeks to grasp the overall picture of the task under development [4]. These two roles are exchanged at regular intervals, and the two developers together become the owners of the resulting product—and also become knowledgeable about the development topic.

This paper explores whether PP would help students in a mobile development course to better understand mobile concepts and enhance the quality of their applications. There are fundamental differences in programming topics for mobile, including a very different user interface style centered on a small screen and finger-based interactions, a greater emphasis on parallelism and asynchronous operations, and a large number of sensors. These fundamentally important topics have not been studied with PP previously, but their challenging nature highlight potential benefits for pair-based learning.

SIGCSE'16, March 02 - 05, 2016, Memphis, TN, USA

ACM 978-1-4503-3685-7/16/03…$15.00
DOI: http://dx.doi.org/10.1145/2839509.2844644

The next section provides background on research that has been done on applying PP in classroom settings. We then show how mobile development differs from traditional web/desktop programming. The paper proceeds with showing how we apply PP in our mobile development class. We conclude the paper with a discussion on the findings in five PP sessions we performed, with a set of guidelines that would help instructors facilitate PP sessions in their own mobile development classes.

2. PAIR PROGRAMMING FOR MOBILE DEVELOPMENT CLASS
During the past two decades, PP has been studied in both industrial and classroom settings. For industrial environments, [5] presents studies that have shown that PP enhanced understandability and maintainability of code and design [6], decreased defect rates [7] and supported knowledge transfer [8].

Since we are interested in PP for classroom settings, we focus on studies that have been performed on students, especially in higher education who are enrolled in Computer Science (CS) or Software Engineering (SE) classes. One of the original studies of PP suggested that it is a promising approach to use as a pedagogical tool due to its capability of increasing learning capacity [9], with foundational studies demonstrating how PP as a pedagogical tool that can help students enhance productivity and learn technology design and programming [10-13]. A review of PP studies shows areas of success with evidence suggesting that PP could enhance enjoyment, increase student's confidence level, reduce workload, improve course completion rate, increase homework submission rate, improve exam performance, and facilitate working efficiently on programming assignments [14].

Since it became popular in the 1990s, PP has been used by developers who work on desktop applications. With the rise of web applications, PP became a popular practice for web development. During the past decade, mobile applications became more mainstream, and several approaches and techniques have been developed to serve the goals of mobile application developers. As a result, CS educators started to incorporate mobile development as a core component of their programming classes. Rahimian and Ramsin [15] show that mobile development has certain issues that differentiate it from desktop/web development related to communication, mobility, portability, standards and protocols, power, storage, and user interfaces. Researchers have tried to come up with modern teaching approaches that can accommodate the special nature of mobile application development. Tigrek and Obadat [16] proposed a pedagogy for teaching smartphone programming, which incorporated PP as a practice to encourage guided teamwork. However, PP was just a supporting practice and not at the core of this pedagogy, and the research did not provide details on the results of implementing PP. A recent research

paper demonstrated teaching programming for smartwatches as part of an undergraduate mobile development class [17]. This research shows the need to consider several extensions while dealing with mobile development, such as wearable devices, different sensors, and multiple platforms. A focus on design education requires careful consideration of user interface issues. Williams et al. [18] presented a study of using PP for teaching Human Computer Interaction (HCI) class, but they did not show the impact of applying PP on the user interface. Our work explores how PP can be applied while developing mobile applications, and what would be the opportunities and challenges associated with students developing mobile applications in pairs.

Based on these characteristics of mobile development, we believed that students can benefit from PP when working on mobile development in-class activities, with three concerns:

1- Multi-screen mobile development environment. Mobile design relies on multiple screens, indeed multiple platforms, to craft a realistic and meaningful design experience. PP must be adapted to address this issue.
2- Connectivity issues. A definitive feature of mobile devices is that they rely on connectivity both with external devices like smartwatches and health sensors as well as internal hardware like GPS and location data, accelerometers, light sensors, and cameras. PP must account for these features.
3- User interface (UI) and user experience (UX) issues. Mobile devices are very different than more traditional desktop, laptop, and web environments, with challenges and opportunities for PP.

To understand how students would react to these concerns, we examined student course work during pair programming sessions of a mobile development class.

3. PAIR PROGRAMMING SESSIONS

To gain a better understanding for the context of developers in a classroom setting, we conducted expert reviews with two pairs of experienced Android developers. Two active walkthrough PP sessions were conducted to get the participants expert reviews about the process. We observed their interactions throughout the session, and they shared their insights during and after the sessions [19]. The sessions concluded by showing some pitfalls for in-class PP, such as differences in developers' backgrounds and experience levels, poor planning and time management, and not allowing enough time for design prior to coding. The expert reviews also provided some recommendations that the experts think would enhance students' performance, such as considering pairing preferences, handling quality requirements as essential and basic requirements, and explicitly asking for usage scenarios and contextual requirements to be considered.

We applied PP as part of in-class activities for a mobile development course. Our objectives were to:

1- Introduce students to PP as a grounded well-known agile practice that has been used widely in several environments
2- Present different models for software engineering
3- Encourage students to evaluate, compare, and critique different approaches for software development
4- Assess student understanding for mobile topics
5- Study how PP would help students in particular and developers in general to develop better mobile applications
6- Investigate how applying PP can differ between regular web/desktop application development and mobile application development

To achieve these goals, we facilitated five 75-minute PP sessions for the 53-student class, which covered five different aspects related to mobile development:

1- Dealing with interface and data management through fragments
2- Using the camera and processing images
3- Connecting two mobile devices via Bluetooth
4- Using accelerometer and GPS data from mobile device
5- Recording and playing audio via mobile device

The driver and navigator exchanged roles every 15 minutes. The class was managed by a main facilitator for PP and two assistants to answer programming-related questions. Every session started with a 10-minute introduction that worked like a retrospective for previous sessions and preparation for the new one. At the end of every session (except the first introductory one), students were required to fill-in an online questionnaire right after they deliver their application. It was not mandatory for students to answer every question, which means that questions might have different number of responses within the same session. Below is a summary of what was performed in every session, and the outputs from each session.

3.1 Session 1

An introductory session featuring a 10-minute overview of PP and a 5-minute presentation about the activity, which was on creating a simple shopping cart app. PP rounds were 15 minutes each, so each student worked twice as a driver and twice as a navigator. Students could select their partner, select locations, and create their own UI.

3.1.1 Observations

Most students had prior basic knowledge of PP, but with minimum previous practice. They welcomed the idea during the presentation, but seemed uncomfortable during the session. After beginning, they immediately started coding, giving no time to consider the application design. Indeed, they did not design a draft interface until second or third rounds. At this stage, many students struggled to adapt their work to fit the activity's minimal requirements. Therefore, core functionality was present, but with poor attention to UI/UX concerns.

The class instructor noted that the number of questions raised during this session was comparable to other sessions. Students did not pay attention to the time constraints, so the facilitator managed time and ask students to switch roles every 15 minutes.

3.1.2 Questionnaires

There was no questionnaire for this session as it was meant only to get students familiar with PP.

3.1.3 Findings

1- Students got annoyed when asked to work on only one screen (i.e., no second laptop)
2- Students didn't like to switch while in the middle of coding certain code segments, and they ignored the time checks until the driver finishes the part in hand
3- In many cases the navigator put his/her hands on the keyboard to write a piece of code instead of explaining what he wants to do to the driver. Some students said that it happened subconsciously while others argued that it makes them work faster.
4- Upon delivery, when students were asked about UI/UX factors they kept showing how the application is "working as required for the activity", which was the only success factor for them, rather than usable interfaces.

3.2 Session 2

The activity exercised camera and image processing tasks. The session started with instructor comments on student work during the previous PP session, mainly related to the roles of driver and navigator. Students were randomly paired based on a seating location rubric. It was announced that a questionnaire should be filled before leaving the classroom.

3.2.1 Observations

Students were surprised by not being able to self-select pairs. However, they accepted it smoothly and started working with their assigned pairs. The activity involved using the mobile camera, so the navigator has always been responsible for holding the mobile and using the camera. Since the application required students to do image processing on selfies, pairs spent time having fun testing their applications by taking selfies. The fun nature of the application was reflected on the coding and class atmosphere. During this session, the number of questions raised by students was noticeably fewer than in session 1. Similar to the first session, little effort was put towards UI/UX.

Navigators were allowed to hold the mobile devices to test code—augmenting to the traditional PP tasks of watching, testing, and collaborating with the driver on coding. This is a special case for mobile application development as rapid parallel testing is done on another screen, rather than the development screen as in web/desktop apps.

3.2.2 Questionnaires

Out of the 22 responses to the open ended question: "How would you describe your experience of today's Pair Programming session?" 18 students provided positive feedback about the session, while 4 had minor complaints (classroom environment, problems with the code, not being comfortable with the experience). The positive comments were mainly from students who delivered their code on time. It is important to note that students' satisfaction with the activity correlates to their ability to deliver it on time.

3.2.3 Findings

1- Students got more used to switching roles and stick to the assigned turns. However, they needed the facilitator to manage time and remind them when to switch
2- Students managed to give themselves more flexibility in switching time. i.e. to wait until driver finishes what s/he is working on before switching
3- It was still hard to keep pairs working on the same screen, as the navigator sought to use another laptop screen for other supporting functions (online search, displaying class material, looking for previous related assignments)

3.3 Session 3

This session involved using Pebble smartwatches to learn about connectivity issues among wearable devices and mobile phones. Students had freedom to choose their partners for two reasons: to compare their performance against the sessions when they were assigned to random pairs; and to incentivize them to work with the same partners who will work with them on that week's homework assignment. Although students seemed, based on our observations, relieved that they would freely choose their partners, the questionnaire results did not concur with such observations.

3.3.1 Observations

Since it was students' first time to program for Pebble, they were not comfortable dealing with it. Connection issues and multiple coding languages had to be addressed, leading to most of them not to be able to deliver a working piece of code, or even to make much progress either in learning or development. They complained about having short time to work on the activity, affecting the atmosphere of the PP session. There were many questions for instructors during that session, and at times partners were not able to help each other. There was disappointment that pervaded this session: Pebble programming, PP, and programming for wearable devices in general.

3.3.2 Questionnaires

Work distribution among pairs was not as good as the previous session. The 50/50 percentage appeared in free-partner-selection mode less than with random pairs. One major reason for this is that when students work with "strangers", they tend to adhere to rules, be more competitive, and do what's required from them to do. Out of the 34 answers to the open ended question: "How would you describe your experience of today's Pair Programming session?" 14 were negative, mainly discussing the insufficient time to complete the activity. 12 students of those who provided negative feedback delivered less than 50% of the required functionality. More positive feedback came from students who delivered 50%-75% of the required functionality. The scope of the activity was a major drawback for this session, and PP need to adapt to deal with such situation.

3.3.3 Findings

1- In-class activities should balance the difficulty of the task with session time.
2- Topics related to connectivity demand more time even for simple activities. PP sessions need to be tailored to give students time to learn new topics together.
3- For new topics, students may need extra time and preparation in advance prior to working on PP.
4- Student disappointment should identify reasons for the problem, whether from working in pairs, using connectivity protocols like Bluetooth, working with wearables, learning new topics, or working within tight time limitations.

3.4 Session 4

This session featured an activity combining GPS and gyroscope. (Note that students had a previous activity on GPS.) Linking the two topics was new for students. Students were assigned to random pairs. To get students to feel more involved with the sessions, the facilitator displayed anonymous comments from the previous session's questionnaire and discussed them with students during the introduction. That step sought to encourage students to give deeper feedback.

3.4.1 Observations

Since students had a previous lab on GPS, they kept referring to prior material and finding complementary resources online. Discussions among pairs seemed richer and questions to the instructors were much fewer. Differences in knowledge levels between pairs led them to have useful discussions and to get advice on how to proceed with coding. Checking previous materials and online searching was time-consuming but helped pairs complete the activity. A majority of the pairs again objected to the "one screen" rule in PP.

Although the activity introduction asked students to consider how users would use the application, only five groups included guidance for users to be able to use their applications. When students delivered their assignments at the end of the session, the facilitator discussed with them how a user will use their application and why they did not pay attention to that issue. As

in previous sessions, students were more worried about providing a functioning project than building a "usable" one.

3.4.2 Questionnaires

25 out of the 33 recorded responses indicated they were able to deliver more than 75% of the required functionality. This rate caused the level of satisfaction in students' answers to the open ended questions to be higher than the previous session. 23 of the 29 responses provided positive feedback. Many students indicated that they were able to understand how to use GPS through discussions with their partners.

3.4.3 Findings

1- Navigators do not spend the whole time just watching and observing drivers; they want to be more active by looking things up while the driver is coding
2- Students, again, tried to use another laptop screen for tasks other than coding, not stopping until the facilitator asks them to go back to a single screen
3- PP seems to work better when students have prior knowledge about the activity, as this increases their ability to exchange knowledge and learn from each other
4- In situations where a navigator tried to do something other than watching the driver, s/he becomes fully attendant when the driver is facing a problem. This kind of self-organized pairs was common among teams
5- Feedback provided and answers for open-ended questions were lengthier than the previous sessions, perhaps because of the comments discussed in the beginning of the session about the previous questionnaire answers

3.5 Session 5

The objective of the activity was to familiarize students with audio in/out tools. This last PP session of the semester added the statement "You are required to include any components that help users understand and use your app". Based on previous session results, students were assigned to random pairs. One objective was to study how students would consider UI/UX issues when it was explicitly stated as a requirement. Students were also asked to be responsible for time-keeping.

3.5.1 Observations

The number of questions directed to instructors was at its minimum during this session. Students were able to explore how to deal with the activities on their own through discussions and web search. As in previous sessions, many tried using other screens to do the online search for some resources despite being directed not to. About half of the groups missed the first switch and needed to be reminded by the facilitator, but only 3 pairs missed it on the second round. On the third and final switch, all pairs managed to switch without reminders.

The final delivered applications had many signs of considering UX requirements, which was not the case with the previous sessions.

3.5.2 Questionnaires

23 out of 29 responses provided positive feedback regarding the session. Some students indicated that PP helped them get the required skills needed to work with the activity with no need to ask for instructor's help. Work distribution among pairs is getting much better as the 50/50 workload appeared more than the previous sessions.

3.5.3 Findings

1- UX requirements need to be explicitly requested rather than broadly referenced; otherwise students do not realize the need to dedicate time for it.
2- Students are able to practice PP on their own after training
3- Online resources and course material are important during the session. Planning for PP sessions should incorporate such resources within the development process

4. DISCUSSION

Questionnaire results showed how PP affects student performance on in-class activities. For example, the percentage of functionality delivered per session changes with the topic being introduced in such session (see Figure 1). It is clear that the multi-device connectivity activity was the one with the least assignments completed. Therefore, this type of activities should be planned carefully to fit within the allowed time. However, student answers to open-ended questions provided more positive feedback (based on the terms they use to describe the session)—increasingly so as sessions go on (see Figure 2). While this suggests PP was helping students deal with complex problems, it also shows that students work better with PP when they get sufficient practice time.

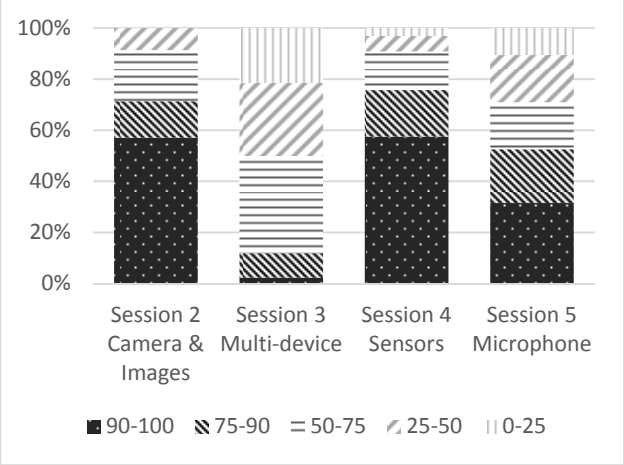

Figure 1. Percentages of delivered functionality

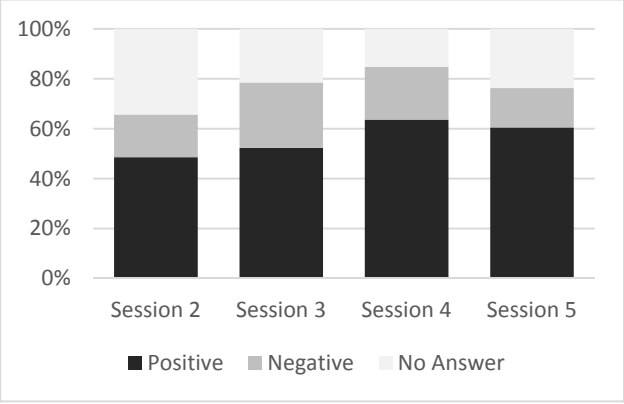

Figure 2. Percentages of positive and negative responses

Along with information from questionnaires, our observations provided different aspects on how students worked with PP. Section 2 postulated three major differences between developing

for web/desktop applications and developing for mobile devices. After the five sessions, we discuss in this section what we found relevant and important to consider regarding the differences.

4.1 Unique Nature of Mobile Development

Our PP variant in working on mobile applications allowed a mobile device in the hands of the navigator. We noted that as the driver writes code, the navigator transitions between the driver's laptop screen and the mobile device. The navigator also looks at the mobile screen to the driver occasionally to discuss some issues. Having two screens seems inevitable when students work on mobile applications, but perhaps the small screen size leads to minimal distraction—allowing the navigator to remain connected to the driver's actions.

4.2 Connectivity, Sensors, and Smartwatches

Sessions 3 and 4 required students to program for multiple devices or sensors. Session 3 leveraged Bluetooth to connect mobile device with Pebble watch, while Session 4 leveraged mobile's GPS sensor to control application responses. During these sessions, we observed that interaction between students peaked, although most of them did not deliver the required application on time. Since many mobile applications require multiple-device connectivity (e.g., watches, sensors, wearables), it takes more time and effort of students to be able to learn, understand, and know how to deal with such applications.

Programming for mobile devices cannot be seen as merely writing code—they also searched online resources, checked lecture materials, explored connectivity issues with their mobile devices, and learned core functionality of the new devices. PP discourages parallelism, asking students to work on the tasks as a team. However, it was hard to keep the two students (especially the navigator) working on the same aspect of the task (especially for non-programming tasks). PP does not include explicit guidelines such situations. Further consideration is needed on this issue, perhaps resulting in a PP approach tailored for mobile design.

4.3 Introducing UI/UX Concepts

Students often feel they can start writing code for their assignments without dedicating time for UI/UX issues. However, UI/UX issues are acknowledged as critical for application functionality due to the different factors that affect mobile interfaces (e.g. screen size, orientation, touch and swipe controls). Hence, UI/UX concepts need to be handled carefully in mobile development environments.

During the several PP sessions conducted (both during expert reviews and in-class activities), there have been two approaches to deal with UI/UX concepts:

1- Present only functional requirements of activities

In multiple cases, students start by writing code segments related to the familiar parts, rather than considering the interface. Only after 2-3 rounds do they find themselves with functional code that lacks a usable interface. At this point, they usually delivered products that lack even the basics of UI design concepts.

This approach is more student-based and it helps when the student is a self-learner. The PP approach helps students to elaborate on how users would deal with their applications, shifting from pair "programming" to pair "design". Moreover, this approach is important for mobile development as there are many UI/UX issues that need to be discussed and evolved by the students' own work—not always the case with traditional programming with its less dynamic components.

2- Specify UI/UX issues as essential requirement

As seen in the fifth session, when usage scenarios were a requirement and part of the activity description, students do dedicate time on their first round to discuss interface issues. When they delivered the application, they were keen to explain its usability, and how users can deal with it in different ways.

This second approach can fit with classes that do not have time to do many PP sessions (when lab time is very limited). It directs students from the beginning to consider UI/UX aspects so that they better plan on how to use their time. However, it will lack the ability to show students through practice the importance of these aspects, especially for mobile development.

Both approaches will educate students on UI concepts during PP sessions. However, the first approach encourages experiential learning if student activities are carefully scoped. Whether an instructor decides to go with the first or second approach, it is important not to disregard UI/UX concepts when working on in-class activities, particularly for mobile platforms. PP can help students discuss and focus on those aspects, especially when they work on applications that require extra attention to how users will interact with such applications.

5. RECOMMENDATIONS

Based on our exploratory study for how to introduce mobile development concepts to students using PP, we came to find some practices that can help instructors get the best of PP while achieving their course learning objectives. Below is a list of our recommendations.

1- Allow students to choose partners on the first session, then move to random pairing

The first session lets students unfamiliar with PP focus on learning how PP works instead of figuring out how to deal with an unfamiliar work partner. Based on questionnaire answers and our observations, students tend to be more productive and satisfied when they deal with partner they may not know as well (sessions 2, 4, and 5). Two possible reasons for this are that students are more relaxed when they work with someone they know (no pressure for competency and excellence), and they gain more knowledge from someone whom they did not have interaction with before (as opposed to a friend, whose knowledge about new topics is usually shared among his/her close mates/group).

2- Allow the navigator to look at the mobile device screen while the driver works on coding

Although this contradicts a PP core tenet, this practice seems inevitable for mobile development. Navigators usually need to check the mobile device they are working on, test and debug the application on the mobile device, and work on issues like connectivity and sensors. Asking pairs to keep focused on the coding screen was a constant disappointment for many pairs, with claims that it hindered their progress. It is also important to monitor that navigator gives full attention to coding when the driver faces issues.

3- Decide on the strategy to be used to introduce UI/UX requirements based on the amount of time available for each session

Sessions 4 and 5 show students giving more attention to UI/UX issues much more than the previous sessions. As shown in section 5.3, the instructor should dedicate the first round to only work on UI/UX issues if students will not practice PP themselves. Otherwise, students may be left to learn the

importance of including UI/UX thinking the hard way during the multiple sessions. In either case, PP can address UI/UX issues as students exchange views and create scenarios in pairs on how users would deal with their applications. Moreover, since mobile development requires more attention to details when it comes to UI/UX, PP seems that it can be a good fit—but with careful attention from the instructor.

4- Plan for extra time and pre-class preparation for activities that involve using multiple devices and connections

Most mobile development applications require dealing with multiple platforms and connections, new techniques and technologies, different programming interfaces, and multiple sensors. Instructors need to carefully consider whether students can handle all the required tasks within the class time. Moreover, students may be asked to prepare for the task before they come to the class, so that they become ready to work on the assignment within the allowed session time. The value in PP lies in the ability to discuss and exchange ideas, but the reward lies in completing a task. Students' performance and feedback is highly affected by the ability to complete a task. Therefore, it is important to size activities to fit within the allowed time so that students can keep working with good spirit.

6. CONCLUSIONS

Although there has been lots of research on applying PP in CS classrooms, this paper highlights that mobile development requires further investigation on how PP will perform—and how it might change. Our in-class study showed that activities in a mobile development course would differ than regular CS programming course activities in several aspects. Issues related to mobile UI/UX, sensors, and multi-device connectivity introduce unique challenges to students in mobile development classes. PP seems to provide a reasonable approach to handle such challenges, though with changes put in place. For PP to perform better in mobile development classes, instructors should consider how PP has traditionally been applied, as well as why some changes may positively influence student learning and performance. We expect that this work introduces more questions than it answers, and we look forward to learning about the answers through future PP efforts.

7. REFERENCES

[1] 2001. Manifesto for Agile Software Development. http://agilemanifesto.org Accessed: June 15, 2015.

[2] Seyam, M. and Galal-Edeen, G.H., 2011. Traditional versus Agile: The Tragile Framework for Information Systems development. *International Journal of Software Engineering (IJSE) 4*, 1, 63-93.

[3] Williams, L. and Kessler, R., 2002. *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc.

[4] Beck, K. and Andres, C., 2004. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

[5] Plonka, L., Sharp, H., Van Der Linden, J., and Dittrich, Y., 2015. Knowledge transfer in pair programming: An in-depth analysis. *International Journal of Human-Computer Studies 73*(1//), 66-78.

[6] Vanhanen, J. and Lassenius, C., 2007. Perceived effects of pair programming in an industrial context. In *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications*.

[7] Phaphoom, N., Sillitti, A., and Succi, G., 2011. Pair Programming and Software Defects – An Industrial Case Study. In *Agile Processes in Software Engineering and Extreme Programming*, A. SILLITTI, O. HAZZAN, E. BACHE and X. ALBALADEJO Eds. Springer Berlin Heidelberg, 208-222.

[8] Katriou, S. and Tolias, E., 2009. From twin training to pair programming. In *Proceedings of the 2nd India software engineering conference, ISEC '09* ACM, New York, USA.

[9] Cockburn, A. and Williams, L., 2001. The Costs and Benefits of Pair Programming. In *Proceedings of Second International Conference on Extreme Programming and Flexible Processes in Software Engineering*.

[10] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S., 2003. Improving the CS1 experience with pair programming. In *Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE'03)*.

[11] Mcdowell, C., Werner, L., Bullock, H., and Fernald, J., 2006. Pair programming improves student retention, confidence, and program quality. *Communications of ACM 49*, 8, 90-95.

[12] Kuppuswami, S. and Vivekanandan, K., 2004. The effects of pair programming on learning efficiency in short programming assignments. *Journal of Information Education 3*, 2, 251-266.

[13] Simon, B. and Hanks, B., 2007. First Year Students' Impressions of Pair Programming in CS1. In *Proceedings of the Third International Computing Education Research Workshop*, 73-86.

[14] Salleh, N., Mendes, E., and Grundy, J., 2011. Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *Software Engineering, IEEE Transactions on 37*, 4, 509-525.

[15] Rahimian, V. and Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach. In *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*, 337-342.

[16] Tigrek, S. and Obadat, M., 2012. Teaching smartphones programming using (Android Java): Pedagogy and innovation. In *Information Technology Based Higher Education and Training (ITHET), 2012 International Conference on*, 1-7.

[17] Esakia, A., Niu, S., and McCrickard, D.S., 2015. Augmenting Undergraduate Computer Science Education With Programmable Smartwatches. In *Proceedings of the Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (Kansas City, Missouri, USA2015), ACM, 2677285, 66-71.

[18] Williams, L., McCrickard, S., Layman, L., and Hussein, K., 2008. Eleven Guidelines for Implementing Pair Programming in the Classroom. In *Agile, 2008. AGILE '08. Conference*, 445-452.

[19] Seyam, M. and McCrickard, S., 2015. Collaborating on Mobile App Design through Pair Programming. In *Proceedings of International Conference on Collaboration Technologies and Systems (CTS 2015)*, Atlanta, GA.