Guidelines for Implementing Pair Programming in Introductory CS Courses: Experience Report

Alex Radermacher, Gursimran S. Walia, Sameer Abufardeh, Oksana Myronovych

Department of Computer Science
North Dakota State University
Fargo, ND 58108
{alex.radermacher, gursimran.walia, Sameer.abufardeh, Oksana.myronovych}@ndsu.edu

ABSTRACT

Pair programming has been shown to be an effective method of improving the learning outcomes of students in introductory computer science courses. However, much of the existing literature related to pair programming does not focus how to effectively implement pair programming. Researchers studying multiple aspects of pair programming have conducted several empirical studies at our university over the past two years. During this time, researchers received valuable feedback from course instructors about the effects of implementing pair programming in their introductory computer science courses. These instructors also expressed concerns about the use of pair programming in their courses. These include being able to ensure equal participation from pair members and not being able to assess individual learning outcomes effectively. This paper reports these concerns and uses empirical evidence from the pair programming studies to provide guidelines for the effective use of pair programming in beginning programming courses. Based on the experiences at our university along with those experiences of other researchers, we provide recommendations for course design when using pair programming.

Keywords

Pair programming, CS1, CS2, course design.

1. INTRODUCTION

Pair programming began as part of Extreme Programming (XP), an agile development methodology proposed by Beck, et al. to help software developers be more productive [1]. Initial research showed that pair programming was effective when used by professionals [17]. Eventually researchers conducted studies that found the pair programming to be effective when used by novice programmers as well [13, 25].

Over the past two years, numerous studies related to pair programming has been conducted in introductory computer programming courses at our university. This research has produced valuable empirical evidence related to student learning, pairing strategies, and other aspects of pair programming (e.g., student-teacher interactions, students' mental model consistency etc). Throughout these studies, the primary researchers have worked closely with the three instructors who taught the classes in which the studies were conducted. These instructors have provided valuable feedback regarding the implementation of pair programming in introductory computer science courses to help researchers better understand the results, and on how to address the problems faced while designing a programming course when using pair programming.

Although pair programming has been demonstrated to be an effective method of improving student learning, there are several considerations that must be taken into account when using pair programming in an introductory computer science course. Examples include as how students will be paired and how to ensure that both pair members are contributing. Previous research has already provided many useful suggestions including strategies for effectively pairing students [5, 26], but there is a general lack of empirical evidence to guide course design when using pair programming.

This paper reports on instructor concerns regarding the use of pair programming and provides data from empirical studies conducted at several universities (including the studies conducted at the researchers' university) in order to provide a set of recommendations that can be used to most effectively implement pair programming in an introductory computer science course. Similar reports have been published by other authors in the past [3, 27, 28], but these papers mainly focus on problems experienced at a single university or were published several years ago and do not take recent advancements or discoveries into account.

The remainder of this paper is constructed as follows: Section 2 provides a brief description of pair programming and presents related work that forms the basis of this study. A description of the empirical studies on pair programming used to identify issues and solutions is given in Section 3. Section 4 lists instructor feedback and concerns which are validated by empirical evidence. Recommendations for remedying these problems are presented in section 5. Conclusions to this study are given in section 6. Finally, Section 7 briefly describes the ongoing and future research studies in pair programming at researcher's university.

2. BACKGROUND

Pair programming describes a programming technique where two programmers work together on the same programming task. One programmer, referred to as the driver, actively controls the keyboard and writes code, while the other programmer, referred to as the navigator, watches the driver for mistakes and helps him develop the code. The pair programmers work collaboratively while frequently exchanging the roles of the driver and the navigator.

The term pair programming was first used by Beck and his colleagues in *Extreme Programming Explained: Embrace Change* [1]; however, the effects of collaborative programming [17] as well as the use of collaborative learning in introductory computer science courses have previously been studied [21].

Substantial empirical studies were performed by Williams, et al. at NCSU (North Carolina State University) [24, 25] and McDowell,

et al. at UCSC (University of California Santa Cruz) [3, 13, 14] that provided significant evidence to support the use of pair programming in introductory computer science courses, showing that it improved student performance on programming exercises and improved their enjoyment of programming. These initial findings provided good support for the use of pair programming in introductory computer science courses and motivated further research into pair programming.

Since then, additional empirical research has studied other aspects of pair programming such as the effects on student retention [7, 15], the effects of pair programming on female computer science students [2], and factors related to creating effective student pairs [5, 26].

Recent research conducted at NDSU (North Dakota State University) has provided further empirical support for the effectiveness of pair programming [20], studied newer alternative methods for pairing students that improved the pair performance [18], examined the effects that the use of pair programming has on student-instructor interactions during laboratory sessions [19], and are currently evaluating the effects of pair programming on mental-model consistency in introductory computer programming students.

The following section provides a more in-depth description and analysis of pair programming studies from which information has been extracted to support the research findings presented in later sections of the paper.

3. RELATED EMPIRICAL STUDIES

This section presents a more detailed description of the empirical studies on pair programming in an educational context and the major results from those studies. This is done to highlight that the identified issues with pair programming implementation (as discussed in Section 4) and the proposed solutions to those issues (as discussed in Section 5) are based on a large collection of empirical evidence across different research sites.

Inclusion and Exclusion Criteria: Papers included in the analysis were those that reported the results from an empirical investigation of pair programming or using the results of previous empirical studies to form conclusions. Some studies were excluded from the analysis as they included participating subjects who worked together remotely or because the participating subjects were not students. Also, those studies whose findings were unclear or ambiguous were excluded from the analysis.

Many papers present results or analysis based on the same data sets or share common authors. Because of this, papers are grouped according to the university site where the research was conducted.

3.1 NCSU Studies

A large number of studies related to pair programming have been conducted at NCSU over the course of the last decade [12, 16, 22, 25, 26, 27, 28].

The initial studies conducted at NCSU were part of an NSF (National Science Foundation) supported longitudinal study [25]. During the Fall 2001 semester, two sections of the introductory programming course were included in the experiment. The study required 69 subjects in one section to work individually and served as a control group, whereas 44 subjects in the other section served as the experimental group and used pair programming. The focus of this study was to gain a better understanding of how pair programming affected student learning and performance. Quantitative results from the study indicated that pair programming improved student performance on programming exercises, but that pair programming did not improve student performance on exams. The study also included qualitative findings related to student behavior when using pair programming and role of instructors during laboratory sessions. One of the most notable reported observations was that when pair programming was used in programming labs, students appeared to spend less time waiting for assistance from an instructor.

A follow-up study that added additional data points was conducted during Spring 2002 semester [16]. An additional 102 subjects worked individually as part of a control group and 280 subjects used pair programming. As with the previous study, only the results of freshman and sophomore students who took the course for credit were analyzed. The results of this study indicated that pair programming was effective for improving the learning outcome for non-computer science majors, but did not have a significant impact on computer science majors. The study also provided further support that the use of pair programming did not improve student performance on exams. The study also provided qualitative analysis that stipulated that random pairing led to incompatible pairs in a small percentage of cases and that lab instructors should monitor pairs to ensure that pair programming is being used properly.

Katira, et al. conducted a study during the Fall 2002 and Spring 2003 semesters to study factors influencing pair compatibility [11, 12]. The study analyzed 564 students from three courses for pair compatibility based on personality, skill level, perceived competence, and self-esteem. There was moderate support to indicate that partners with different personality types were more compatible and that pairs were more effective if both partners believed that they had similar levels of technical competence. However, the researchers did not find that the pair compatibility was dependent on actual skill level (measured by performance on a midterm exam).

Shrikanth, et al. reported on student and instructor perceptions of the viability of pair rotation [22]. Subjects in this study were 240 students in a CS1 course and 170 students in an undergraduate SE (software engineering) course, which were a subset of the subjects from the experiment conducted by Katira [11]. All participating subjects used pair programming and were assigned four partners over the course of the semester. The researchers concluded that pair rotation had benefits for both students and instructors, but also pointed out several issues that could arise when constantly adjusting student pairs. One prominent problem identified by the authors is that mandatory pair rotation could break apart a functional pair and result in new pairs that were less effective. The researchers noted that because only a small number (17) of subjects from the SE course completed the survey, the results may be slightly biased.

Williams, et al. released a follow-up study to further examine compatibility of student pair programmers [26]. This study used data collected by Katira [11] but included 133 additional subjects in a software engineering course in the Fall 2004 and Spring 2005 semesters. Researchers examined the impact that learning styles, work ethic, and time management skills had on pair compatibility. Results of the study indicated that pairing Myers Briggs sensors with Myers Briggs intuitors, classifications (used to describe how

people perceive and understand information) produced more compatible pairs and that pairing students with similar work ethics also produced more compatible pairs.

Williams also released two papers that provided a list of lessons learned from using pair programming [27] and suggested guidelines for using pair programming [28]. These papers were based on the results of previous studies conducted at NCSU. A few of the major points not discussed in previous work include providing students with training in order to help them understand how to effectively use pair programming. Both papers also expressed the importance of ensuring that instructors or other teaching staff are engaged in managing pair interactions.

3.2 UCSC Studies

Several early studies of pair programming were conducted at USCS by McDowell and colleagues [3, 13, 14, 15].

The initial studies conducted at UCSC evaluated the effectiveness of pair programming on the performance and retention of female students in computer science or related fields [13, 15]. Subjects for the study were students enrolled in two different sections of the introductory programming course during the 2000 – 2001 academic year. In this study, 172 subjects from one section completed programming assignments in pairs, while 141 subjects from the other section work independently. The results of this research indicated that pair programming improved student grades on programming exercises, but did not improve student grades on exams. The authors also noted that the use of pair programming may lead to fewer students dropping the course, but did not have a sufficient number of data points to determine if this result also applied specifically to female students.

McDowell, et al. reported on another study focused on examining the effects of pair programming on student persistence, perception, and performance [14]. Subjects in this study were 555 students enrolled in the introductory programming class in the 2000 – 2001 academic year, and also included the subjects from the previous study. The major findings of this study were that students who use pair programming indicated that they were more confident in their work, were more satisfied with programming, and had greater enjoyment than students who worked independently.

Beven, et al. released an experience report, providing guidelines and suggestions for successfully implementing pair programming [3]. The recommendations were based on lessons learned and issues experienced with implementing pair programming during the 2000 – 2001 academic year. The researchers reported the main sources of difficulty were due to student scheduling conflicts and large disparities in skill level. The authors of the study also presented suggestions for course design (e.g. designing lab exercises that can be completed with minimal out-of-class time requirements.) and pair arrangement, indicating that students of somewhat similar ability should be paired together.

3.3 NDSU Studies [Researchers' Site]

Recently, a series of studies related to pair programming were conducted at NDSU [18, 19, 20].

Radermacher et al., reports on the results of two different studies conducted during the Spring 2010 semester [18]. Subjects in the first study were 35 students enrolled in one section of the CS1 course and the second study included 39 students enrolled in two sections of the CS2 course. Subjects in the CS1 course were split into two groups, one which used pair programming and one that did not; whereas the subjects in the CS2 course were paired based on declared major. Researchers reported that subjects from both the CS1 and CS2 courses indicated that they felt pair programming improved their understanding of programming concepts. Another major result indicated that pairing a computer science (CS) student with a non-computer science (nonCS) student produced less compatible pairs as compared to CS-CS pairs and nonCS-nonCS pairs.

Radermacher et al., reported another empirical study that investigated the effects of pair programming on student-instructor interactions during programming laboratory sessions [19]. Subjects in this study were 44 students enrolled in one section of the CS1 course and 53 students enrolled two sections of the CS2 course during the Fall 2010 semester. Subjects in the CS1 course alternated between using pair programming and working individually during lab sessions, whereas subjects in the CS2 course only used pair programming. Researchers monitored these lab sessions, marking the number of questions asked, how long it took before an instructor could address the subject's question, and how long the instructor spent interacting with the subject. Results of the study indicate that when pair programming is used, students spend less time waiting for assistance from an instructor and spend more time interacting with the instructor, likely due to a decrease in questions related to syntax errors or other minor problems.

An ongoing experiment at NDSU investigated the effects of pairing subjects based on their mental model consistency levels (ranging from highly consistent to highly inconsistent) at the beginning of the semester to evaluate changes in the students' mental model consistency and their programming performance [20]. The initial evidence suggest that such a pairing strategy can be an effective way if previous performance data is not available and that certain mental-model-based pairing arrangements (and not all) are more effective in migrating students towards greater consistency and resulted in better performance on exams.

3.4 Dickinson College Studies

Braught, et al. reported a study examining the effects of pairing students based on ability [5]. Subjects in this study were 259 students enrolled in 13 different sections of an introductory programming course taught between 2005 and 2008. 142 subjects were pair based on ability, 41 were paired randomly, and 72 were not paired. Subjects in this study completed weekly programming assignments and completed 5 exams, two of which were programming exams where students produced code to complete a program. The results from this study indicate that students in the lower quartile who were paired by ability performed better on programming exams than students in the lower quartile who were paired randomly or not paired at all. This contrasts earlier research findings from studies conducted at NCSU and UCSC, which did not find any significant improvement for students on exams when using pair programming.

Braught, et al. also examined how the results of their studies compared with the results from several previous studies conducted by other researchers [6]. The results of this report provided additional support for the conclusion that pair programming provided benefits for students at lower SAT levels. The researchers also provided qualitative data supporting the

conclusion that using pair programming in programming labs resulted in fewer questions related to syntax issues or other lower-level problems.

3.5 University of Sussex Studies

Chaparro, et al. examined factors that affected student perceptions of the effectiveness of pair programming [9]. Subjects in the study were 80 post-graduate students enrolled in an object oriented programming course during the Fall 2004 semester. Researchers found that subjects preferred to work with someone who was similarly skilled or more skilled than they were and that subjects felt that pair programming was not useful if the programming task was not challenging. Observational evidence also suggested that when a large skill disparity existed between subjects, the more skilled subject would take control and relegate the less skilled subject to a more passive role.

4. PROBLEMS AND CONCERNS WITH THE USE OF PAIR PROGRAMMING

This section details problems encountered by instructors when using pair programming in an introductory computer science course. Both issues which have been encountered at NDSU and issues which have been described in previous research papers are considered. Although these problems and concerns are broken into five different categories, there is often a relationship between them, such that issues experienced in one area often exacerbate problems that are experienced in another area.

4.1 Individual Assessment

Difficulties assessing individual learning and ensuring that both partners are benefiting from the use of pair programming is a common issue with pair programming. One of the CS1 instructors at our university also expressed concerns that several students, who he felt would not have normally been able to pass the class, had been able to pass the class because their partner was able to help carry them. Table 1 shows the percentage of students passing the course when pair programming was used and when students worked individually based on eight semesters of historic data for this instructor's class.

This shows a large increase in the number of students who were able to pass the course. The results are even more skewed than they appear as the drop-rate for the course during the Fall 2010 semester was slightly more than 8% compared to the historical drop-rate of approximately 18%. Bevan, et al. had also reported that students were willing to submit assignments that only one of the students had completed [3]. Williams, et al. also expressed similar concerns when they discovered instances of students who performed well on pair programming exercises, but scored poorly on exams, suggesting that one partner may have been completing most or all of the work on the programming exercises, a phenomenon that was also observed by our own instructors. Another of our instructors indicated that this may have been an issue as a large number of laboratory assignments required substantial out-of-class work in order to complete and that there was no easy way to ensure that both partners had contributed equally.

4.2 Subject Pairing

Another issue regarding pair programming is ensuring that individuals are paired effectively, especially early in the semester. Previous research has indicated that it is not feasible to match

Table 1. Pre and Post Pair Programming Result Comparison of Student Grades

Percentage of Students Receiving Grade or Better	*Historical Average*	*Fall 2010*
Percentage of students with 'C' or better grade.	68.87%	74.68%
Percentage of students with 'D' or better grade .	80.21%	93.57%

students based on existing available measures such as SAT score, GRE, or GPA [26]. Nagappan, et al. reported that random pairing lead to conflict and undesirable pairs so it is preferable to avoid pairing students in this manner [16]. It was also reported that pairing computer science students with non-computer science students produced less compatible pairs [18].

Another important aspect related to effective pairing is ensuring that student pairs will be able to work together outside of the class room. One of our instructors mentioned that some of the laboratory assignments were too large to be completed during laboratory periods. In these cases, it is necessary to ensure that students will be able to meet outside of class to work on the assignment together. Bevan, et al. also described this issue, but indicated that students may not always report scheduling issues or conflicts with their partner [3].

Another important question to ask is how frequently pairs should be rearranged, if at all. Because some pairs will experience scheduling conflicts or other problems that make the pair ineffective, some amount of rearrangement will be necessary. Shrikanth, et al. indicated that while there were advantages of frequently exchanging pairs, a large number of students indicated that it took time to become adjusted to working with a new partner and that in some cases, mandatory rearrangement destroyed an existing, highly compatible pair [19].

4.3 Assignments and Grading

One of the major issues with pair programming that we experienced was related to assignments and grading. One of the CS1 instructors found that when pair programming was used in his course the student grades became much less diverse, making it more difficult to assign letter grades based on performance.

Figure 1 shows the distribution of student grades when pair programming was used and compares it against the student grades from eight semesters of historic data. A chi square test indicated that the distributions were significantly different ($p = .014$). Additionally, drop rates for the course were significantly lower ($p < 0.001$), and the student grades were spread over a much narrower range when pair programming was used.

This is likely related to issues discussed in 4.1 where it was reported that one student may be carrying the other. McDowell, et al. also discussed the possibility of grade inflation, i.e. that higher grades are a result of one member carrying the team [13]. Based on analysis of student performance at our university, it appears as though this was the cause of the change in grade distribution.

Chaparro, et al. also discussed the necessity of ensuring that programming assignments are adequately challenging [9]. Students in that study indicated that pair programming was not as beneficial if the assigned programming task was trivial or quickly completed. One of our instructors reported a similar issue for the last assignment of the semester, which was designed to be considerably easier than previous assignments. He noted few

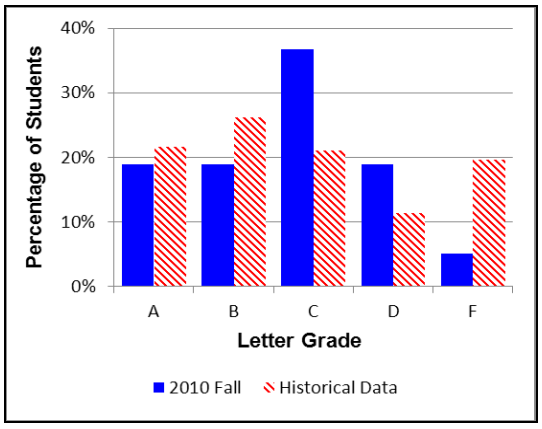

Figure 1. Comparison of Distribution of Student Grades Prior to and After Implementing Pair Programming

cases where one member of the pair would individually complete the exercise within the first day of it being assigned without consulting his or her partner.

4.4 Pair Interaction

Multiple researchers have indicated issues with how students interact. Williams, et al. found that students often did not properly use pair programming [25]. The most common identified issues from that research were that some pairs did not exchange roles at all while pair programming. Researchers at NCSU also indicated that students were unlikely to change roles unless instructed to during lab sessions. Chaparro, et al. did note that students should be allowed some leeway in role-switching and that forcing students to exchange roles at set intervals could interrupt students [9].

Our experiences were similar. In instances where the instructor took an actively role in telling pairs to switch, students were more likely to exchange roles, but if they were not instructed to do so, it was not uncommon for students to maintain whichever role that they had assumed for the duration of the session.

4.5 How Much Pair Programming

All three of the instructors with whom we worked have expressed some concern that using pair programming for every assignment may not be as effective as only using it for some assignments. One of the CS1 instructors felt that most new students did not have any programming experience and that until they gained some knowledge they wouldn't be able to effectively pair. One instructor noted instances where one member of a pair would actually be providing the other with incorrect information.

Another instructor felt that some assignments should be completed individually in order to provide the opportunity for students to show that they have mastered the ability to program without the need of a partner. The instructor stated that occasionally one partner would be absent from a lab and that the remaining student seemed to struggle, even though that student had done well on previous lab exercises.

Previous research has also indicated that some students feel that it is important to work individually [20].

5. SOLUTIONS AND RECOMENDATIONS

Based on our own experiences along with recommendations and results from previous research, we present a list of possible solutions for the problems and issues outlined in the previous section. It is believed that these suggestions are useful and valid as they are based on over a decade of empirical studies conducted on pair programming.

5.1 Individual Assessment

Our instructors felt that there were multiple ways to approach this issue. One instructor felt that having students complete lab assignments using a mixture of pair programming and individual work would provide an adequate amount of information to determine that students are able to program. When using a laboratory format where students alternated between using pair programming and working individually for each programming exercise, this instructor felt that there were not students who were able to take advantage of a strong partner in order to pass the class.

Another instructor indicated that having short in-class quizzes of programming concepts related to the previous lab would help to determine if each member of a pair was learning and benefitting equally. There were some concerns that this would create an excessive amount of work, but it was suggested that using a student response system could reduce the necessary amount of work on the part of the instructor and other researchers have reported that using these systems have other positive impacts in introductory programming courses [8].

In their initial studies, Williams, et al. implemented a policy where only students who scored above a certain threshold on exams were allowed to work in pairs [25]. This was done in order to prevent cases where an unskilled individual could get by with having a strong partner. This solution may not be ideal, especially given the recent findings that pair programming is most beneficial to students in the lowest quartile and improves their performance on programming exams [5]. This evidence contradicted most existing evidence regarding student performance on exams, although the authors were careful to distinguish that this was only observed on programming exams where students wrote code, and not the more traditional, written exams.

5.2 Subject Pairing

Subject pairing can be somewhat difficult. Williams, et al. indicated that some effective methods do exist [26, 27]. The most effective methods were pairing based on midterm score, pairing a Myers Briggs sensor with a Myers Briggs intuitor, or pairing students who share a similar work ethic.

Pairing based on midterm grade is impossible at the beginning of a CS1 course, but previous exam scores could be used to pair students taking a CS2 course. There is a general consensus among researchers that pairing students who are similar in ability, perceived or actual, is effective [5, 12]. Radermacher, et al. are investigating using mental model consistency tests to assess students and create pairs based on similarities in mental model consistency [20]. These tests which were developed by Bornat and Dehnadi can be given to students without previous programming experience and provide a reasonable assessment of ability [4, 10]. Other possible tests described by Simon, et al. could also be used to create ability-based pairs at the onset of the

semester [23]. However, the effectiveness of such pairing methods have not been well examined.

Considering that all of our instructors also felt that it was important for students to complete some assignments individually, another possibility is to have students complete several initial programming exercises individually before assigning them to pairs. This approach provides data which can be used to construct ability-based pairs.

5.3 Assignments and Grading

Previous studies have provided several different suggestions regarding assignments. Chaparro, et al. indicated that it is important to create programming exercises that were sufficiently challenging, otherwise students were less likely to view pair programming as beneficial [9].

Guidelines published by Beven, et al. indicated the importance of producing programming exercises as a function of class time [3]. The authors stipulate that programming exercises should be designed so that the vast majority of students will be able to complete the assignment during scheduled laboratory sessions.

Williams, et al. suggest balancing the grades so that individual work has a larger impact on the overall grade than pair work [28]. In the introductory programming class at NCSU, pair work only accounts for 10% of the total grade. Another policy for other courses where pair programming was used is that students must have a passing grade on the individual portions of the class in order to receive a passing grade [27]. This solution allows pair programming to benefit a student, but does not allow them to pass unless they've shown sufficient individual mastery of the course content.

5.4 Pair Interaction

Multiple researchers have indicated that a closed, mandatory laboratory session is necessary for pair programming to be effective [3, 27]. This has multiple benefits because it allows instructors to monitor students and alleviate the lack of role switching [25]. It can also reduce the frequency of scheduling conflicts [3]. Additionally, this affords instructors an opportunity to reinforce good pair programming behavior by reminding students to frequently switch roles.

In courses where a substantial amount of work must be completed outside of the course, Williams, et al. indicate the importance of using a peer review system [28]. This allows instructors to identify dysfunctional pairs where partners are either unable to find sufficient time to work together or where one partner is completing a majority of the work.

5.5 How Much Pair Programming

There is no doubt that pair programming is beneficial to student learning [5, 25] and enjoyment [14] and has benefits for instructors as well [19]. It has already been suggested that pair work should not constitute the majority of a student's grade, but has not been suggested how much pair activity should be conducted.

In introductory computer science courses where a majority of subjects have little or no experience with programming, our instructors feel that they should spend some initial time at the beginning of the course working individually in order to acquire some programming knowledge before working with a partner.

Such an arrangement has several benefits. First, it provides performance data that can be used to pair based on ability, as most researchers agree that this is the most desirable pairing strategy [3, 5, 26]. Secondly, it allows a period of time for students who do not intend to complete the course to drop before pairs are created, minimizing the need to re-pair students. Additionally, initial programming assignments are more likely be trivial or straightforward and may not benefit from the use of pair programming [9].

6. CONCLUSION

Pair programming is an area where there are still many unanswered questions, but there is a consensus among researchers that the use of pair programming is beneficial for multiple reasons. However, there are many important considerations to keep in mind when implementing pair programming for the first time.

Based on the experiences at our university along with study results and recommendations from previous researchers, it is suggested that when using pair programming for the first time, the following guidelines should be followed in order to ensure the experience is beneficial and enjoyable for students and instructors:

- Subjects should be paired such that they have similar levels of ability and availability. This can be accomplished by requiring students to complete some minimal amount of work individually to determine programming ability. The use of mental model consistency or other tests may also be used to establish a reasonable baseline [4, 23].
- Mandatory laboratory sessions are essential in order to ensure that students are appropriately using pair programming and that there is at least some time that both partners can easily meet.
- Programming exercises should be designed such that they do not require substantial amounts of out-of-class time in order to complete. It should be largely possible for students to complete programming exercise assignments during scheduled laboratory sessions. This is suggested in order to maximize the amount of time spent in a location where appropriate pair programming behavior can be reinforced by an instructor and to minimize the amount of time where students can engage in undesired behavior.
- Pair work should not constitute a majority of a student's grade, but students should still be graded on their ability to write code. Using programming exams allows instructors to grade more heavily on programming ability and also provides an opportunity for additional individual assessment.
- When using pair programming in introductory courses, pairs should be re-arranged. This exposes students to a larger number of classmates and helps them make acquaintances, whom they will likely work with in future classes. It also helps alleviate problems with ineffective pairs and ensures that no one student will be stuck with a bad partner.

These guidelines, while useful, should not be considered concrete. In some cases, there is not sufficient empirical data to provide strong support for one particular point of view. Additionally, every university faces different conditions and constraints.

7. ONGOING AND FUTURE WORK

There is no doubt that pair programming is beneficial to student learning [5, 25] and enjoyment [14] and has benefits for instructors as well [19]. One question which does not contain a significant amount of empirical support is to what extent pair programming should be used in introductory computer science courses. In the future we plan to investigate this by studying student learning outcomes when pair programming is only used partially. Two possible approaches are to only use PP during the second half of the semester, and to alternate between using pair programming and working individually

8. REFERENCES

[1] Beck, K. 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading, MA, USA.

[2] Berenson, S.B., Slaten, K. M., Williams, L., and Ho, C. 2004. Voices of women in a software engineering course: reflections on collaboration. *Journal on Educational Resources in Computing.* 4, 1, (March 2004), Article 3.

[3] Bevan, J., Werner, L., and McDowell, C. 2002. Guidelines for the Use of Pair Programming in a Freshman Programming Class. In *Proceedings of the 15th Conference on Software Engineering Education and Training* (CSEET '02). IEEE Computer Society, Washington, DC, USA, 100-108.

[4] Bornat, R., Dehnadi, S., and Simon. 2008. Mental models, consistency and programming aptitude. In *Proceedings of the tenth conference on Australasian computing education* (ACE '08), S. Hamilton and M. Hamilton (Eds.), Vol. 78. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 53-61.

[5] Braught, G., MacCormick, J., Wahls. T. 2010. The benefits of pairing by ability. In *Proceedings of the 41st ACM technical symposium on computer science education* (SIGCSE '10). ACM, New York, NY, USA, 249-253.

[6] 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 (February 2011), Article 2.

[7] Carver, J. C., Henderson, L., He, L., Hodges, J., and Reese, D. 2007. Increased Retention of Early Computer Science and Software Engineering Students Using Pair Programming. In *Proceedings of the 20th Conference on Software Engineering Education & Training* (CSEET '07). IEEE Computer Society, Washington, DC, USA, 115-122.

[8] Chamillard, A. T.. 2011. Using a student response system in CS1 and CS2. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 299-304.

[9] Chaparro,, E.A, Yuksel, A., Romero, P., and Bryant, S. 2005. Factors affecting the perceived effectiveness of pair programming in higher education. In *Psychology of Programming Interest Group 17th Workshop*, 5-18.

[10] Dehnadi., S. 2006. Testing programming aptitude. In *Proceedings of the Psychology of Programming Interest Group 18th Annual Workshop,* 22-37.

[11] Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S., and Gehringer, E. 2004. On understanding compatibility of student pair programmers. In *Proceedings of the 35th SIGCSE technical symposium on computer science education* (SIGCSE '04). ACM, New York, NY, USA, 7-11.

[12] Katira, N., Williams, L., and Osborne, J. 2005. Towards increasing the compatibility of student pair programmers. In *Proceedings of the 27th international conference on Software engineering* (ICSE '05). ACM, New York, NY, USA, 625-626.

[13] McDowell, C. Werner, L., Bullock, H., and Fernald, J. 2002. The effects of pair-programming on performance in an introductory programming course. *SIGCSE Bulletin*. 34, 1 (February 2002), 38-42.

[14] McDowell, C., Werner, L., Bullock, H., and Fernald, J. 2003. The impact of pair programming on student performance, perception, and persistence. In *Proceedings of the 25th international conference on software engineering* (ICSE '03), IEEE Computer Society, Washington, DC, USA, 602-607.

[15] McDowell, C., Werner, L., Bullock, H., and Fernald, J. 2006. Pair programming improves student retention, confidence, and programming quality. *Communications of the ACM*, 49, 8 (August 2006), 90-95.

[16] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S. 2003. Improving the CS1 experience with pair programming. *SIGCSE Bulletin.* 35, 1 (January 2003), 359-362.

[17] Nosek, J. The case for collaborative programming. 1998. *Communications of the ACM*, 41, 3 (March 1998), 105-108.

[18] Radermacher, A., and Walia, G. S. 2011. Investigating the effective implementation of pair programming: an empirical investigation. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 655-660.

[19] Radermacher, A., and Walia, G. S. 2011. Investigating student-instructor interactions when using pair programming: an empirical study. To be published in *24th Conference on Software Engineering Education & Training* (CSEET '11). IEEE Computer Society, Washington, DC, USA.

[20] Radermacher, A. Walia, G. S., Myronovych, O., Abufardeh, S., and Rummelt, R. 2010. *Investigating the use of pair programming at North Dakota State University: a family of empirical studies*. Technical report, the department of computer science, North Dakota State University, http://cs.ndsu.edu/research/reports.

[21] Sabin, R. E., and Sabin E. P. 1994. Collaborative learning in an introductory computer science course. In *Proceedings of the 25th SIGCSE symposium on computer science education* (SIGCSE '94). ACM, New York, NY, USA, 304-308.

[22] Srikanth, H., Williams, L., Wiebe, E., Miller, C., and Balik, S. 2004. On Pair Rotation in the Computer Science Course. In *Proceedings of the 17th Conference on Software Engineering Education and Training* (CSEET '04). IEEE Computer Society, Washington, DC, USA, 144-149.

[23] Simon, Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., de Raadt, M., Haden, P., Hamer, J., Hamilton, M., Lister, R., Petre, M., Sutton, K., Tolhurst, D., and Tutty, J. 2006. Predictors of success in a first programming course. In *Proceedings of the 8th Austalian conference on Computing education - Volume 52* (ACE '06), Denise Tolhurst and Samuel Mann (Eds.), Vol. 52. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 189-196.

[24] Williams, L., Kessler, R. R., Cunningham, W., and Jeffries, R. 2000. Strengthening the case for pair programming. *IEEE Software.* 17, 4 (July 2000), 19-25.

[25] Williams, L., Wiebe, E., Yang, K., Ferzli, M., and Miller, C. 2002. In support of pair programming in the introductory computer science course. In *Computer Science Education*, 12, 3 (September 2002), 197-212.

[26] Williams, L., Layman, L., Osborne, J., and Katira, N. 2006. Examining the Compatibility of Student Pair Programmers. In *Proceedings of the conference on AGILE 2006* (AGILE '06). IEEE Computer Society, Washington, DC, USA, 411-420.

[27] Williams, L. 2007. Lessons learned from seven years of pair programming at North Carolina State University. *SIGCSE Bull.* 39, 4 (December 2007), 79-83.

[28] Williams, L., McCrickard, D. S., Layman, L., and Hussein, K. 2008. Eleven Guidelines for Implementing Pair Programming in the Classroom. In *Proceedings of the Agile 2008* (AGILE '08). IEEE Computer Society, Washington, DC, USA, 445-452.