https://doi.org/10.48009/2_iis_2011_207-216

Volume XII, No. 2, pp 207-216, 2011

PAIR PROGRAMMING AS A TEAM BASED LEARNING ACTIVITY: A REVIEW OF RESEARCH

Silvana Faja, University of Central Missouri, sfaja@ucmo.edu

ABSTRACT

Ability to work in teams has been considered one of the most important learning outcomes of the information systems curriculum. Industry's pair programming methodology has been adopted in the recent years as a form of collaborative learning in academic settings. This study reviews the existing body of research aiming two achieve two objectives. First, analyze adoption of pair programming as a pedagogical method using theories based on team learning research. The team effectiveness model was adapted as a framework for analysis, focusing on the outcomes such as facilitation of student learning and fostering of team skills. Second, this study aims to inform information system educators about the viability of using this technique in programming courses.

Keywords: Pair Programming, Collaborative Learning, Teaching Methods, Team Performance

INTRODUCTION

It is argued that IS educators should be exploring more the adoption of team-based learning activities into IS teaching [33]. Curriculum guidelines for undergraduate degree programs in information system, IS 2010, identified team leadership and collaboration skills as one of the main exit characteristics of IS graduates, emphasizing that it is essential that IS programs prepare their graduates to be effective collaborators [31]. McMurtrey et al. [22] conducted a survey of IT professionals to determine which skills current IT professionals consider the most important for new, entry-level IT professionals. They found that, from a list of 42 important skills that represented IS Core Knowledge, Technical Proficiencies, Business Expertise, and Personal Attributes, IT professionals viewed problem-solving, critical-thinking, and team skills as three most important skills for new IT professionals.

Team work activities are frequently incorporated in upper level courses in Information System programs. However, in introductory programming classes there
is a predominance of emphasis on individual learning activities. Most beginning programming classes do not incorporate collaborative practices. Frequently, the practice of students working together on a programming assignment is considered as "cheating". Learning to program is generally considered to be difficult and challenging for most students, who very often experience high levels of frustration and isolation [27]. Some of the challenges that beginner programmers face may be overcome by allowing students to collaborate with their peers. The pedagogical advantages of student interaction in collaborative construction of knowledge are grounded in the social constructivist perspective of learning. Based on the constructivist pedagogical approach, actual learning takes place when students actively construct their knowledge through social interactions with their peers [34]. Knowledge is discovered and constructed through communication and collective sense making.

One of the pedagogical methods that has been adopted recently in the area of computer science and software engineering is an adaptation of the industry's pair programming. A significant number of studies have investigated different aspects of using this method as a teaching method. This paper reviews the existing body of research aiming to achieve two main objectives. First, provide a framework to analyze adoption of pair programming as a team based activity in information systems courses. Second, present a summary of findings of the current research to inform educators in the Information Systems area about the potential benefits and drawbacks of using pair programming as a learning activity, and it provides suggestions for future research. The remainder of the paper is organized as follows: First, a description of the pair programming method and the theoretical framework used in this study are provided. Next, findings from the literature review are presented based on the theoretical model of team effectiveness. Throughout the paper, the study addresses the need for further research on effectiveness of pair programming.

PAIR PROGRAMMING AND RESEARCH FRAMEWORK

Pair programming is the term used to describe the process where two programmers work together on the same task at one computer. Generally, one programmer is the ”driver” and has control of the mouse and the keyboard. The driver types the code at the computer, and the other programmer is the “navigator”. The navigator observes the work of the driver, looking for errors and considering the algorithm being used for the task. Each programmer takes a turn at being the “driver” and the “navigator”. The two programmers collaborate in designing, coding and reviewing.

Pair programming originated in the industry and is one of the key practices in Extreme Programming (XP) [2]. XP is one of the most popular agile methodologies for systems development with a strong focus on personal interactions among software developers. Wray [40] described four mechanisms that make pair programming successful: 1) pair programming chat, which refers to the ability to talk problems out loud; 2) pair programmers notice more detail. He suggested that pairs should be rotated often (once or twice a day) to avoid pair fatigue, which refers to the fact that with time the two programmers become very similar in the things they notice or do not notice; 3) pair programming helps in fighting poor practices - programmers do not always use the best programming practices. Pair programming can encourage the use of best programming practices due to pair pressure, or the feeling of not wanting to let your partner down; 4) sharing and judging expertise - while working in pairs it becomes easier to judge a programmer’s expertise. As such, the estimates of time and difficulty can be made more accurately by a pair of programmers.

During the last ten years, there has been significant research that seeks to establish the viability of pair programming as a learning activity. However, most research on pair programming in educational settings has paid little attention to theories that have the potential to reveal factors important to the successful adoption of this method. One such attempt has been by Preston [26] who indicated the critical attributes of collaborative learning as a framework to review the research on pair programming. These attributes are: 1) common task; 2) small group learning; 3) cooperative behavior; 4) interdependence; and 5) individual accountability. Mentz et al. [24] incorporated the principles of cooperative learning in their approach of implementation of pair programming in the classroom. However, none of the previous studies have introduced a model or a framework to explain or analyze various aspects of this method. To fill this gap, this study draws from the research on collaborative learning and team performance. Specifically, the team effectiveness model was adapted as a conceptual framework to analyze current research on the use of pair programming in education and to understand its potential as a team learning activity. The team effectiveness model was introduced by Hughes et al. [18] and was applied in the organizational context. The model presented here was based on the original model as well as on an adaptation of the model by a study on team effectiveness in educational settings [12]. In addition to the factors included in those models, the model presented here includes several new factors that were identified by the review of the literature on pair programming.

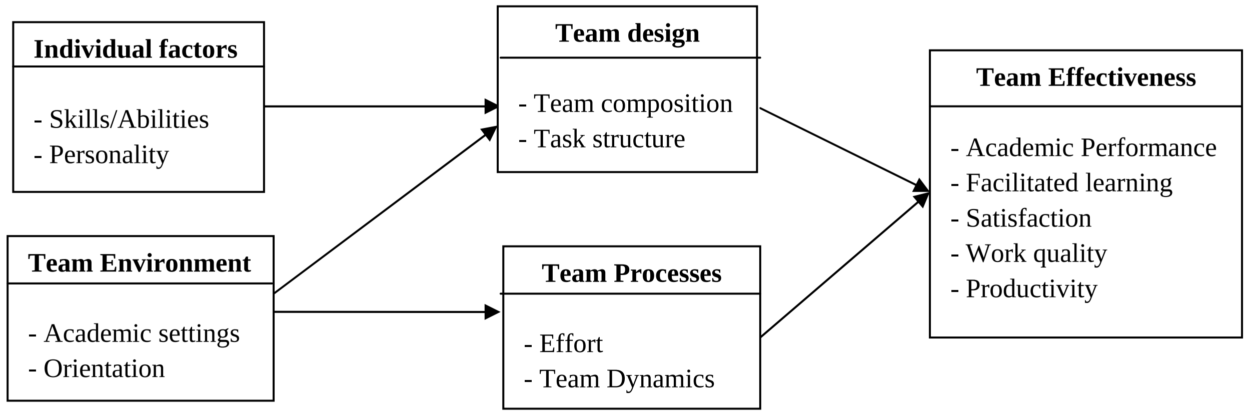

Figure 1. A model of team effectiveness of pair programming as a learning activity

REVIEW OF LITERATURE ON ADOPTION OF PAIR PROGRAMMING AS A LEARNING ACTIVITY

An extensive review of literature was conducted for the purposes of this study to identify relevant empirical studies. Two main criteria were applied during the selection process. First, only studies that used pair programming as defined in the XP methodology were included, excluding studies on other types of team activity. Additionally, studies that focused on other aspects of agile software development such as pair design, or that only described software that support pair programming were not included. Second, studies that used pair programming in higher education settings were included. The following presents the findings from the literature review based on the team effectiveness model presented in Figure 1.

Team Environment

Academic settings. Two themes emerged from the review of the literature: a) application of pair programming in the traditional classroom; and b) a relatively new application of pair programming in distance education. While most of the research on pair programming has focused on co-located or face-to-face pair programming, some studies have examined the adoption of pair programming in distance learning. This second approach is referred to as virtual pair programming or distributed pair programming. Zacharis [42] investigated the effectiveness of virtual pair programming in an introductory Java course suggesting that it can be used successfully. Hanks [14] used distributed pair programming in an introductory programming course. He found that students who used distributed pair programming performed just as well as students that used co-located pair programming. Contrary to these findings, Edwards et al. [11] compared distributed pair programming versus co-located programming and reported that students' satisfaction with distributed pair programming was lower than co-located pair programming. While these studies examined synchronous virtual pair programming, a study by Zin et al. [43] implemented asynchronous virtual pair programming. Their study found that students considered virtual pair programming to be effective, motivating and enjoying, while also suggesting that some synchronous features would be preferable. In addition to factors that have been considered in the traditional classroom, an emerging stream of research in distributed pair programming have addressed the development and evaluation of the various tools and platforms that enable synchronous pair collaboration in the online course settings.

As universities expand their online course offerings, this seems to be an area that may attract future research. Educators are trying to identify learning strategies and activities that can promote student learning in online curriculum. Pair programming can be one of these activities.

Orientation. Pair programming is not simply two people sharing a computer. Students need to take the roles of driver and navigator, they should alternate roles during the course of the activity and both members should review each other's input. The review of the literature shows that the amount of information provided to students about pair programming varies among studies. For instance, Howard [17] had students involved in the study read a description of pair programming in the syllabus. The instructor actively monitored pairs during the first activity to ensure they understood the method. Werner et al. [35] had their students read an article on pair programming before using the method for class activities. In several other studies, the instructor explained the concept of pair programming at the beginning of the course [13, 23, 5, 10]. McDowell et al. [21] and Williams [37] described pair programming to students and had them read an article on pair programming. These variations in the orientation approach may explain some of observations in these studies that students had to be reminded to follow the correct procedure. In one case, because the instructor did not emphasize what was expected from students, it resulted in cases of misunderstanding. For example, students were dividing the task in two parts and working on those parts separately [3].

One of the keys to successful implementation of pair programming is the training of students in pair programming requirements in a supervised setting before assigning them to work in pairs [39]. Mentz et al. [24] showed that conducting intensive training sessions on pair programming versus explanation of pair programming seemed to improve goal setting and fostered positive interdependence.

Individual Factors and Team Composition

Skills/Abilities. Team composition is an important factor that affects collaboration and performance of teams. In the reviewed studies, various methods were implemented for forming pairs. These methods can be categorized as: 1) matching pairs vs. random pair assignment and 2) same partners throughout the course vs. changing partners. Some studies matched pairs based on their academic performance by using grades earned by students in previous course activities. Choi et al. [9] and Zacharis [42] paired students with similar grades. Williams [37] paired them based on the GPA, while Chigona and Pollock [8] paired a strong student with a weak one. Howard [17] applied various pair formation approaches: assigned pairs through random selection, paired students with different Myer-Briggs personality types, and matched compatible students based on the instructor's observation. In some other studies, pairs were assigned randomly. Hahn et al. [13] assigned pairs randomly and ensured that they had different partners for each lab assignment. McDowell et al.[21] assigned students to pairs based on their preferences or randomly if they did not have any preference and they remained with the same partner. Muller [25] and Mendes [23] also assigned pairs randomly. Braught [5] rotated pairs every three to four lab sessions. Initially, pairs were assigned randomly, later they were matched based pairing students with similar course performance. Carver et al. [7] also followed a similar approach. They rotated pairs three times during the semester.

Several of the studies reviewed suggested that pairing students of similar ability is more beneficial for student learning. Cliburn [10], based on his classroom experience, suggested that the best strategy is to pair students at near or the same skill level for best outcomes. While VanToll [32] suggested that pair programming works best when pairs are of slightly different skill level. Bevan [3] also suggested pairing based on the skill level. Based on their experience of using pair programming in the classroom, disparity between the experience levels of students in a pair was one of the sources of intra-pair stress. This was especially emphasized for outside activities. Studies showed that if students are paired with less adept partners, they may consider the experience a waste of time and they might complete the assignment on their own and submit it as a combined effort [3]. Except for one study [32], none of the reviewed studies have examined the effect of various skill levels matching during pair formation on team satisfaction and performance in controlled settings. In addition, none of the studies have examined the effect of having the same partner or rotating partners during the semester. In professional settings, "pair jelling" refers to collaborative teams in pair programming, which typically happens when members of the team learn the strength and weaknesses of each other [36]. Future research may investigate this aspect of team composition to determine which approach of structuring this team activity might yield better results in terms of student learning.

Personality. Personality traits make one of the attributes that has been considered to have a significant effect on the results of pair programming among professionals. Salleh et al. [28] indicated that research on pair programming has considered 14 different compatibility factors. However, personality type has been one of the most researched ones. They also found that results have been inconsistent in terms of effects of personality towards pair programming. This was also confirmed by this review of implementation of pair programming in educational settings. Choi et al. [9] investigated the effect of personality factors, communication skills and cognitive style on the pair programming experience; specifically in the levels of satisfaction, compatibility, communication, and confidence. The subjects were undergraduate and graduate students majoring in MIS, IS and IT. They found that personality factors and communication levels had no impact on the pair's satisfaction, communication, and confidence. This is also supported by a study on effects of personality traits on the pair programming performance among professionals [15]. They did not find strong support that personality effects programming performance. Instead, their study showed that expertise and task complexity were stronger predictors of pair programming performance. Carver et al. [7] did not find any effect of personality in their application of pair programming in the classroom settings. Other studies provide evidence that personality traits might be a factor to be considered for effective team formation. Katira [19] found that freshmen students work better with partners of different personality type as measured using Meyer-Briggs type indicator. Sfetsos [30] indicated that pairs of heterogonous types perform better than pairs of homogenous types, had a higher collaboration satisfaction and better knowledge acquisition. This may be another area that deserves further attention. However, it can be argued that in educational settings, matching pairs based on personality types may not be as relevant compared to matching them based on their skill sets for two main reasons. First, administering personality questionnaires at the beginning of the semester or before conducting pair activities may not be feasible or may be time consuming if done on a regular basis. Second, one of the purposes of using pair programming in the classroom is to develop team building skills among information systems students. Similar to the workplace settings, students should learn how to work with team members with different types of personalities and be able to communicate and collaborate with them to achieve the desired results. Behavioral theories and research on group processes and performance may be incorporated in future studies to consider this aspect.

Gender. Gender was considered as a factor by some of the studies reviewed. One study showed that pair programming is especially beneficial for female students [35]. They found that female student who pair programmed in their introductory course were more likely to declare a CS-related major after completing the course. Braught et al. [5] did not find any gender effect on the confidence of the completed work. However, they reported that pair programming reduced the level of frustration with assignments for female students. Same gender pair groups showed higher level of satisfaction, communication, and compatibility [9].

Team Design/Task Structure

Analysis of existing research revealed two aspects of task structure related to pair programming. First, whether pair programming was used to complete in class activities or outside course assignments, and second, task complexity. The review of the literature showed that pair programming was used for both in class activities and outside course assignments. This seems to be a significant factor that effects the ability to generalize the findings of the existing research. Unlike the typical team activity, pair programming has a set of specific roles and settings. As stated earlier, pairs have to use one computer, each member should play the role of the driver and the navigator and they should take turns in these roles. When applied to outside assignments, instructors do not have a lot of control on the actual implementation of the method by students.

In some of the studies, pair programming was used in closed computer lab environment for lab assignments [5, 7, 8, 23]. In other studies, pair programming was implemented for both outside programming assignments and for some in-class laboratories [17, 29, 37]. Howard [17] had students complete the first lab and the two first programming assignments individually to ensure that each student knew how to use the software. McDowell et al. [21] used it for outside assignments. A common concern reported by studies that used pair programming was that students found working on outside assignments more difficult due to scheduling challenge. A number of studies found that students perceived scheduling to be a big problem in pair programming [3, 17, 29, 35]. A study by Mentz et al. [24] suggested that pair programming works better for work completed in class under controlled circumstances compared to assignments outside the classroom.

Task complexity was one of the factors that had not received significant attention by research. Studies conducted among professionals have considered task complexity as a potential factor that might affect team performance. A meta-analysis on use of pair programming in general, showed that pair programing is faster when task complexity is low, and it yields solutions of higher quality when task complexity is high [16]. Another study indicated that task complexity was not found to impact performance with respect to software quality of pairs versus individual programmers [1]. However, while task complexity may not be an issue for experienced programmers, it might effect novice pairs in educational settings. Future research might consider the levels of task complexity that are more suitable for optimal learning experience and team performance. This may assist educators in choosing the right activities that pair programming can be applied to.

Team Process

Effort. Like any other team activity, one concern educators have about using pair programming is that the stronger student will do most of the work. Social loafing, a well-known concept in team work, could also happen when using pair programming. Social loafing occurs when individuals working together in groups put less effort than when they work individually [1].

Williams et al. [39] have extensively applied and researched pair programming as a teaching methodology and have provided a set of guidelines for the implementation of pair programming. One of the guidelines suggests that teaching staff must actively engage in the management of pair interactions. For example, they have to ensure that students are switching roles periodically. One study notes that some teams would never have changed drivers had they not been encouraged to do so [17].

Another guideline suggested by Williams et al. that is relevant in the context of team effort is to conduct peer evaluations to ensure equal participation of pair members. Importance of peer evaluation to the successful implementation was also supported empirically. Mentz et al. [24] had better results in their use of pair programming after they incorporated peer assessment in their study. Hahn et al. [13] focused on assessment strategies for pair programming. They applied a combination of self-assessment, peer-assessment and facilitator-assessment for each individual participant. Their study showed that as students became more familiar with this assessment approach, they became more realistic in assessing their own and their partner's work. It also increased the commitment of both pair members, as indicated by their individual assessment. They suggest that self, peer, and facilitator assessment combination might be a more reliable assessment strategy for pair programming. Cliburn [10] also reported about the importance of peer evaluations and also changing partners as a way to deal with "free-riders". Furthermore, Williams et al. [39] suggested that evaluating effort becomes even more important when students are pair programming outside the classroom setting.

Team dynamics. Students reported a positive attitude toward collaboration and communication [17]. Pair programming helped them develop teamwork skills [11]. Cliburn [10] also showed that students reported that pair programming helped them develop better interaction skills. Participants in his study expressed that they could see that their way of approaching a problem was not the only way.
Williams et al. [37] used the term "pair pressure" to refer to the effect of members in the pair keeping one another motivated. Working in teams puts pressure on each member to perform and keeps them focused on the task [24]. Having someone constantly observing one's work, serves as an incentive for students not to skip important steps of the process. It also helps students in motivating each other to stay on track, especially with outside assignments [42]. Wray [40] stated that, among professional programmers, pair programming can encourage the use of best programming practices due to pair pressure, which they tend to neglect at times. This is especially important for students as they are actually learning about these practices. Beginners to programming frequently forget to incorporate these practices in their programs. Pair pressure might help them get used to applying good programming practices.

Another process that would fall under team dynamics is what Williams et al. [37] call "pair relaying". This is the effect of having two people working to resolve a problem together. Williams et al. [37] indicated that students reported that it helped them understand things better when they had to explain it to another student. Canfora et al. [6] showed that working in pairs can speed up the process of knowledge building and transfer. Knowledge transfer is considered one of the main benefits of pair programming in both professional and educational setting. This is supported by findings of studies presented in the next section of this paper.

Team Effectiveness Outcomes

Most of the previous research showed that pair programming has benefits over solo programming. For the purposes of this study, first the major outcome categories as well some subcategories were identified in the literature. The findings from each study were then put in each outcome category to indicate whether a study's findings indicate a positive or negative/neutral effect on the outcome. The summary of the findings is presented in Table 1.

Even though for some of the outcomes research shows contradictory findings, there are several themes that clearly emerge from the review. Student satisfaction of this type of course activity was a common theme in almost all of the studies that were included in this literature review. Many studies also showed positive findings with regard to academic performance and learning. Even if students that pair program do not show significantly better grades than those who work individually, none of the studies indicate that pair programming effects their academic performance negatively. This may be important to know for those educators who may have been considering the use of this method in the classroom, but may have been worried that this might effect negatively the individual student learning. The majority of the studies also showed that pair programming improves the quality of the programs.

Positive findings	Negative/Neutral Findings
<u>Academic performance/Learning</u> -Higher academic achievement when enforcing individual accountability [24] -Assignment grades for pairs were higher than the solo programmers [8], [38], [23] -Pair programming helped development of individual programming skills for lower SAT students [5] -Students who pair programmed in introductory course were more likely to attempt subsequent programming courses [35] -Students who pair programmed were more likely to remain in or select a computing related major [7] -Students who used pair programming were more likely to complete the course and had higher pass rates [21], [5], [38], [23] -Pair Programming helped students better understand programming concepts [17] -Process of learning is faster for pairs compared to individuals [41] -More ideas and alternative solutions are explored with pair programming [29] -Pairs resolve more problems on their own, less help from the instructor [5] -Higher self-evaluated learning outcome [4], [7], [10], [11] -Students feel this is a good way to learn programming [29]	- Assignment grades for pairs were not significantly different from solo students [42] -Course pass rates not significantly different between paired and solo programmers [21], [35] -Individual exam grades (a measure of their knowledge of course material) were not different for those who used pair programming vs. those who did not [21], [38] -No impact on knowledge transfer among students [8] -No differences on acquisition of individual skills between students with higher SAT levels that pair programmed and those who did not [5] -Students felt they understand their programs better when they work by themselves [29] -Pair programmers did not perceive to have a gained a better understating of the concepts by explaining them to their partner [7]
<u>Students Perceptions</u> • **Enjoyment** -Students enjoyed working in teams [1], [4], [8], [10], [17], [21], [24], [37], [41], [42] -High satisfaction with pair programming [11] • **Knowledge transfer** -Students perceive they learned more by working with a partner than they would have by working alone [7], [41] -Reduced student frustration [5], [17], [29], [37] • **Confidence** - Students who paired reported higher confidence in the correctness of program solutions compared to individual programmers [1], [5], [21], [35], [37] • **Social aspects** -Enjoyed meeting fellow students [29] -Students felt pair programming made them better at working with others [10]	
<u>Work quality</u> -Higher software quality [37], [41], [42] -Pairs produced better programs [8], [21] -Code produced by pairs was easier to read and understand [4] -For simple problems, pair programming lead to fewer mistakes [25] -Programs developed by pairs passed more test cases [36]	-Pair Programmers had as many algorithm mistakes as individual programmers [25]
<u>Productivity</u> -Pair students were more productive than the individual programmers [42] -Students perceived pair programming to be more productive [8], [37] -Pair programmers completed tasks in a shorter time [28], [41] -Working in pairs helped find errors earlier [4]	-Pair programming may or may not take less time [29] -Time for pairs and individuals was the same [37] -Pairs produced nearly the same amount of code as individual programmers [4]

Table 1. Summary of Research Findings on the Effectiveness of PP as a Teaching Tool

Research on the impact of pair programming on productivity is inconclusive. One reason might be that in many studies pair programming was used during fixed timed laboratory sessions, where it was difficult to establish whether pairs or individuals required less time. This is in line with research findings on pair programming in general that indicate that collaborating pairs do not exceed the performance of its best member working alone [1]. A meta-analysis by Hannay et al. [16] showed only a small significant effect of pair programming on software quality and a negative effect on effort. However, compared to industrial settings where productivity and quality are the main concerns associated with the adoption of pair programming, educators are more concerned with learning outcomes.

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

As a discipline, IT is constantly changing and it is relevant to introduce students to some of the newer software development methods by applying them in the classroom. In addition, most modern organizations require individuals to work in teams to perform their task. Pair programming could be a promising teaching method that allows IS educators to incorporate both these aspects. Pair programming is also considered beneficial to faculty because it reduces workload. First, there are half as many assignments to grade and second due to knowledge transfer taking place between the pair members, student pairs are more self-sufficient and rely less on the instructor to help them with issues that arise during the assignments. Students discuss their assignments with each other, thus becoming effective teaching resources for each other.

Most of the studies indicated that pair programming has been mostly used in Computer Science courses and its adoption by IS courses is in its early stages. This paper conducted an extensive review of the literature aiming two achieve two objectives. First, provide a theoretical framework to analyze the current research on pair programming and to guide the future research in this area. This is the first study to adopt a team effectiveness model to analyze pair programming in academic settings. Second, to inform information systems educators about the findings of the research on the implementation and the results of using pair programming as a learning activity.

The research showed several benefits of using pair programming in academic settings such as enhanced learning, greater confidence in work quality, higher problem solving skills, enhanced interaction skills, and improved team building skills. Students also receive greater satisfaction when they work in pairs. The case for the use of pair programming in classroom seems to be compelling.

The study also indicated several areas for future research. Future studies can examine the effects of team design on the effectiveness of pair programming. Specifically, areas of interest would be to study the effects of task structure (in class activities versus outside assignment) on learning and the level of task complexity that is most suitable for pair programming. Team composition and its effects on team effectiveness is another suggested future area of research. Most of the previous studies on pair programming have dealt with Computer Science students, only a few studies have used Information Systems students. There are important differences in the skills of the students in these two disciplines as well as programming requirements. Therefore, adoption of pair programming and study of the results may be a future area of research for educators in Information Systems area.

REFERENCES

1. 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),* 99-118.
2. Beck, K. (2000) *Extreme Programming Explained,* Reading, MA: Addison-Wesley.
3. Bevan, J., Werner, L., McDowell, C. (2002). Guidelines for the Use of Pair Programming in a Freshman Programming Class. *Proceedings of the 15th Conference on Software Engineering Education and Training,* 100-107.
4. Bipp, T., Lepper, A., & Schmedding, D. (2008). Pair Progamming in Software Development Teams – An Emprirical Study of its Benefits. *Information and Software Technology, 50,* 231-240.
5. Braught, G., Wahls, T., & Eby, L. M. (2011). The Case for Pair Programming in the Computer Science Classroom. *ACM Transactions on Computing Education, 11*(1), Forthcoming.

6. Canfora, G., Cimitile, A., & Vissaggio, CA. (2004). Working in Pairs as a Means for Design Knowledge Building: An Empirical Study, *12th IEEE International Workshopon Program Comprehension*, 62-68.
7. Carver, J.C., Henderson, L., He, L., Hodges, J., & Reese, D. (2007). Increased Retention of early Computer Science and Software Engineering Students usign Pair Programming, *20th Conference on Software Engineering Education and Training,* 115-122.
8. Chigona, W., & Pollock, M. (2008). Pair Programming for Inforamtion Systems Students New To Programming: Students' Experiences and Teachers' Challenges. *PICMET 2008 Proceedings,*1587-1594.
9. Choi, K.S.., Deek., F.P. & Im., I. (2009). Pair Dynamics in Team Collaboration. *Computers in Human Behavior, 25(4),* 844-852.
10. Cliburn, D. C. (2003). Experiences with Pair Programming at a Small College. *Journal of Computing Sciences in Colleges, 19*(1), 20-29.
11. 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.
12. Grzeda, M., Haq, R., & LeBrasseur, R. (2008). Team Building in an Online Organizational Behavior Course. *Journal of Education for Business, May/June,* 275-281
13. Hahn, J., Mentz, E., & Meyer, L. (2009). Assessment Strategies for Pair Programming. *Journal of Information Technology Education*, *8*, 273-284.
14. Hanks, B. (2008). Empirical Evaluations of Distributed Pair Programming. *International Journal of Human-Computer Studies*, *66,* 530-544.
15. Hannay, J. E., Arisholm, E., Engvik, H., & Sjoberg, D. I. (2010). Effects of Personality on Pair Programming. *IEEE Transactions on Software Engineering, 36*(1), 61-80.
16. Hannay, J.E, DybaT., Arisholm, E., & Sjoberg D.I.K. (2009). The Effectiveness of Pair Programming: A Meta-Analysis. *Information and Software Technology, 51(7),* 1110-1122.
17. Howard, E. V. (2007). Attitudes on Using Pair- Programming. *Journal of Educational Technology Systems, 35*(1), 89-103.
18. Hughes, R.L., Ginnett, R.C., & Curphy, G.J. (1993). *Leadership: Enhancing the Lessons of Experience,* Burr Ridge, Il: Richard Irwin
19. Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S., & Gehringer, E. (2004). On Understading Compatibility of Student Pair Programmers. *ACM Technical Symposium on Computer Science Education,* pp.7-11.
20. Matzko, S., & Davis, T. (2006). Pair Design in Undergraduate Labs. *Journal of Computing Sciences in Colleges, 22*(2), 123-130.
21. McDowell, C., Werner, L., Bullock, H.E., Fernald, J. (2006). Pair Programming Improves Student Retention, Confidence, and Program Quality. *Communication of the ACM,49(8),* 90-95.
22. McMurtrey, M.E., Downey, J.P., Zeltmann, S.M., & Friedman, W. (2008). Critical Skill Sets of Entry-Level IT Professionals: An Emprirical Examination of Perceptisons from Field Personnel. *Journal of Information Tecnology Education, 7,* 101-120.
23. Mendes, E., Al-Fakhri, L.B., & Luxton-Reilly, A. (2006). A Replicated Experiment of Pair-Programming in a 2nd year Software Development and Design Computer Science Course. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 108-112.
24. Mentz, E., van der Walt, J., & Goosen, L. (2008). The effect of incorporating learning principles in pair programming for student teachers. *Computer Science Education, 18*(4), 247-260.
25. Muller, M. (2007). Do Programmers Pairs Make Different Mistakes than Solo Programmers? *The Journal of Systems and Software, 80,* 1460-1471.
26. Preston, D. (2005). Pair Programming as a Model of Collaborative Learning: A Review of the Research. *Journal of Computing in Small Colleges, 20*(4), 39-45.
27. Rogerson, C., Scott., E. (2010). The Fear Factor: How it Affects Students Learning to Program in a Tertiary Environment. *Journal of Inforamtion Technology Education,9*, 148-171.
28. Salleh, N., Mendes, E., & Grundy, J. (forthcoming). Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering.*
29. Simon, B., & Hanks, B. (2008). First-Year Students' Impressions of Pair Programming. *ACM Journal on Educational Resources in Computing*, *7*(4), Article 5.

30. Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. (2009). An Experimental Investigation of Personality Types Impact on pair effectiveness in Pair Programming. *Empirical Software Engineering, 14,* 187-226.
31. Topi, H., Valacich, J. S., Wright, R.T., Kaiser, K.M., Nunamaker, J.F., Sipior, J.C., de Vreede, G.J. (2010). IS2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *ACM Curricula Recommendations,* Retrieved April, 2011, from http://www.acm.org/education/curricula-recommendations.
32. Van Toll, T., Lee, R., & Ahlswede, T. (2007). Evaluating Usefulness of Pair Programming in a Classroom Setting. *Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Systems,* 302-308.
33. Van Der Vyver, G., & Lane, M. (2003). Using a Team-based Approach in an IS Course: An Empirical Study. *Journal of Information Technology Education, 2*, 393-406.
34. Vygotsky, L. (1978). *Mind in Society: The Development of Higher Psycological Processes.* Cambridge, MA: Harvard University Press.
35. Werner, L., Hanks, B., & McDowell, C. (2004). Pair-Programming Helps Female Computer Science Students. *ACM Journal of Educational Resources in Computing, 4*(1), 1-8.
36. Williams, L., Kessler, R.R., Cunningham, W., & Jeffries, R. (2000). Strenghtening the Case for pair Programming. *IEEE Software, July/August,* 19-25.
37. Williams, L. & Kessler R. (2001). Experiments with Industry's "Pair-Programming" Model in the Computer Sciene Classroom. *Computer Science Education, 11(1)*,7-20.
38. Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Compute Science Course. *Computer Science Education, 12(3),* 197-212.
39. Williams, L., McCrickard, D. S., Layman, L., & Hussein, K. (2008). Eleven guidelines for Implementing Pair Programming in the Classroom. *Agile 2008 Conference*, 445-452.
40. Wray, S. (2010). How Pair Programming Really Works. *IEEE Software, January/February*, 50-55.
41. Xu, S., & Rajlich, V. (2005). Pair Programming in Graduate Software Engineering Projects. *35th IEEE Frontiers in Education Conference,* FIG7-FIG12.
42. 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.
43. Zin, A.M., Idris, S., & Subramaniam, N.K. (2006). Implementing Pair Programming in E-Learning Environment. *Journal of Information Systems Education, 17(2),* 113-117.