

# Investigating Student-Instructor Interactions When Using Pair Programming: An Empirical Study

Alex Radermacher  
North Dakota State University  
alex.radermacher@ndsu.edu

Gursimran Walia  
North Dakota State University  
gursimran.walia@ndsu.edu

## **Abstract**

*At North Dakota State University, there are multiple sections of the CS1 and CS2 introductory computer science courses. A large number of students are enrolled in each section, making it difficult to hold laboratory sessions as there is not enough space for all of the students in one room. This results in diminished student attendance and a decrease in student understanding of the course material. Pair programming has been shown to have multiple benefits in educational use. Previous research has shown that it benefits student learning in addition to increasing the student retention in computer science programming courses. Using pair programming would also allow students to share laboratory resources and make it possible to accommodate more students in laboratory session. To study the effects of pair programming on student-instructor interactions in laboratory sessions of introductory computer science courses, an empirical study was conducted at North Dakota State University. Data about student-instructor interactions was collected by monitoring the laboratory sessions during the study run, as well as through a post-study survey given to students and interviews with the instructors. The results from this study indicate that having students work in pairs as opposed to individually reduces the number of questions from students and decreases the amount of time that a student must wait for instructor assistance.*

## **1. Introduction**

At North Dakota State University (NDSU), there are multiple sections of introductory computer science courses every semester. At the beginning of the semester, these sections have enrollments of over forty students, many of whom have never taken a computer programming course before. Additionally, there is limited space in the rooms where the laboratory sessions of these courses are held, which forces instructors to divide the students between two classrooms. This makes it difficult for instructors to identify the students in need of assistance and it is difficult for students to receive help from an instructor. Based on comments from instructors of these classes, it is believed that this creates an environment where numerous students are constantly waiting for assistance from an instructor that leads them to stop attending laboratory sessions or drop the course out of frustration.

Previous research into pair programming at NDSU and at other universities has been shown to improve the quality of student work, and an increase in the student's level of enjoyment of learning of programming among other effects [3], [4], [7], [10]. These reasons alone suggest that pair programming would help alleviate frustration levels for the students, but it can also help address some of the problems faced by instructors while assisting students in the laboratory sessions. Pair programming would allow for two students to utilize one computer, permitting a larger number of students to be present in an individual room for lab sessions. Additionally, observational results from previous research indicate that this would reduce the number of questions and allow instructors the ability to address student questions more quickly [13].

This paper presents the results from the observations of the interactions between students and instructors in introductory computer science course lab sessions at NDSU as well as results from a student survey and comments from the course instructors. The results from this study indicate that using pair programming results in fewer questions per student and that student's spend a shorter amount of time waiting for assistance from an instructor.

The remainder of this paper is organized as follows: Background and related work on pair programming is discussed in section 2. Section 3 details the study design. Data analysis and study results are presented in section 4. Section 5 discusses threats to validity. A discussion of the study results is found in section 6. Finally, section 7 presents a conclusion.

## 2. Background and Related Work

The term pair programming was originally introduced by Beck, et al. in *Extreme Programming Explained: Embrace Change* [2]. It is described as a programming technique in which two programmers work collaboratively by alternating between the roles of the driver (i.e., the programmer who is in control of the keyboard and enters the program into the computer) and the navigator (i.e., the programmer who helps shape the design of code and catch mistakes made by the driver) as they develop a program.

Pair programming has been extensively studied in academia and has been shown to have numerous benefits for student learning. An early study of pair programming conducted at the University of Utah found that students who used pair programming produced solutions of higher quality than the students who worked individually [10]. Researchers also indicated that pair teams were able to produce solutions more quickly than individual programmers and that in the best case, pair programming teams only required 15% more programmer hours to produce a solution than individual programmers. Researchers at the University of California Santa Cruz found similar results when examining solution quality and performed statistical analysis to show that the results were due to the stronger programmer carrying the pair [4].

Another widely investigated benefit of pair programming has been in the area of student retention in computer science. A study conducted at Mississippi State University determined that using pair programming in the introductory computer science course leads to an increase in the number of students who retained their major one year after the class [3]. Similar results for student retention were found at a series of studies conducted at USCS [5], [6]. The results indicated that pair programming is useful for retaining women in computer as female students who used pair programming reported being more confident in their work. A case study at North Carolina State University also found that pair programming helped to boost the confidence levels of female programmers and increase their interest in IT careers [8].

The main motivation for this research however, comes from a study conducted by Williams, et al. at NCSU [13]. Although researchers in this study did not empirically measure interactions between students and instructors, they did release qualitative findings regarding such interactions. They noticed that when using pair programming students appeared to have fewer questions and that the questions which students did have tended to be more advanced. They also noted that when working individually, students who ran into problems spent more time waiting for instructor assistance. In order to better understand the extent of these effects and to ensure that the differences between individual and pair work were significant, it was necessary to conduct an empirical study.

## 3. Study Design

The high-level goal of this study was to evaluate the impact of pair programming on student-instructor interactions in a computer science laboratory setting. This study follows a

classic, repeated-measures approach where student-instructor interactions were monitored and recorded using a timer software tool in multiple computer science laboratory sessions where students worked both individually and in pairs to complete their laboratory assignments.

### 3.1. Study Goals and Hypotheses

A series of hypotheses were created using the Goal Question Metric (GQM) approach [1]. These hypotheses are listed in Table 1.

**Table 1: Study Hypotheses**

Number	Hypothesis
<b>H1</b>	The use of pair programming results in fewer questions asked per student.
<b>H2</b>	The use of pair programming results in less time spent by students waiting for an instructor's assistance.
<b>H3</b>	The use of pair programming results in less time spent by instructors in addressing student questions.

Hypothesis H1 postulates that the use of pair programming will result in fewer questions per student being asked in laboratory sessions. The next two hypotheses (H2 and H3) postulate that the use of pair programming will result in both a reduction in the amount time a student must spend waiting for instructor assistance and the amount of time spent interacting with an instructor in order for a question to be answered.

The first hypothesis was chosen because previous research has indicated that when using pair programming, students are able combine their knowledge and strengths by working collaboratively [2], [10]. An obvious implication of this result is that students will be less reliant on instructors and therefore need less assistance from an instructor. The second hypothesis follows from the assumption that if there are fewer questions asked by students, it is logical to assume that the amount of time a student must wait will also decrease. The third hypothesis assumes that an instructor will be able to spend shorter periods of time interacting with pairs as opposed to individual students. When an instructor interacts with a pair team, the instructor would only need to provide assistance until one member of group understands the information and is able to move forward independently of the instructor.

### 3.2. Study Variables

As part of the GQM approach, a set of metrics devised for the study served as independent and dependent variables for the study. Brief descriptions of these variables are listed below.

The experiment manipulated following independent variables:

- *Programming Technique*: Whether subjects completed the laboratory assignment working individually or working in pairs.
- *Course*: Whether subjects were students enrolled in the CS1 course or CS2 course.
- *Pairing Technique*: How the subjects were arranged into pairs.

We also measured the following dependent variables:

- *Class Size*: Measures the number of subjects present in the lab session as recorded ten minutes after the start of the course period.
- *Wait time*: Measures the amount of time a subject spent waiting for instructor assistance, and measured as the duration of time from when a subject first raised his or her hand until the instructor was able to respond to that subject.
- *Interaction time*: Measures the amount of time a subject spent interacting with an instructor, and measured as the duration of time from when the instructor began

addressing a subject's question until the instructor had finished addressing that question

- *Total time*: Measures the total amount of time a subject spent waiting for a question to be answered, measured as the sum of the wait time and the interaction time.

Initially, there was an intention to collect additional measures, such as the number of abandoned questions (a question which a subject ceased to have before it could be addressed by an instructor) and the amount of time before a question was abandoned. Due to difficulties in accurately measuring these metrics, it was decided to remove them from the study. This is discussed in detail along with other issues in acquiring precise measurements in Section 5.

### 3.3. Study Subjects

The participating subjects in this study were 44 students enrolled in one section of the CS1 course during the fall 2010 semester. Additionally students enrolled in two sections (one section contained 17 students; the other section contained 36 students) of the CS2 course during the fall 2010 semester were monitored to help understand if the results from the study could be generalized to other computer science laboratory sessions.

### 3.4. Study Procedure

The study procedure is shown in Figure 1 and the study steps are described as follows:

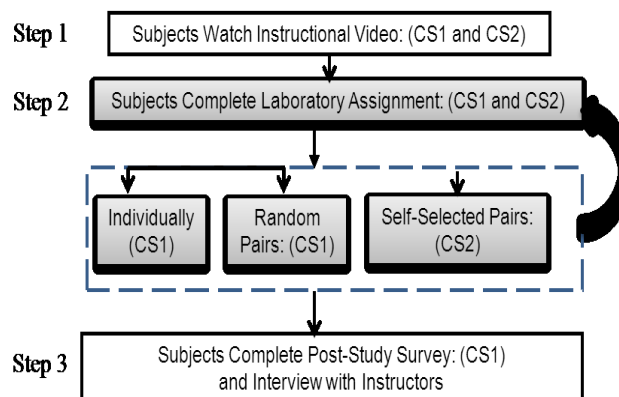
**Step 1:** Prior to beginning the first pair programming exercise, subjects from both courses were shown an instructional video about pair programming [12]. This video provided a brief description of pair programming and a list of do's and don'ts for effective pair programming.

**Step 2:** Next, subjects were monitored in laboratory sessions in which they worked on programming assignments either in pairs or individually. Subjects in the CS1 course were paired randomly for each programming assignment and were notified of their partner the day before the laboratory session. Subjects in the CS2 course were allowed to choose their partner. Based on observations from the researcher monitoring the laboratory sessions, subjects in the CS2 course tended to keep the same partner.

Subjects in the CS1 course were split across two adjacent rooms which the instructor would move between because there were too many subjects to fit everyone into the same room in many of the observed laboratory sessions. The researcher monitoring the labs used two separate instances of the timer software tool (discussed in more detail in the next section) on separate computers in each room to measure student-instructor interactions. There were no instances in which subjects in the CS2 course needed to be split across two rooms.

There were 19 recorded laboratory sessions in which CS1 subjects worked individually, 12 recorded laboratory sessions in which CS1 subjects worked in pairs, and 12 recorded laboratory sessions in which the CS2 subjects worked in pairs.

**Step 3:** Subjects from the CS1 course were given a post-study survey to gain the student's perspective of student-instructor interactions. Also, interviews were conducted with the instructors of both courses to gain further insight into the study results.



**Figure 1: Study Procedure**

### 3.5. Evaluation Criteria

The researcher who monitored the laboratory sessions used a software tool specifically designed for this study to record interactions between subjects and instructors. Whenever a subject raised his or her hand, the researcher would start a timer in the software tool. This timer would continue to run until the instructor addressed the subject's question and began interacting the subject. At this point, a new timer was started to measure the duration of the interaction. The tool allowed multiple timers to run simultaneously so that multiple different interactions could be monitored at the same time.

Only interactions initiated by student subjects were counted. Occasionally, the instructor from the CS1 course would walk around the room and visually identify students who appeared to be having problems and provide assistance. These occurrences of assistance were not recorded as the subject did not provide any indication of needing assistance. The instructor from the CS2 course did not engage in this behavior. The potential impacts of this (and other evaluation criteria) are discussed with other threats to validity in section 5.

Occasionally, subjects who spent multiple minutes waiting for instructor assistance would lower their hands before their question could be addressed by the instructor. In this case, the initial timer was left running until the researcher could ascertain whether or not the subject had abandoned the question or was merely tired of holding up his or her while waiting. If the subject re-raised his or her hand after the instructor finished assisting another student, it was treated as an indication that the subject still had the same question. If the subject did not make any indication that they still needed assistance, the timer was reset.

Any questions that began after the end of the class period were not counted; however questions which began before the end of the period, but were not addressed or answered until after the end of the period were considered. This only affected a small number of questions.

## 4. Data Analysis and Study Results

This section provides an analysis of quantitative data gathered from student-instructor interactions recorded during laboratory sessions and qualitative data gathered from a post-study survey completed by students and interviews with the course instructors. The results are presented for each hypothesis. Additional results which do not correspond to any particular hypothesis are presented in section 4.4. An alpha value of 0.05 was used for statistical tests.

### 4.1. Hypothesis 1

To evaluate whether or not pair programming resulted in fewer student-instructor interactions, the number of questions and the number of questions per subject were recorded across the different laboratory sessions for each course. Table 2 compares the mean number of questions asked and the mean number of questions asked per student when working individually and when working in pairs in the laboratory sessions. The results show a lower mean number of questions asked by students in laboratory sessions where pair programming was used. There are similar results for the mean number of questions per student.

**Table 2: Comparison of the Number of Questions across Lab Sessions**

	CS1 Individual	CS1 Pairs	CS2 Pairs
<i>Number of Sessions</i>	19	12	12
<i>Mean number of Questions</i>	14.53	7.417	4.75
<i>Mean number of Questions per Student</i>	1.19	0.44	0.29

In order to determine whether or not the differences were statistically significant, a two sample t-test was used. In the CS1 course, using pair programming resulted in a significantly lower number of questions per student in comparison to working individually ( $p = 0.000$ ). Even when comparing the number of questions per individual student against the number of questions per pair teams (in order to account for pair teams being composed of two students), the pair programming still resulted in significantly fewer questions per student ( $p = 0.005$ ). When comparing subjects from the CS1 course using pair programming with subjects from the CS2 course using pair programming, there were significantly fewer questions per student in the CS2 course ( $p = 0.005$ ).

The post-study survey completed by subjects in the CS1 course contained questions related to hypothesis 1. The survey questions were based on a five-point Likert scale with responses ranging from a strong agreement response to a strong disagreement response. One question (question 3) dealt with whether or not students felt as though they needed more instructor assistance when working individually when compared to working in pairs. The mean response to this survey question is shown in Table 3. In order to determine whether the mean response of students was significantly different than the midpoint of the scale, a non-parametric binomial test was used. The responses to question 3 indicate that subjects felt as though they needed more instructor assistance when working individually when compared to working in pairs. The mean response to the other survey questions (that will be discussed later) as well as p-values from the non-parametric binomial tests are shown in Table 3.

**Table 3: Results of Post-Study Survey of Subjects in the CS1 Course**

	Mean	p-value < 3	p-value > 3
<i>Q3: Help needed from instructor</i>	2.47	<b>0.023</b>	0.977
<i>Q4: Complexity of questions</i>	3.00	0.500	0.500
<i>Q5: Likelihood to attend lab sessions</i>	3.89	1.000	<b>0.000</b>

#### 4.2. Hypothesis 2

Subject interactions with the instructor were measured to determine how wait times differed based on programming technique. Table 4 shows a comparison between interactions between the different laboratory sessions. The table shows a smaller mean wait time when pair programming was used (an average of 21.70 seconds) compared to the interactions when students worked individually (an average of 67.84 seconds) in the CS1 course).

**Table 4: Comparison of Student-Teacher Interactions between Treatments**

	CS1 Individual	CS1 Pairs	CS2 Pairs
<i>Number of Interactions Recorded</i>	264	89	58
<i>Mean Wait Time (in seconds)</i>	67.84	21.70	13.81
<i>Mean Interaction Time (in seconds)</i>	87.28	121.50	146.30
<i>Mean Total Time (in seconds)</i>	155.12	143.20	160.10

Two sample t-tests were used to determine if the differences in the mean wait times were statistically significant. In the CS1 course, using pair programming resulted in a significantly shorter wait time ( $p = 0.000$ ) when compared to the wait time for subjects working individually. The difference between wait time for interactions when pair programming was used for the CS1 and CS2 courses was not significantly different ( $p = 0.210$ ).

### 4.3. Hypothesis 3

The amount of time spent interaction with the instructor was recorded for each interaction. Table 4 provides a comparison of mean time that subjects spent interacting with an instructor for the different laboratory sessions. The table shows that using pair programming in the CS1 course resulted in larger amounts of time spent interacting with an instructor (an average of 121.50 seconds) when compared to interactions recorded in laboratory sessions where subjects worked individually (an average of 87.28 seconds for interactions).

To determine if the difference in the interaction times were statistically significant, a two sample t-test was used. The results show that the amount of time spent interacting with an instructor was significantly higher when subjects in the CS1 course worked in pairs as compared to when those subjects worked individually ( $p = 0.021$ ). Again, no such difference was observed in the amount of time spent interacting when using pair programming between subjects in the CS1 and CS2 courses ( $p = 0.429$ ).

To gain insights into the longer interaction times for the pair-based laboratory sessions in the CS1 course, subjects in the CS1 course were asked to describe the types of issues that they frequently had while working on assignments in the laboratory sessions. 67% of respondents indicated frequently have problems with implementing an algorithm to solve the assignment problem; 67% indicated that they had problems with compilation errors. 44% indicated that they frequently had problems related to language syntax or usage; 17% indicated problems regarding clarification of assignment instructions. When asked whether or not they felt as though the types of issues they frequently had was any different when using pair programming as compared to working individually, 33% responded affirmatively. Based on the results of Question 4 from the post study survey, subjects did not feel as though their questions were more complex when using one technique instead of the other ( $p = 0.500$ ).

### 4.4. Additional Results and Analysis

Subject attendance for laboratory sessions also differed depending on the programming technique that was used. When pair programming was used in the CS1 course, the average attendance for laboratory sessions was 33.33 students compared to 21.55 students for laboratory sessions where assignments were completed individually. The results of a two sample t-test indicate that attendance was significantly larger for laboratory sessions where pair programming was used ( $p = 0.000$ ).

A question from the post-study survey (question 5) asked subjects about differences in their likelihood to attend laboratory sessions when different programming techniques were used. Table 3 shows the mean response and results of non-parametric binomial tests for survey question 5 which asked if students were more likely to attend laboratory sessions when they worked individually as opposed to when they worked in pairs. Subject responses indicate that subjects were significantly more likely to attend laboratory sessions when pair programming was used ( $p = 0.000$ ).

## 5. Threats to Validity

A major threat to validity deals with difficulties in being able to gather precise measurements for student-instructor interactions. Interactions were only measured when subjects made some gesture indicating the need for assistance. Because the instructor in the CS1 course would occasionally work with a subject who appeared to be having difficulties, but had not indicated a need for assistance, other subjects who had indicated a need for assistance were forced to wait longer. The researcher monitoring the lab indicated that this was not a frequent occurrence. Alternatively, subjects may not have immediately indicated a

need for assistance if the instructor was busy assisting another student. It was not uncommon for two or more subjects to indicate a need for assistance immediately after an instructor finished assisting another student. This may indicate that average waiting times may be artificially low.

Initially there were plans to measure the number of questions abandoned by subjects and the amount of time before a subject abandoned their question, but this became impossible to measure because subjects would not always keep their hand raised for the entire amount of time that they had to spend waiting for instructor assistance. This made it impossible to determine if and at which point a student had actually abandoned that question.

There were also several instances of subjects seeking help from a neighbor or friend when working individually. The instructor chided students for this behavior, but because students were spread across multiple rooms, there was no way to prevent this behavior from occurring or measure the extent to which it occurred. This may have resulted in a lower number of recorded questions, especially during lab sessions where subjects worked individually. It may also indicate that when subjects must wait sufficiently long for instructor assistance, they become frustrated and seek assistance from a classmate.

Not all subjects completed the survey and the results may be indicative of only those subjects who frequently attended the lab sessions. The survey was handed out during a lab session in which the subjects were working individually, potentially skewing the results.

Subjects in this study were not made aware of the goals of the study in order to minimize bias; however, the instructors were fully aware of the study goals as well as the types of information which would be measured. It is possible that this could lead to potential bias in the results, but it is believed that the instructors did not have any preconceived notions of how pair programming would influence student-instructor interactions.

## **6. Discussion of Results**

In this section, the implications of the results from Section 4 are discussed for each of the original hypotheses posted in Section 3.1.

### **6.1. Hypothesis H1**

Hypothesis H1 postulated that the number of questions per student per laboratory session would be lower when using pair programming. Table 2 shows that when using pair programming, there are on average of 0.44 questions per student per lab session as opposed to an average of 1.19 questions when students work individually. This result was statistically significant even when comparing questions per student to questions per pair, suggesting that when students use pair programming they are significantly less reliant on the instructor for help. Results from the post-study survey also indicated that subjects needed significantly less instructor assistance when working in pairs.

The instructor for the course further indicated that the assignments given to students for the pair programming lab sessions were designed to be more challenging and require more work than the assignments which were given when students worked individually. This makes it unlikely that the reduction in the number of questions was due to the assignments being trivial or too easy.

Subjects in the CS2 course had an average of 0.29 questions per student per laboratory session. This was significantly lower than the average number of questions per student per laboratory session in the CS1 course. When asked about potential reasons for this result, the instructor for the CS2 course indicated that it was likely due to two possible reasons: that CS2 students needed less assistance because they had more programming experience, and that laboratory assignments tended to build on top of

each other (e.g. for one assignment subjects implemented in a linked list; in the next assignment they implemented a stack using a linked list). Either or both reasons may have reduced the number of questions. Based on the results, hypothesis H1 is accepted.

## 6.2. Hypothesis H2

Hypothesis H2 postulated that the amount of time a subject would need to wait for instructor assistance would be lower when using pair programming. Table 4 shows the average time subjects spent waiting before their question could be addressed by the instructor. The average wait time for an interaction asked in a pair programming lab was 21.70 seconds as opposed to 67.84 seconds for an interaction asked in an individual work lab. The difference was statistically significant.

There were a few responses from subjects on the post-study survey indicating that they did not feel as though they received enough assistance during the lab sessions, but did not specify if this was independent the programming technique that was used. One respondent indicated a preference for pair programming because the respondent didn't have to spend as long waiting for the instructor. Based on the results, hypothesis H2 is accepted.

## 6.3. Hypothesis H3

Hypothesis H3 postulated that the amount of time subjects spent interacting with the instructor would be lower when using pair programming. Table 3 also shows the average time subjects spent interacting with an instructor. The average interaction time was 87.28 seconds for individuals compared to 121.50 seconds for pairs. This difference was also statistically significant, but exactly the opposite of the assumption of the hypothesis.

Results from the post-study survey indicate that subjects in the CS1 course did not feel as though the questions that they had differed in complexity when pair programming was used. 53% of respondents indicated that there was no difference at all, whereas the remaining respondents were equally split in believing that their questions were more complex when using pair programming (or working individually). When interviewed regarding this result, the instructor stated that when using pair programming student questions tended to be about the program logic or function and that these types of questions take longer to answer compared to questions about language syntax or compiler errors which occurred much less frequently when pair programming was used. Based on the results, hypothesis H3 is rejected.

## 6.4. Additional Results

Attendance was also significantly higher when pair programming was used. A large number of subjects' responses on the post-study survey indicated that this was partially due to a subject not wanting to let his or her partner down. This may be related to what previous researchers have described as "pair pressure", a tendency for students to be more motivated when using pair programming due to a shared responsibility [9].

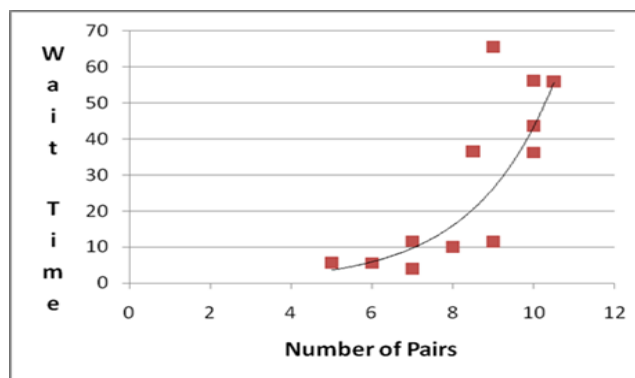


Figure 2: Number of Pairs vs. Mean Wait Time (in seconds)

Regardless of whether pair programming was used or not, results from both the courses showed strong correlations between attendance and the number of interactions that took place. Furthermore, there was a very strong correlation in the CS1 course between attendance and the average wait time per interaction ( $r = 0.784$ ).

Figure 2 shows those results from the CS1 course fit with an exponential curve. This suggests that further increasing the number of subjects in a laboratory without increasing the number of instructors is likely to result in significant increases in the mean wait time for subjects. During the interview, the instructor for the CS1 course mentioned that, ideally, there should be no more than 25 students in a laboratory session, which fits with the implication of higher wait times for more students.

## 7. Conclusion

Pair programming has numerous benefits ranging from improvements in the quality of student work to increased rates in the retention of students majoring in computer science. The results from this study indicate that pair programming is also useful in situations where the number of computers in the classrooms is limited or the class sizes are large. Even if these conditions do not exist, the use of pair programming is still beneficial because it provides students with more access to the instructor and reduces the amount of time that they spend waiting for assistance.

## 8. References

- [1] V. R. Basili, G. Caldiera, H. D. Rombach, "The Goal Question Metric Approach," Technical Report, Department of Computer Science, University of Maryland, 1994, <ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf>.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, MA, 2000.
- [3] J. Carver, L. Henderson, L. He, J. Hodges, and D. Reese, "Increased Retention of Early Computer Science and Software Engineering Students Using Pair Programming", In Proceedings of the 20<sup>th</sup> Conference on Software Engineering Education & Training, Washington D.C., USA, 2007, pp. 115-122.
- [4] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Effects of Pair-Programming on Performance in an Introductory Programming Course", In SIGCSE Bulletin 34(1), ACM, New York, NY, 2002, pp. 38-42.
- [5] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Impact of Pair Programming on Student Performance, Perception, and Persistence", In Proceedings of the 25th International Conference on Software Engineering, Washington D.C., USA, IEEE Computer Society, 2003, pp. 602-607.
- [6] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "Pair Programming Improves Student Retention, Confidence, and Program Quality", *Communications of the ACM* 49(8), ACM, New York, NY, 2006, pp. 90-95.
- [7] A. Radermacher, G. Walia, O. Myronovych, S. Abufardeh, and R. Rummelt, "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/>
- [8] L. Werner, B. Hanks, C. McDowell, "Pair Programming Helps Female Computer Science Students" In *Journal on Educational Resources in Computing* 4(1), ACM, New York, NY, 2004.
- [9] L. Williams, R. Kissler, "The Effects of 'Pair-Pressure' and 'Pair-Learning' on Software Engineering Education", In Proceedings of the 13th Conference on Software Engineering Education and Training, IEEE Computer Society, Austin, TX, 2000, 59-65.
- [10] L. Williams, R. Kessler, W. Cunningham, R. Jeffries, "Strengthening the Case for Pair Programming", In *IEEE Software* (17)4, IEEE Computer Society, 2000, pp. 19-25.
- [11] L. Williams, L. Layman, J. Osborne, N. Katira, "Examining the Compatibility of Student Pair Programmers", In Proceedings of the Conference on AGILE, IEEE Computer Society, Washington D.C., USA, 2006, pp. 411-420.
- [12] L. Williams, *Introduction to Pair Programming, Version 2*, YouTube video, 2008 [http://www.youtube.com/watch?v=rG\\_U12uqRhE](http://www.youtube.com/watch?v=rG_U12uqRhE)
- [13] L. Williams, E. Wiebe, K. Yang, M. Ferzli, C. Miller, "In Support of Pair Programming in the Introductory Computer Science Course", In *Computer Science Education* (12)3, 2002, pp. 197-202.