

# Investigating the Effective Implementation of Pair Programming: An Empirical Investigation

Alex Radermacher  
North Dakota State University  
IACC Bldg., Room 258  
Fargo, ND 58108  
701-231-8562

alex.radermacher@ndsu.edu

Gursimran S. Walia  
North Dakota State University  
IACC Bldg., Room 258-A14  
Fargo, ND 58108  
701-231-8185

gursimran.walia@ndsu.edu

## ABSTRACT

Pair programming is a programming technique where two programmers work together on the same programming task. Previous research has shown that it is effective for improving the learning effectiveness, efficiency, and enjoyment of students in introductory programming courses. Much research has also been dedicated to determining effective strategies for forming pairs. This paper discusses two different empirical studies conducted at North Dakota State University to a) test the feasibility of using pair programming in introductory computer science courses and b) determine whether or not major-based pairing produces effective pairs. The results of these studies provide support for implementing pair programming in introductory computer science courses and show that pairing of computer science and non-computer science students may produce pairs which are less compatible than other pairing methods.

## Categories and Subject Descriptors

K.3.2 [Computer and Information Science Education]: Computer Science Education.

## General Terms

Measurement, Performance, Experimentation, Human Factors.

## Keywords

Pair Programming.

## 1. INTRODUCTION

Pair programming has been shown to have multiple positive benefits. Previous research has empirically demonstrated that using pair programming in introductory computer science courses leads to an improvement in the quality of student work, an increase in student retention in computer science programs, and an increase in the student's level of enjoyment of learning [4, 7, 8, 9]. At North Dakota State University (NDSU), it was noticed that the student retention among first year students was low and that the instructors of these first-year courses were experiencing low

class attendance in addition to several students dropping those courses. To investigate the use of pair programming as a potential solution for these issues, multiple empirical studies (each with a different research goal) was planned and conducted in introductory computer science courses at NDSU near the end of the spring 2010 semester. This paper presents findings from two of those studies.

The first study evaluated the feasibility of implementing pair programming in introductory computer science courses at NDSU. Prior research indicates that pair programming is effective when used in introductory computer science courses, so this study served as a study replicated to verify the results from previous studies [6, 9, 10, 12]. The results from this study provided increased support that the pair programming is more effective in improving student's learning of programming concepts as compared to individual learning.

Researchers have also investigated the effect of the factors (e.g., student's skill levels) on the pair performance of students. In order to understand how pair programming can be most effectively implemented in an introductory computer science course, the second study explored a major-based pairing strategy. In this study, the subjects were arranged into pairs based on declared major to determine if any particular pairing arrangements were more effective than the others. It was thought that it would be most effective in terms of student learning and enjoyment to pair students such that one of those students had a computer science major and the other student had some other major. The results from this study indicate that the pairing a computer science student with a non-computer science student may produce a less compatible pair than other pairing methods.

The remainder of this paper is organized as follows: Section 2 provides a general background of pair programming and discusses previous empirical studies of pair programming related to our research. Section 3 details the design of two experiments conducted as part of our research of pair programming. Section 4 presents an analysis of the results of the two experiments. Section 5 lists threats to validity that were identified. Section 6 gives a summary and short discussion of the results. Section 7 concludes the paper and discusses future work that is being conducted as part of ongoing research into pair programming at NDSU.

## 2. BACKGROUND AND RELATED WORK

Pair programming as a term was first coined by Kent Beck and his colleagues in *Extreme Programming Explained: Embrace Change* [1], but the concept of collaborative programming had been

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCSE'11, March 9-12, 2011, Dallas, Texas, USA.

Copyright 2011 ACM 978-1-4503-0500-6/11/03...\$10.00.

studied previously. Pair programming was introduced as a part of Beck's Extreme Programming (XP) as a programming technique where two individuals collaborate on a programming task by taking on the roles of a driver (i.e. the person who controls the keyboard and types in the program) and a navigator (i.e. the person who writes the code) and by frequently exchanging these roles. After the publication of this book, a large number of researchers began to actively investigate pair programming through controlled experiments to gain empirical evidence of the effectiveness of pair programming for student learning.

An early empirical study of pair programming conducted at Temple University studied fifteen full-time systems programmers employed at a program trading firm [9]. Results from this study indicated that programmers who worked as a pair were able to produce higher quality code, both in terms of readability and functionality. Additionally, the programmers that used pair programming reported a greater amount of confidence in their work and a higher level of enjoyment than programmers who worked individually. A study conducted at the University of Utah that used students in the university's Senior Software Engineering course found similar results [12]. Researchers additionally found that pair programmers were able to produce solutions significantly faster than individuals and found that in the best case, the pair programming teams only required 15% more programmer hours than an individual required to complete a programming exercise.

Studies have also been conducted to find effective methods for pairing and examining other factors which may impact the effectiveness of pair programming. Chaparoo et al., performed a study on student perceptions of pair programming where they found that the difficulty of the programming task and the skill level of individual students played a major role in the effectiveness of pair programming [5]. They found that the students preferred to work with a partner with a similar skill level so that they can contribute equally to the pair. Students also felt that the pair programming was less useful if the programming exercises were trivial or could easily be completed by a single person Williams et al., performed a comprehensive study of the effects of student ability, learning style, personality type, and other factors on pair programming [13]. The results showed that the perceived technical competence (as opposed to the actual skill level) plays a role in determining pair compatibility, and that the personality type did not play a large role in pair compatibility.

Researchers have also shown that pair programming can help improve student retention in computer science. McDowell et al., reported that students who used pair programming in an introductory computer courses were more likely to remain enrolled three quarters after the study was conducted and were more likely to have a declared computer science major than students who had worked individually [7, 8]. A similar study at Mississippi State University found that the use of pair programming in an introductory computer science course lead to a higher rate of retention among computer science, software engineering, and computer engineering students after one year [4].

### 3. EXPERIMENT DESIGN

The major objectives of this research were to:

- Evaluate the impact of pair programming on student learning effectiveness, efficiency, and enjoyment of programming compared to individual learning.
- Investigate the effects of major-based pairing strategies on student learning effectiveness and enjoyment of using pair programming.

Study 1 examined the effects of pair programming on students' learning and performance in a highly controlled setting and serves as a baseline with which to compare the results from the previous studies. One group of subjects worked with pairs and used pair-programming to complete a programming assignment whereas the other group of subjects worked individually to complete the same assignment.

Study 2 examined effects on pair performance when students are paired based on their declared majors. Three different pairing archetypes were created and evaluated for their effect on the pair programming. These archetypes are explained in section 4.3.2. All subjects used pair programming to complete the same assignment.

#### 3.1 Study Goals and Hypotheses

Using the Goal Question Metric (GQM) approach, the following five goals and hypotheses were created.

**Goal 1:** Analyze pair based learning and individual learning for the purpose of their evaluation with respect to the students' preferences of learning method.

*Hypothesis 1:* Students prefer working in pairs rather than working individually.

**Goal 2:** Analyze pair based learning and individual learning for the purpose of their evaluation with respect to students' perception of the effectiveness of learning methods.

*Hypothesis 2:* Students perceive working in pairs as being a more effective learning method when compared with working individually.

**Goal 3:** Analyze pair based learning and individual learning for the purpose of their evaluation with respect to the effectiveness of students' actual learning of programming concepts.

*Hypothesis 3:* Students working in pairs are more effective (when examining the correctness of their work.) than the students who work individually.

**Goal 4:** Analyze major-based pairing methods for the purpose of understanding the effects on pair performance with respect to students' effectiveness and enjoyment.

*Hypothesis 4:* Pairing a computer science student and a non-computer science student will be more effective in improving their learning and their level of enjoyment than other major-based pair types when using the pair programming.

**Goal 5:** Analyze pair programming pairs and individuals for the purpose of evaluating their efficiency with respect to the time taken to complete a given programming task.

*Hypothesis 5:* Students working in pairs are more efficient (able to complete a programming task more quickly) than students working individually.

#### 3.2 Independent and Dependent Variables

The experiments manipulated following independent variables:

- *Time Limit*: the amount of time given to complete the programming exercise.
- *Subject's major*: the declared major of a subject.

We also measured the following dependent variables:

- *Effectiveness*: measures the correctness of a subject's implementation of a programming exercise.
- *Efficiency*: measures the amount of time taken by a subject to complete a programming exercise.
- *Enjoyment*: measures the student's preference for using pair programming rather than working individually.

### 3.3 Participating Subjects

*Study 1*: The subjects were thirty-five undergraduate students enrolled in the CS1 course at NDSU during the spring 2010 semester. The subjects were randomly divided into either an experimental group, which included eighteen students, or a control group, which included seventeen subjects. Eight subjects dropped out of study, chose not to participate, or did not attend class on the day of the experiment run, leaving sixteen students in the experiment group and eleven students in the control group.

*Study 2*: The subjects were sixty-four undergraduate students enrolled in two sections of the CS2 course at NDSU during the spring 2010 semester. Twenty-five subjects chose not to participate in the study, did not attend class on the initial day of the study, or were removed from the analysis because they did not complete the assignment as a pair. Table 1 provides a list of the remaining thirty-nine participating subjects' majors.

**Table 1: List of Subject Majors**

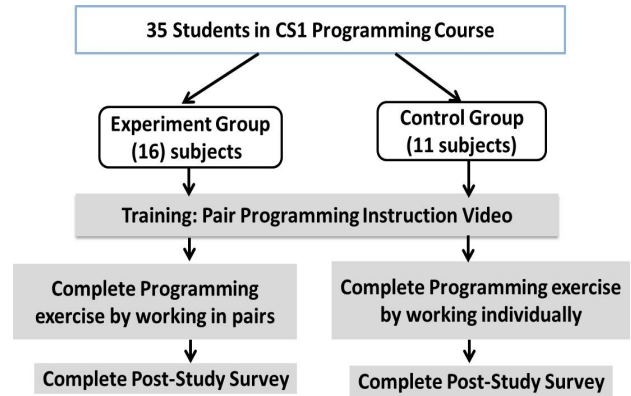
Declared Major	Number of Subjects
Computer Science	20
Computer Engineering	6
Electrical Engineering	4
Undecided/Non-Degree	4
Mechanical Engineering	1
Mathematics	1
English	1
Biotechnology	1
Statistics	1

### 3.4 Experiment Procedure

*Study 1*: This study used a control group experiment design with students in the CS1 course randomly assigned into either the experimental group (in which they used pair programming), or a control group (in which they worked individually). Experimental group pairs were formed by selecting subjects with similar grades, as previous research has suggested that this is an effective method of pairing students [2, 3, 4].

On the day prior to beginning the programming exercise, subjects were shown a ten minute video in class that described how to use pair programming effectively on their programming assignment [11]. The next day during their lab session, subjects in both groups worked on a programming exercise that was specifically designed by the instructor such that it should not take more than one lab period (fifty minutes) to complete.

Finally, the subjects were asked to complete a short post-study survey that was used to provide feedback about their pair programming experience, their perception of their learning of programming concepts, and their preferences for using pair programming. Two different sets of surveys were used for the subjects in the experiment and control group. The experiment steps and training lectures are illustrated in Figure 1.

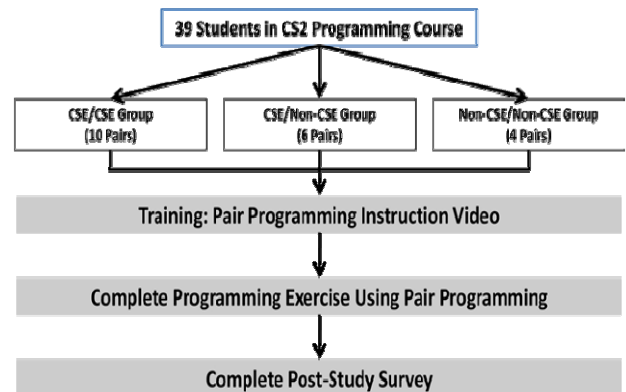


**Figure 1: Experiment Design for Study 1**

*Study 2*: This study used a repeated measures design (without a control group) in the CS2 course. Thirty-nine participating subjects were randomly assigned into one of three groups:

- *CSE/CSE group*: pairs were comprised of two subjects, each with a declared major in computer science or computer engineering.
- *CSE/non-CSE group*: pairs were comprised of one subject with a declared major in computer science or computer engineering, and one subject with a declared major other than computer science or computer engineering (e.g. Math, English, Psychology, etc.).
- *Non-CSE/Non-CSE group*: pairs were comprised of subjects, each with a declared major other than computer science or computer engineering.

Similar to the Study 1, pairing within all groups was done by pairing subjects with the closest grades together. The subjects in this study were shown the same instructional video from Study 1.



**Figure 2: Experiment Design for Study 2**

Subjects were given a programming exercise related to the topics that were being covered in class that week and were given two lab periods (with each lasting fifty minutes) and three days of out-of-class time to complete the programming exercise. Subjects were only monitored while working on the exercise during the designated lab periods. Subjects worked with their assigned partner to complete the programming exercise.

After completing the exercise, subjects were asked to complete a short post-study survey. The survey used in this study was the same eleven question survey that was used by subjects in the experimental group in Study 1. The experiment steps, and training lectures in this study are shown in Figure 2.

## 4. ANALYSIS AND RESULTS

This section provides an analysis of qualitative data and quantitative data collected during the Study 1 and Study 2. The data analyzed includes the subject's responses to post-study survey questions, the grade received on the programming exercise, and the time taken to complete the programming exercise. Survey questions were based on a five-point Likert scale ranging from a strong agreement response to a strong disagreement response. A full list of the survey questions is available in the technical report for these studies [10]. This section is organized around the hypotheses presented in Section 3.1. An alpha value of 0.05 was used for all statistical tests and an  $r^2$  value of 0.30 was used for correlations.

### 4.1 Hypothesis 1: Analysis of Student's Preference of Learning Method

Subjects rated their level of preference of using pair programming on the following 5-point scale: 1- strongly prefer to work in pairs, 2- prefer to work in pairs, 3- no preference, 4- prefer to work individually, 5- strongly prefer to work individually.

As shown in Table 2, the results from Study 1 showed that the experimental group expressed a strong opinion for using pair programming, but the control group did not show any preference for using pair programming or working individually. The results from Study 2 showed that the subjects belonging to the CSE/CSE group and non-CSE/non-CSE group expressed a strong preference for using pair programming rather than working individually.

**Table 2: Preference for using Pair Programming**

Group	Study 1		Study 2		
	Exp.	Control	CSE / CSE	CSE / non-CSE	non-CSE / non-CSE
p-value	0.002	0.572	0.001	0.125	0.016

Table 2 also shows the p-values for a non-parametric binomial test used to determine whether the mean response of subjects in studies 1 and 2 was significantly less than the midpoint of the scale. Significantly positive characteristics are shaded in Table 2.

Based on these results, hypothesis 1 can be accepted as the majority of students, who used pair programming, show a significantly strong preference for using pair programming.

### 4.2 Hypothesis 2: Analysis of Students' Perception of Learning Improvements

Subjects who used the pair programming in both studies rated their perception of how pair programming affected their learning by answering three survey questions. The three questions asked: 1) Whether or not students felt pair programming impacted their understanding of the concepts present in the programming exercise, 2) Whether or not they felt they learned more through using pair programming, and 3) Whether or not they believed that pair programming would have improved their understanding of the entire course if it had been used for the entire semester. A similar five-point scale was used for all questions ranging from strong disagreement to strong agreement. For analyzing the subject's response to each of the three questions, a non-parametric binomial test was used to determine whether the mean response of subjects is significantly less than the midpoint of the scale.

Analysis of the subject's response to the first question (as shown in Table 3) shows that, in all three groups from Study 2, subjects strongly agreed that pair programming improved their understanding of the programming concepts present in the programming exercise. Subjects from the experimental group in Study 1 also strongly agreed with that assessment. Table 3 also shows the p-values from a non-parametric binomial test. Significantly positive characteristics are shaded.

**Table 3: Impact of Pair Programming on Concept Learning**

Group	Study 1	Study 2		
	Experimental	CSE / CSE	CSE / non-CSE	non-CSE / non-CSE
p-value	0.000	0.000	0.003	0.002

Table 4 shows the p-values for a non-parametric binomial test performed on responses to the second question. Two of the three groups in study 2 strongly agreed that they learned more through using pair programming on this programming exercise than they would have if they had worked individually on it. The exception in this case was the CSE / non-CSE group from Study 2. Also, subjects in the both the experimental and control groups of Study 1 felt as though pair programming would have improved their learning of the programming concepts present in the programming exercise.

**Table 4: Learning Individually vs. Pair Programming**

Group	Study 1		Study 2		
	Exp.	Control	CSE / CSE	CSE / non-CSE	non-CSE / non-CSE
p-value	0.000	0.003	0.001	0.056	0.002

Analysis of the subject's response to the third question shows that, two out of three groups in Study 2 strongly agreed that pair programming would have improved their learning of the course content if pair programming had been used for the entire semester.

**Table 5: Learning with a Full Semester of Pair Programming**

Group	Study 1		Study 2		
	Exp.	Control	CSE / CSE	CSE / non-CSE	non-CSE / non-CSE
p-value	0.000	0.044	0.102	0.009	0.016

(as shown in Table 5). Subjects in the CSE / CSE group from Study 2 were the exception in this case. Both experiment and control group subjects also strongly agreed with this assessment.

Based on these above results, hypothesis 2 can be accepted. A majority of students perceive pair programming as being a more effective learning method than working individually.

### 4.3 Hypothesis 3: Analysis of Students' Actual Learning Improvements

To measure student effectiveness, their grades from previous laboratory assignments were compared against the grade received on the programming exercise used as part of the study. None of the groups from either study showed a statistically significant increase in their grades.

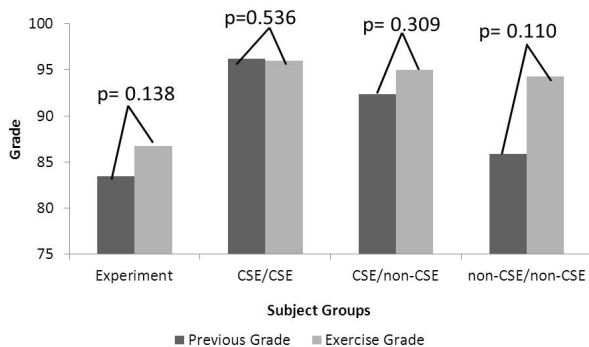


Figure 3: Comparison of Previous and Exercise Grades

Figure 3 shows the mean grade for previous course programming assignments, mean grade for the programming exercise completed as part of the study, and p-value from a paired t-test used to determine whether there was a significant difference between the grades. A comparison was also made between the exercise grades of the experimental and control groups subjects in Study 1. The average grade for the experimental group subjects was 86.80 and the average grade for the control group subjects was 76.40. A two-sample t-test was used to compare the results and yielded a p-value of 0.074, which was not statistically significant

In both experiments, many of the subjects who participated had previous course grades above 90%, which leaves marginal room for improvement. When examining only subjects who had previous course grades below 90%, there was a statistically significant increase in the grades of subjects from Study 2 as shown in Table 6.

Table 6: Comparison of Grades for Subjects Below 90%

	Study 1	Study 2
Previous Grade	77.20	75.14
Exercise Grade	81.00	93.89
p-value	0.224	0.008

In Study 1 there was a strong inverse correlation ( $r^2 = .468$ ) between the amount of time taken to complete the programming exercise and the previous course grade of the subjects. This suggests that those subjects who were not doing as well in the course may not have had sufficient time to complete the programming exercise within the allotted fifty-minute time limit.

This may be a possible explanation of why subjects in Study 1 with previous course grades below 90% did not perform significantly better on the pair programming exercise.

Based on these results, hypothesis 3 can be accepted. However, it may be difficult to measure improvements among students who are already doing well in the course. It is also possible that pair programming is most effective for students who would otherwise struggle with the course. Future work will investigate this issue.

### 4.4 Hypothesis 4: Analysis of Pair Effectiveness on Students' Learning

As seen from Tables 2, 4, and 5, there are some differences in students' perceptions of pair programming when different forms of major-based pairing are used. Subjects in the CSE / non-CSE group did not have a strong perception that pair programming can improve their learning or a strong preference for using pair programming in the future.

A non-parametric binomial test was used on survey question that asked subjects how well they got along with their partner. The mean response for all groups was lower than the midpoint of the scale indicating that subjects agreed that they got along with their partner. This suggests that general conflict between group members was not a cause. One other potential explanation may be that members of the CSE / non-CSE pairs reported a greater perceived skill disparity between each other than other groups. This was interesting as there was little disparity between the actual grades of each pair member. Previous research has shown that perceived similarities in skill are actually more important than actual similarity in skill for pair compatibility [13]. This may explain the differences in results between the different groups, but does not the cause of those differences.

Based on these results, hypothesis 4 can be rejected. There are differences that arise as the result of using different major-based pairing strategies. However, it is not evident that major-based pairing is the cause of these differences. Also, the initial assumption that pairing computer science students with non-computer science students may produce pairs which are less compatible than pairs created using other methods.

### 4.5 Hypothesis 5: Evaluating the Efficiency of Students using Pair-Programming

In Study 1 the amount of time taken to complete the programming exercise was measured for subjects in both groups. Subjects in the experimental group completed that exercise in a mean time of 39.13 minutes and subjects in the control group completed it in a mean time of 43.54 minutes. A two-sample t-test used to compare the times from each group showed that the difference in completion times was statistically significant (p-value = 0.016).

Based on these results, hypothesis 5 can be accepted. Students who use pair programming are able to complete a programming exercise in less time than students who work individually.

## 5. THREATS TO VALIDITY

*Study 1:* As mentioned in the analysis of hypothesis 3, subjects may have scored lower on the programming exercise used during the study due to the strict, fifty-minute time limit. It may have also had an impact on the students' perceptions of pair

programming as well. The training video, which was shown to students prior to completing the programming exercise, included multiple statements and testimonials about how pair programming was more fun and would provide a better education experience than working individually. This may have biased the students' perception of the benefits of using pair programming.

*Study 2:* was conducted shortly after the drop date for the course. Many students who were struggling with the course had either dropped the course or had stopped coming to class. This may have led to the inflation in student grades as seen in the analysis of hypothesis 3. This also means that it may not be possible to generalize the results of this study. Some pairs did not complete the assignment together. These results were removed from analysis and it is believed that all such cases were correctly identified. However it is possible that one instance was missed and may have affected the results.

## 6. SUMMARY OF RESULTS

Results from both studies show that students perceive pair programming as being beneficial and all of the subjects who used pair programming indicated that they would prefer using it again in the future as opposed to working individually. It also shows that pair programming is beneficial to student performance among students who may be struggling with the course. Additionally, the results indicate that pair programming allows a student pair to complete a given amount of work more quickly than individuals.

Major based pairing has also been shown to have an impact on student perceptions of pair programming. However, it is not clear exactly what causes these differences. Previous research has shown that personality does not have a major impact on pair compatibility so it is unlikely that this was the cause [13]. It is also unknown what caused a large perceived difference in skill between subjects in the CSE / non-CSE group, even when grades from previous course exercises indicated that there was little difference in actual skill.

## 7. CONCLUSION AND FUTURE WORK

Because it is likely that there are remaining unknown factors which contribute to the differences identified between pairs using different major-based pairing strategies, further research is necessary to determine what causes these differences and if these factors can be manipulated to produce more effective pairs.

It is suspected that while a student's declared major is not the cause of these results, it may be correlated to other factors which do have a causal effect on the student effectiveness and enjoyment when using pair programming. Ongoing research at NDSU is being conducted to evaluate other factors that may determine pair effectiveness. The current studies at NDSU are being run over the course of an entire semester and the impact of pair programming on student retention will be monitored.

## 8. ACKNOWLEDGMENTS

Our thanks to Sameer Abufardeh and Oksana Myronovych for their assistance in conducting these studies.

## 9. REFERENCES

- [1] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, Massachusetts, 2000.
- [2] Bevan, J., Werner, L., and McDowell, C. Guidelines for the use of pair programming in a freshman programming class. In *Proceedings of the 15th Conference on Software Engineering Education and Training*, Washington, DC, USA, 2002. IEEE Computer Society.
- [3] Braught, G., MacCormick, J., and Wahls, T. The benefits of pairing by ability. In *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education*, pages 249-253, New York, NY, USA, 2010.
- [4] Carver, J., Henderson, L., He, L., Hodges, J., and Reese, D. Increased retention of early computer science and software engineering students using pair programming. In *Proceedings of the 20th Conference on Software Engineering Education & Training*, p. 115-122, Washington, DC, USA, 2007. IEEE Computer Society.
- [5] Chaparro, E., Yuksel, A., Romero, P., and Bryant, S. Factors affecting the perceived effectiveness of pair programming in higher education. In *Psychology of Programming Interest Group 17th Workshop*, pages 5-18, 2005.
- [6] McDowell, C., Werner, L., Bullock, H., and Fernald, J. The effects of pair-programming on performance in an introductory programming course. *SIGCSE Bull.*, 34(1):38-42, 2002.
- [7] McDowell, C., Werner, L., Bullock, H., and Fernald, J. The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering*, pages 602-607, Washington, DC, USA, 2003. IEEE Computer Society.
- [8] McDowell, C., Werner, L., Bullock, H., and Fernald, J. Pair programming improves student retention, confidence, and program quality. *Commun. ACM*, 49(8):90-95, 2006.
- [9] Nosek, J. The case for collaborative programming. *Commun. ACM*, 41(3):105-108, 1998.
- [10] Radermacher, A., Walia, G., Myronovych, O., Abufardeh, S., and Rummelt, R. "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, 2010, <http://cs.ndsu.edu/research/reports>.
- [11] Williams, L. Introduction to pair programming, version 2. [http://www.youtube.com/watch?v=rG\\_U12uqRhE](http://www.youtube.com/watch?v=rG_U12uqRhE)
- [12] Williams, L., Kessler, R., Cunningham, W., and Jeffries, R. Strengthening the case for pair programming. *IEEE Softw.*, 17(4):19-25, 2000.
- [13] Williams, L., Layman, L., Osborne, J., and Katira, N. Examining the compatibility of student pair programmers. In *Proceedings of the conference on AGILE 2006*, pages 411-420, Washington, DC, USA, 2006. IEEE Computer Society.