Pair Programming as a Teaching Tool

Premalatha Sampath
IT Lecturer
Premalatha.Sampath@whitireia.ac.nz

ABSTRACT

Teaching introductory programming to entry-level undergraduate computing students can be an extremely challenging task, as many first year students have little or no experience of computer programming before commencing their degree course. The challenge is developing the required analytical and problem solving skills in order to learn to program. This can prove to be a very difficult task for first year undergraduates, and it is especially difficult when programming alone.
Pair Programming is an Agile software development technique where two programmers work collaboratively on a single computer on the same program or code. Industry's Pair Programming methodology has been adopted in academic settings as a form of collaborative learning. Pair programming has been widely implemented in education because of the benefits it brings to entry-level students at university. The objective of this study was to explore students' perceptions of the impact of Pair Programming and the influence of Pair Programming on students' learning experience. The results showed that the majority (78%) of students enjoyed the Pair Programming experience and 77% gained knowledge when pairing.

Keywords: Pair Programming, Collaborative Learning

1. INTRODUCTION

Pair Programming (PP) is an approach used to assist first year programming students to overcome a perception that programming is difficult (Smith & Delugach, 2010). Anecdotal evidence suggests that first year programming students found the challenge difficult, leading these students to potentially disengage from the course. PP practice used in industry as an agile development method has been widely implemented in education because of the benefits it brings to entry-level students in tertiary education. PP is the term used to describe the process by which two programmers work side by side, on the same programming task. Typically the two roles in pair programming are a 'driver', who actively types at the computer or records a design and a 'navigator', who watches the work of the driver and attentively identifies problems and makes suggestions. The two programmers collaborate in designing, coding and reviewing.
Computer programming is a compulsory aspect of the majority of undergraduate computing courses. However, teaching introductory programming to first year undergraduate computing students can be an extremely challenging task, particularly as many first year students have little or no experience of computer programming before commencing their degree course (Dunican, 2002; Gomes, Santos & Mendes, 2012). Indeed, the difficulty experienced when learning programming is often thought to be a contributing factor in the retention rates and levels of student engagement on undergraduate computing courses (Miliszewska & Tan, 2007). Programming has traditionally been thought of as a solitary activity and this view has been echoed throughout computing education (Cockburn & Williams, 2000; McDowell, Werner, Bullock, & Fernald, 2003). However, attempting to develop the required analytical and problem solving skills in order to learn to program is a very difficult task for first year undergraduates, and it is especially difficult when doing it alone (Somervell, 2006).

This quality assured paper appeared at the 8th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2017) and the 30h Annual Conference of the National Advisory Committee on Computing Qualifications, Napier, New Zealand, October, 2-4, 2017. Executive Editor: Emre Erturk. Associate Editors: Kathryn MacCallum and David Skelton.

An alternative method of teaching programming is to enable two students to work collaboratively on a single computer on the same program or piece of code, a technique known as pair programming (Williams & Upchurch, 2001). The potential benefits of pair programming include increased productivity, knowledge transfer, learning and morale, and fewer defects (Freeman, Jaeger, & Brougham, 2004; Haungs, 2001).

2. BACKGROUND AND RELATED WORK

Robins, Rountree & Rountree (2003) have acknowledged that Programming is not a single skill, but rather a complex cognitive activity. The complex cognitive activity involved is when the student is required to simultaneously code and apply several higher order cognitive skills to solve a given particular problem. Various authors have discussed the capabilities and weaknesses that students may have. Students may lack the motivation to learn programming as they may be unable to create a mental model (Bruce-Lockhart & Norvell, 2007; Sorva, 2013) to understand the programs associated with the underlying system, or may be unable to visualise a clear model of program flow (Milne & Rowe, 2002). In addition to this, syntax and underlying semantics are required to develop programs. Other difficulties are related to the teaching methodology used to teach Introductory Programming. Due to high failure rates in programming courses, instructors face the additional challenge of adjusting their expectations to the students' level of ability (Utting et al., 2013).

Traynor and Gibson (2004) agree that programming is best learned through practice and experience. Unfortunately, textbooks and lecture materials are often heavy on "declarative knowledge" – with emphasis on the features of programming languages on "how to use them". Therefore, this "declarative knowledge is not adequate for students to learn to write a program. In addition, the programming concepts originated from a purely mathematical discipline. Also, these programming concepts are underpinned by technical conceptual ideas of digital computing, which adds much more complexity to understand programming. While most courses in computing involve a practical component, the most suitable or reachable means of implementing this component remain arguable (Maguire & Maguire, 2013). Miliszewska and Tan

(2007) found that changing to teaching methods that included textbooks and online resources led to little improvement in programming competence.

In industry, most projects are collaborative efforts whereas in academia it is the belief that students should write programs in isolation. The lack of a formalised environment is identified as one of the key issues which may be exacerbating students' difficulties in collaborative peer learning. The advantage of collaborative work is that "Two heads are better than one". The value of collaboration in software development is explicitly encouraged through the PP practice. Several benefits of PP have been acknowledged including improved understand-ability (Vanhanen & Korpi, 2007) and maintainability of code and decreased defect rates (Phaphoom, Sillitti, & Succi, 2011; Phongpaibul & Boehm, 2007) and knowledge transfer (Vanhanen, Lassenius & Mantyla, 2007).

2.1 Collaborative Learning

Curriculum guidelines for undergraduate degree programmes in the Information System (IS 2010) and Skills Framework for the Information Age (SFIA) identified (Institute of IT Professionals New Zealand, 2016) that teamwork and collaboration skills are some of the main required exit characteristics of IS graduates. Collaborative learning involves joint intellectual effort by groups of students working together. PP is a form of collaborative learning. There has been a significant amount of research on PP as a collaborative learning tool in the traditional classroom. Collaborative learning is also considered one of the main benefits of PP in both professional and educational settings. Several studies have indicated that students perceived that they learned more by working with a partner than they would have by working alone (Braught, Wahls, & Eby, 2011; Carver, Henderson, Lulu, Hodges & Reese 2007).

2.2 Pair Programming and Collaborative Learning

Collaborative learning involves joint intellectual effort by groups of students working together. PP is a form of collaborative learning. It incorporates the critical attributes of a collaborative learning activity: 1) common task; 2) small group learning; 3) cooperative behaviour; 4) interdependence; and 5) individual accountability (Preston, 2005). There has been a significant amount of research on pair programming used as a collaborative learning tool in the traditional classroom. Most of the previous research that has explored students' perceptions suggests that students have a positive attitude toward collaboration and communication that takes place during pair programming (Howard, 2006) and that they perceived that pair programming helped them develop teamwork skills (Cliburn, 2003; Edwards, Stewart & Ferati, 2010). It has been reported that students enjoyed working in teams (Chigona & Pollock, 2008; Cliburn, 2003; Faja, 2014; Howard, 2006; McDowell et al., 2006; Mentz, Van der Walt, & Goosen, 2008; Williams & Kessler, 2001; Zacharis, 2011). In addition, students who worked in pairs reported higher confidence in the correctness of program solutions compared to individual programmers (Braught, at el, 2011; Werner, Hanks & McDowell, 2004; McDowell, Werner, Bullock, & Fernald, 2006; Williams & Kessler, 2001). Several studies indicate that pair programming helped to reduce student frustration (Howard, 2006; Simon & Hanks, 2008; Braught et al., 2011). Collaborative learning is considered one of the main benefits of pair programming in both professional and educational settings. Several studies indicated that students perceived that they learned more by working with a partner than they would have by working alone (Cliburn, 2003; Carver et al., 2007; Edwards et al., 2010).

3. EDUCATIONAL BENEFITS OF PAIR PROGRAMMING

Several studies indicate that pair programming has reported educational benefits.

3.1 Academic Performance and Learning

Other studies have focused on the effect of implementation of pair programming on academic performance and learning. Some studies have reported higher assignment grades for pairs compared to solo programmers (Mendes, Al-Fakhri, & Luxton-Reilly, 2005), likelihood of course completion (McDowell et al., 2006) and higher pass rates (Chigona & Pollock, 2008; McDowell et al., 2006; Mendes, Al-Fakhri & Luxton-Reilly, 2006; Williams et al., 2002). In contrast, other studies showed that assignment grades for pairs were not significantly different from solo students (Zacharis, 2011). There were cases when students felt that they understood their programs better when they worked by themselves (Simon & Hanks, 2008). However, these research findings were not consistent. Faja (2014) indicated that students perceived paired programming beneficial in learning and enjoyment. Another study also observed that paired students were significantly more likely to remain in the course through to the final exam (McDowell, Werner, Bullock and Fernald, 2006). It was also acknowledged that paired students were 18% more likely to attempt their next course in programming. Salleh et al. (2011) concluded that the pair programming approach has consistently lead to improved grades and increased student satisfaction.

3.2 Quality and Productivity

In addition to academic performance and learning, previous research has focused on other potential benefits of pair programming such as program quality and productivity. Many studies showed that pair programming had improved the quality of the programs (Chigona and Pollock, 2008; Zacharis, 2011). However, previous research does not provide the same support for the impact on productivity. Zacharis (2011) and Salleh (2011) found that paired students were more productive than individual programmers and they completed tasks in a shorter amount of time.

A review of the literature on pair programming in computing education revealed other areas that have been of interest to educators and researchers. Several studies adopted different pair formation techniques such as pairs assigned randomly (Hahn, Mentz & Meyer, 2009), matching pairs based on academic performance (Choi et al., 2009; Zacharis, 2011), and pairs with slightly different levels (Lee & Ahlswede, 2007).

3.3 Retention

Studies have shown that pairing in introductory programming courses appears to have a positive effect on retention. A two-year study was conducted to investigate the impact on student retention in computer-related majors of pair-programming in CS (Carver et al., 2007). To measure retention, Carver et al., (2007) recorded the major of all students who were first-year enrolled in CS1, and later recorded the major of these students after a year. The results were positive showing that 41 of 48 students 85.4% who were in the pair programming section were still in computing majors a year later, whilst only 64.3% of those who worked alone in the CS1 during the study were still in the same majors.

Students who were paired in McDowell et al.'s study (2006) were much more likely to take a second programming course. Of the students who indicated that they intended to pursue a computing-related major when they were enrolled in CS1, and passed CS1 with a C grade or better, significantly more of those who paired, attempted the second programming course within 1 year (84.9% of paired students vs. 66.7% of solo students).

These students were also much more likely to persist in a computing-related major. For those students who indicated a desire to major in computer science when they took CS1, the paired students were significantly more likely to have declared a computing-related major 1 year later. Nagappa et al. (2003) observed that students were more productive, less frustrated and more motivated to advance. It was also noted that the students who were involved in PP were more active learners than those who used the then traditional approach.

There is a considerable volume of research which has proved the positives of peer learning (Boud & Lee, 2005; Jackson & Bruegmann, 2009; Hammond et al., 2010; Kepell et al., 2006; Menzies & Nelson, 2012; Staarman, Krol & Meijden, 2005; Topping, 2005; Willey & Gardner, 2010). As a result, there has been a gradual shift in the teaching paradigm from an 'instructivist' approach to a 'social constructivist' approach (Tan, Yeo & Lim, 2005). There were numerous explanations for high failure rates in programming courses. One of the main reasons was the approach taken to teach programming, with limited practical lab exercises. Hawi (2010) found that although failures in programming were related to many diverse factors, one of the key reasons was "inappropriate teaching methods" and another reason was lack of opportunity to practice. Moreover, computing was often viewed as an isolated task (Redmond, Evans & Sahmi, 2013) which had a negative impact on students' perceptions and learning strategies.

In educational settings, there is notable evidence that pair programming has had a significant positive impact towards learning to program in introductory programming courses. Pair programming is also used in second-year courses, learning object-oriented programming and even at the graduate level.

4. STUDIES CONDUCTED IN NEW ZEALAND

Mendes, Al-Fakhri, and Luxton-Reilly (2005) from the University of Auckland concluded that the students in PP classes performed better on programming exercises than individuals. Likewise, a study conducted by Auckland University of Technology revealed that the quality of work done by pairs was either equal to or better than that of the individuals, despite having spent only half the amount of time on the task (Sadasivan, 2005). A similar study of Massey University students, acknowledged that there are benefits to pair programming, in particular improved software quality and therefore supporting the claims made by previous studies (Parsons, Ryu, & Lal, 2008). A study conducted by Weltec, confirmed that pair programming reduced students levels of anxiety during the programming classes, however little improvement was observed from their assessment marks (Jeon & Manueli, 2013; Jeon & Hunter, 2015). In summary, a number of previous studies have mainly been concerned with conducting experiments on the pair programming practices in Computer Science and IT (Level 4) courses to observe the benefits of the technique. The objective of this study was to explore students' perceptions of the impact of pair programming and the influence of pair programming on student's learning experience. This study was conducted with Level 5 Information Technology students in the IT5x84 programming course, an entry level programming paper offered to the first year students.

5. METHODOLOGY

The students of the IT5x84 Programming courses at Whitireia Polytechnic in Auckland were invited to participate in the study. The IT5x84 paper introduced the students to fundamental programming skills over 13 weeks in a Trimester. In the 13 week course, students had to complete 7 lab exercises and 3 assignments. Out of 8 lab exercises, 4 exercises were equally divided to complete their tasks individually and 4 in pair programming. Likewise, out of 2 assignments, 1 assignment was completed individually and another assignment in pairs.

5.1 Procedure

The students of the IT5x84 Programming course at Whitireia were typically assessed with 40% of the marks awarded based on the end of trimester written exam, 10% for weekly lab exercises and 50% for assignments (Assignment 1 and Assignment2). Assignment 1 had two milestones, Milestone 1(10%) and Milestone 2 (20%). Assignment 1 Milestone 2 (Object Oriented Programming) was a PP assignment. Due to the complex nature of OOPs programming concepts for this part of the assignment the students were asked to work in pairs. Nearly 50% of the overall course assignment was allocated to pair programming.

5.2 Pre-Intervention Phase (Individual Programming)

For the first 4 weeks in the trimester (pre-intervention phase) students were required to carry out weekly programming exercises individually. In these 4 weeks the lab exercises or even class tasks were carried out individually to understand the students' self-efficacy at programming and confidence at writing a program to solve the problem. These individual exercises obviously enabled the tutor to understand the students' capabilities and their self-efficiency.

5.3 Intervention Phase (Pair Programming)

From the 5th week onwards the students were paired to work on their tasks. The pair programming partners were selected according to skillset. The students were paired in a rotation fashion (Carver et al. 2007; Braught et al., 2011), therefore over the 13 weeks each student programmed with different partners. Each week the pairs were rotated to work on their lab exercises. However, the team selected to work on Assignment 1, Milestone 2 remained the same until they had completed their task. The reason the teams remained the same was to have a clear understanding and contribution to their assignment. If it was a rotating pair then there could have been an issue in managing their tasks and working towards completion of their assignment. The assignments were based on a given case study and the tasks are clearly mentioned with core-functionalities. Unlike weekly lab sheets work, the assignment took nearly 20 hours of work, therefore the student was set to have the same partner until the assignment was completed.

While focused on the benefits of PP the research did not leave out individual accountability, which was assessed through individual weekly lab exercises and tests. In addition to individual accountability, the students were required to individually respond to specific questions posed by the tutor at the end of lab sessions. If the students simply copied the code from their partner without understanding the purpose of the code, and were consequently not able to answer the question, then they lost the individual marks. The objective of PP was to combine both positive interdependence and individual accountability that would encourage students to understand the problem, explain the code to each other and foster a collaborative association. The goal was to ensure that the students were involved in a collaborative learning activity and acquired the skills the activity was intended to teach.

Based on the review of previous research, this study considered the effectiveness of pair programming as a teamwork activity: perceived confidence levels in quality of work completed in pairs, perceived productivity, perceived learning and enjoyment. Like any other team activity, one concern

educators have had about using pair programming was unequal participation of pair members. A suggestion was made to change partners throughout the semester as an approach to deal with "free- riders". Social loafing is a well-known phenomenon in teamwork activities, which occurs when individuals working together in groups put less effort than when they work individually (Balijepally et al., 2009).

At the end of the intervention (ie end of Week 13) an online survey was posted to get feedback from students regarding their experience of pair programming. The students had to rate their experience on a 5 point Likert scale dealing with enjoyment, problem solving, code productivity and quality, error detection, team work, knowledge transfer and overall satisfaction.

6. RESULTS

A quantitative approach was utilized to analyse the results. The data was collected from trimester T1 2016 from 12 participants, T3 2016 from 11 participants and T2 2017 from 9 participants.

6.1 Enjoyment

Student enjoyment of pair programming was measured on a Likeart Scale from 1 (Strongly Disagree) 2 (Disagree) 3 (Neutral) 4 (Agree) and 5 (Strongly Agree). The results showed that 42.85% of the students agreed that they enjoyed pair programming and 31.25% of the students strongly agreed (Figure 1). Therefore, the results suggested that the majority enjoyed pair programming.

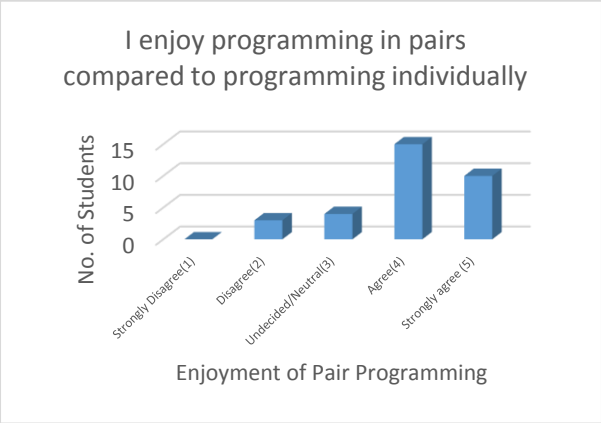

Figure 1: Distribution of responses to the question ''I enjoy programming in pairs compared to programming individually".

6.2 Problem-Solving

The survey included a question on the perceived programming skills gained out of PP. The survey showed that 16% of respondents believed that pair programming enhanced problem-solving skills and 25% their solution design. 33% of respondents found their coding skills were enhanced and most of the respondents acknowledged a combination of all skillsets.

31.25% of students felt that it took the same amount of time to do the same amount of work than doing it alone and 31.25% of students felt it took a little less time compared to working alone. Almost nearly one-third of students felt it took somewhat more time.

6.3 Code Productivity and Quality

The survey also reported that 59% of students were pleased that the quality of the programs created in pairs was higher whereas 35% of students felt that the quality of the program remained the same. Another question was also included in this section for students to comment on the bugs found during their pair programming exercises. 45% of students found that the number of bugs was the same when they worked solo, however 35% of the students commented that they found fewer bugs during pair programming.

6.4 Error Detection

The study also found that 26% of students were almost always able to find the errors efficiently while 55% of students found that sometimes they were able to find errors efficiently. In terms of detecting other students errors 35% students were efficient (almost always) while 52% of students found that sometimes they were able to identify these (Figure 2).

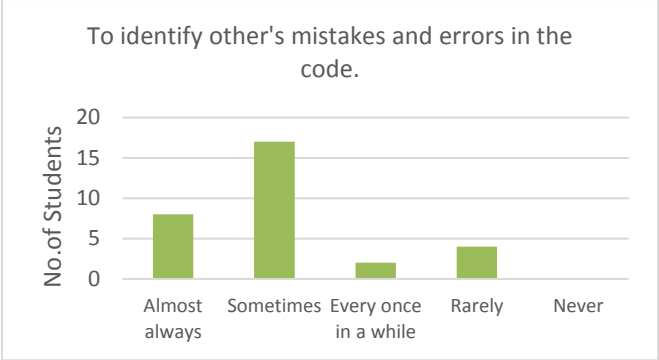

Figure 2: Distribution of responses to the question "To identify other's mistakes and errors in the code".

6.5 Overall Satisfaction

The results highlighting the overall level of satisfaction with the peer programming technique indicated that 66% of respondents believed that pair programming enhanced their programming skills. Another question was included in this section for students to comment on the success of pair programming. Figure 3 shows that overall 78% of students agreed that pair programming was more successful than individual programming.

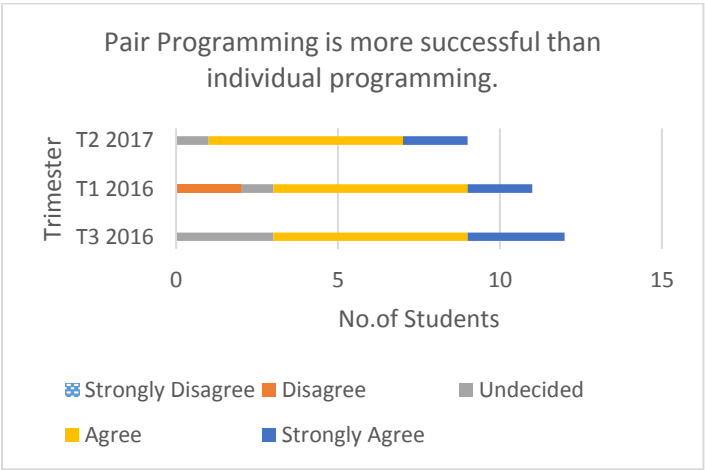

Figure 3: Distribution of responses to the question ''Pair programming is more successful than individual programming form trimester T1, 2016, T3 2016 and T2 2017.

Students who paired stated that they would like to have pair programming as part of future information technology courses. Finally, overall 78% of students indicated that they would recommend pair programming to other students.

7. DISCUSSION

Results from T1 2016, T3 2016 and T2 2017 showed that PP increased the practical lab completion rate significantly for those students who initially showed the lowest confidence levels. This was especially the case for students whose

confidence levels improved between Week 5 and Week 11. These students earned higher final marks, and eventually gained a good grade.

It was apparently difficult to catch up if a student fell behind in the first four weeks of a Programming paper. This finding was supported by Carver et al., (2010) who demonstrated that failure in the earliest weeks can be a significant contributor to rates in a first year programming course. It is important to identify students at risk and provide additional support. This is crucial for the students who did not gain confidence with programming at the earlier weeks. Student feedback on the use of PP was generally positive as it was helpful to discuss programming problems and solutions with their partners. It is evident from the findings of Wood, Parsons, Gasson & Haden (2013) that students need not wait for the instructors' feedback as their partners feedback was prompt.

At times, students expressed concern about incompatible pairing, and the tutor had to intervene when required. Apart from minor shortcomings, positive experiences were had in the class that included shorter waiting times. During practical work time, the tutor moved around the room offering assistance and answering questions when students were not progressing. In the first few weeks of the programming course, when most students have very little knowledge of how to program, this was found to be a tough task for the tutor.

Previously, the key issue experienced by the students in the introductory programming course was the lack of engagement. One of the reasons identified for this was that some students failed the paper previously and were repeating it (Parsons, Ryu, & Lal, 2008). Another key factor also noticed was the lack of interest and fear from the previous trimester. For these students, the earliest weeks can seem pointless. The tutor noted, that when working with another student who they had prior experience they were much more engaged than in the previous trimester. Students seemed to be keener to work with students of a similar level, perhaps to share interesting approaches or to include additional functionalities to the given exercises. This eventually increased motivation among the repeaters and other students who enrolled in the paper.

The indications were that the use of the pair programming technique created a positive learning experience for students in this research group. The survey highlighted that the students enjoyed the experience of pair programming, delivered higher quality coding, and were able to understand more in relation to coding structures. This was evident and consistent with similar studies (Hammond et al., 2010; Willey & Gardner, 2010). As found in the literature, students felt they were able to efficiently identify and fix errors in a shorter time in a team than they would individually (Mendes at al., 2005; Sadasivan, 2005). In addition, the results showed that the majority of students thought their programming skills were enhanced by the pair programming technique, therefore this research was consistent with other research undertaken with students at the tertiary level (Hammond et al., 2010; Mendes at al., 2005, Parsons et al., 2008, Willey & Gardner, 2010).

8. CONCLUSION

As a discipline, IT is constantly changing therefore it is important to introduce students' to new software development approaches in the classroom that connect with the industry. In addition, most modern organizations require individuals to work collaboratively in teams to perform tasks. Most of the previous studies on pair programming dealt with Computer Science students, and only a few studies used Information Systems students. There are important differences in the skills of the students in these two disciplines as well as programming requirements. The research showed several benefits of using pair programming in academic settings such as positive learning, greater confidence in work quality, higher problem solving skills, enhanced interaction skills, and improved team building skills. Definitely, PP is a bridge that leads from individual programming to team collaboration, which is a much-needed skillset needed for someone to work in an Agile environment.

9. REFERENCES

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are two heads better than one for software development? The productivity paradox of pair programming. *MIS Quarterly*, *33*(1) 91-118. Retrieved from https://www.misq.org/

Boud, D., & Lee, A. (2005). 'Peer learning'as pedagogic discourse for research education 1. *Studies in Higher Education, 30*(5), 501-516. http://dx.doi.org/10.1080/03075070500249138

Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE), 11*(1), 1-21. doi: 10.1145/1921607.1921609

Bruce-Lockhart, M. P., & Norvell, T. S. (2007, October). *Developing mental models of computer programming interactively via the web.* Paper presented at the 37th Annual Frontiers In Education Conference -Global Engineering: Knowledge Without Borders, Opportunities Without Passports, Milwaukee, WI.

Carver, J. C., Henderson, L., He, L., Hodges, J., & Reese, D. (2007). Increased retention of early computer science and software engineering students using pair programming. *Proceedings of the 20th Conference on Software Engineering Education & Training*, 115-122. doi: 10.1109/CSEET.2007.29

Chigona, W., & Pollock, M. (2008). Pair programming for information systems students new to programming: Students' experiences and teachers' challenges. *Proceedings of of the Portland International Conference on Management of Engineering & Technology, PICMET 2008*, 1587-1594. doi: 10.1109/PICMET.2008.4599777

Choi, K. S., Deek, F. P., & Im, I. (2009). Pair dynamics in team collaboration. *Computers in Human Behavior, 25*(4), 844-852. https://doi.org/10.1016/j.chb.2008.09.005

Cliburn, D. C. (2003). Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges, 19*(1), 20-29. Retrieved from https://www.ccsc.org/publications/

Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. In G. Succi, & M. Marchesi (Eds.), *Extreme programming examined* (pp. 223-247). Boston, MA: Addison-Wesley Longman.

Dunican, E. (2002, June). *Making the analogy: Alternative delivery techniques for first year programming courses.* Paper presented at the 14th Workshop of the Psychology of Programming Interest Group, Brunel University.

Edwards, R. L., Stewart, J. K., & Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads, 1*(1), 48-54. doi: 10.1145/1721933.1721951

Faja, S. (2014). Evaluating effectiveness of pair programming as a teaching tool in programming courses. *Information Systems Education Journal, 12*(6), 36-45. Retrieved from www.isedj.org

Freeman, S. F., Jaeger, B. K., & Brougham, J. C. (2003, June). *Pair programming: More learning and less anxiety in a first programming course*. Paper presented at the 2003 Annual Conference of the American Society for Engineering Education, Nashville, TN.

Gomes, A. J., Santos, A. N., & Mendes, A. J. (2012). A study on students' behaviours and attitudes towards learning to program. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, 132-137. doi: 10.1145/2325296.2325331

Hahn, J., Mentz, E., & Meyer, L. (2009). Assessment strategies for pair programming. *Journal of Information Technology Education: Research, 8*(1), 273-284. Retrieved from https://www.informingscience.org/Journals/JITEResearch/Overview

Hammond, J. A., Bithell, C. P., Jones, L., & Bidgood, P. (2010). A first year experience of student-directed peer-assisted learning. *Active Learning in Higher Education, 11*(3), 201-212. https://doi.org/10.1177/1469787410379683

Haungs, J. (2001). Pair programming on the C3 project. *Computer, 34*(2), 118-119. doi: 10.1109/2.901173

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education, 54*(4), 1127-1136. doi: 10.1016/j.compedu.2009.10.020

Howard, E. V. (2006). Attitudes on using pair-programming. *Journal of Educational Technology Systems, 35*(1), 89-103. https://doi.org/10.2190/5K87-58W8-G07M-2811

IS. (2010). Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. https://www.acm.org/education/curricula/IS%202010%20ACM%20final.pdf

Institute of IT Professionals New Zealand (2016), SFIA 5 framework, Retrieved from https://itp.nz/upload/files/SFIA5ref.en.r4.pdf

Jackson, C. K., & Bruegmann, E. (2009). Teaching students and teaching each other: The importance of peer learning for teachers. *American Economic Journal: Applied Economics, 1*(4), 85-108. doi: 10.1257/app.1.4.85

Jeon, T., & Hunter, I. (2015). The impact of pair-programming on entry level information technology students. *Proceedings of the Conference of Computing and Information Technology Research and Education New Zealand*. Retrieved from http://www.citrenz.ac.nz/2015-proceedings/

Jeon, T., & Manueli, K. (2013). Investigating the Impact of Pair-programming on Entry Level IT Students. *Proceedings of the Conference of Computing and Information Technology Research and Education New Zealand*. Retrieved from http://www.citrenz.ac.nz/2013-proceedings-2/

Keppell, M., Au, E., Ma, A., & Chan, C. (2006). Peer learning and learning-oriented assessment in technology-enhanced environments. *Assessment & Evaluation in Higher Education, 31*(4), 453-464.

Lee, R., & Ahlswede, T. (2007). *Evaluating the usefulness of pair programming in a classroom setting*. Paper presented at the 6th IEEE/ACIS International Conference Computer and Information Science, 2007. ICIS 2007. Melbourne, Qld., Australia.

Maguire, P., & Maguire, R. (2013). Can clickers enhance team based learning? Findings from a computer science module. *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education, 5*(3), 1-16. Retrieved from http://ojs.aishe.org/

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. *Proceedings of the 25th International Conference on Software Engineering.* doi: 10.1109/ICSE.2003.1201243

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90-95. doi: 10.1145/1145287.1145293

Mendes, E., Al-Fakhri, L., & Luxton-Reilly, A. (2006). A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. *ACM SIGCSE Bulletin, 38*(3), 108-112. doi: 10.1145/1140123.1140155

Mendes, E., Al-Fakhri, L. B., & Luxton-Reilly, A. (2005). Investigating pair-programming in a 2nd-year software development and design computer science course. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 296-300. doi: 10.1145/1067445.1067526

Mentz, E., Van der Walt, J., & Goosen, L. (2008). The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education, 18*(4), 247-260. http://dx.doi.org/10.1080/08993400802461396

Menzies, V. J., & Nelson, K. J. (2012). Enhancing student success and retention: an institution-wide strategy for peer programs. *Proceedings of the 15th International First Year in Higher Education Conference : New Horizons*. Retrieved from http://fyhe.com.au/past_papers/papers12/Papers/2D.pdf

Miliszewska, I., & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Issues in Informing Science and Information Technology, 4*. Retrieved from https://www.informingscience.org/Journals/IISIT/Overview.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information Technologies, 7*(1), 55-66. Retrieved from https://link.springer.com/journal/10639

Nagappan, N., Williams, L., Wiebe, E., Miller, C., Balik, S., Ferzli, M., & Petlick, J. (2003). Pair learning: With an eye toward future success. In F. Maurer, & D. Wells (Eds.), *Proceedings, Extreme Programming and Agile Methods - XP/Agile Universe 2003* (pp. 185-198). Berlin, Germany: Springer.

Parsons, D., Ryu, H., & Lal, R. (2008). Better, not more expensive, faster? The perceived effects of pair programming in survey data. In *ACIS 2008 Proceedings* (pp. 710–719). Christchurch: Association for Information Systems.

Phaphoom, N., Sillitti, A., & Succi, G. (2011). Pair programming and software defects–an industrial case study. In A. Sillitti, O. Hazzan, E. Bache, & X. Albaladejo (Eds.), *Proceedings, 12th International*

Conference on Agile Software Development (pp. 208-222). Berlin. Germany: Springer.

Phongpaibul, M., & Boehm, B. (2007). A replicate empirical comparison between pair development and software development with inspection. In *Proceedings, First International Symposium on Empirical Software Engineering and Measurement*, 265-274. doi: 10.1109/ESEM.2007.33

Preston, D. (2005). Pair programming as a model of collaborative learning: a review of the research. *Journal of Computing Sciences in Colleges, 20*(4), 39-45. Retrieved from https://www.ccsc.org/publications/

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172. Retrieved from http://www.tandfonline.com/loi/ncse20

Sadasivan, P. (2005). Efficacy of pair programming in completing programming tasks. *Journal of Applied Computing and Information Technology, 3*(1).

Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering, 37*(4), 509-525. doi: 10.1109/TSE.2010.59

Simon, B., & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. *Journal on Educational Resources in Computing (JERIC), 7*(4), 1-28. doi: 10.1145/1316450.1316455

Smith, B., & Delugach, H. S. (2010). Work in progress—Using a visual programming language to bridge the cognitive gap between a novice's mental model and program code. *Proceedings 2010 IEEE Frontiers in Education Conference (FIE)*. doi: 10.1109/FIE.2010.5673502

Somervell, J. P. (2006). *Pair programming: Not for everyone?* Retrieved from https://pdfs.semanticscholar.org/6779/b47ba3745c44f5a164d53fd2b6bc70ca4897.pdf

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE), 13*(2), 1-31. doi: 10.1145/2483710.2483713

Staarman, J. K., Krol, K., & van der Meijden, H. (2005). Peer interaction in three collaborative learning environments. *Journal of Classroom Interaction, 40*(1), 29-39. Retrieved from http://www.jciuh.org/

Tan, S. C., Yeo, A. C. J., & Lim, W. Y. (2005). Changing epistemology of science learning through inquiry with computer-supported collaborative learning. *The Journal of Computers in Mathematics and Science Teaching, 24*(4), 367-386. Retrieved from https://www.learntechlib.org/p/5760/

Topping, K. J. (2005). Trends in peer learning. *Educational Psychology, 25*(6), 631-645. doi: 10.1080/01443410500345172

Traynor, D., & Gibson, P. (2004). Towards the development of a cognitive model of programming: a software engineering approach. In, *Proceedings of the 16th Workshop of Psychology of Programming Interest Group*. Retrieved from http://www.ppig.org/workshops/ppig-2004-16th-annual-workshop

Utting, I., Tew, A. E., McCracken, M., Thomas, L., Bouvier, D., Frye, R., …Wilusz, T. (2013). *A fresh look at novice programmers' performance and their teachers' expectations.* Proceedings of the ITiCSE Working Group Reports, Conference on Innovation and Technology in Computer Science Education. doi: http://dx.doi.org/10.1145/2543882.2543884

Vanhanen, J., & Korpi, H. (2007). Experiences of using pair programming in an agile project. *Proceedings, 40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007*. doi: 10.1109/HICSS.2007.218

Vanhanen, J., Lassenius, C., & Mantyla, M. V. (2007). Issues and tactics when adopting pair programming: A longitudinal case study. *Proceedings, International Conference on Software Engineering Advances, ICSEA 2007*. doi: 10.1109/ICSEA.2007.48

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC), 4*(1), 1-8.doi: 10.1145/1060071.1060075

Willey, K., & Gardner, A. (2010). Investigating the capacity of self and peer assessment activities to engage students and promote learning. *European Journal of Engineering Education, 35*(4), 429-443. http://dx.doi.org/10.1080/03043797.2010.490577

Williams, L., & Upchurch, R. L. (2001). In support of student pair-programming. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*. doi: 10.1145/364447.364614

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197-212. Retrieved from http://www.tandfonline.com/loi/ncse20

Williams, L. A., & Kessler, R. R. (2001). Experiments with industry's "pair-programming" model in the computer science classroom. *Computer Science Education, 11*(1), 7-20. Retrieved from http://www.tandfonline.com/loi/ncse20

Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education, 54*(1), 168-170. doi: 10.1109/TE.2010.2048328