Pair Formation in CS1: Self-Selection vs Random Pairing

by
Aurelia T. Williams, B.S., M.S.

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

Pace University

School of Computer Science & Information Systems

June 6, 2007

Dissertation Signature Page

We hereby certify that this dissertation, submitted by Aurelia T. Williams, satisfies the dissertation requirements for the degree of Doctor of Professional Studies and has been approved.

____ - ___________
Joseph Bergin Date
Chairperson of Dissertation Committee

____ - ____________
Fred Grossman Date
Dissertation Committee Member

____ - ____________
Frances Gustavson Date
Dissertation Committee Member

School of Computer Science and Information Systems
Pace University 2006

An Abstract

Pair Formation in CS1: Self-Selection vs Random Pairing

by
Aurelia T. Williams

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

School of Computer Science and Information Systems

Pace University

September 2006

Attrition in Computer Science education is at its highest level in the last 20 years. Many researchers are searching for practices to decrease the attrition rate nationwide. Pair programming has been borrowed from industry and placed in today's computer science academic environment. Many studies have shown that pair programming increases students' confidence levels and aids in the retention of those students that have selected computer science as their academic major.

This research builds on this premise and investigates the effects of self-selection and random methods of pair formation. These methods of pair formation were used in beginning computer science courses at a historically black college and university (HBCU) to statistically demonstrate that students prefer to self-select their partner. Honoring this preference aids in the students overall satisfaction with beginning CS courses therefore decreasing the students possibility of changing their academic major. Data will show that in a CS1 course, the election to implement random or self-selection pair formation is trivial and students will perform equally whether the pairs are assigned or self-selected, however self-selection aids in the satisfaction of a student and therefore impacts their decision to retain computer science as their major.

Acknowledgements

Embarking upon the road to obtain a terminal degree was a major undertaking for me – a wife and mother of 2 very young children. However this goal was pressed upon me by a wonderful mentor and encouraged and supported by many family members and friends who told me that the goal was indeed obtainable in many moments when even I didn't believe in myself. Each of these persons deserves their own page of thanks but space is limited so I will do my best in just a paragraph or two.

To my husband, Aurthor C. Williams, Jr., thanks for all of your patience and understanding during my 3 year journey. I know there were many days when I just couldn't be satisfied and all of my frustration was directed toward you – my closest friend. We shared high moments and very low moments during this tenure and I sincerely that you for not taking me up on some of my offers. My love for you is endless!

To my son, Azad, thanks for understanding when Mom couldn't read your book because she needed to read her own and for responding when Mom would say "can you just watch TV for a little while". To my son, Atrell, you don't even know how much you gave up at 7mos so that Mom could pursue her degree but I do and will never forget. To my daughter (via the EASY button), Jhanae, you probably noticed the largest difference. Thank you for adjusting into the role of the more responsible big sister. All of you make my heart sing.

To my parents, Lewis & Constance Taylor, what can I say? Your support and encouragement are indescribable. That you for the many prayers sent on my behalf. Thank you for the many sleepovers with the kids just so that I could finish one last thing – although it never seemed to be the last. Thanks for the nagging and constant reminders to make me get things done even when I didn't want to hear it. Thanks for words of encouragement when I could no longer see the light at the end of the tunnel. Had it not been for the two of you, I am sure this degree would not have been possible. Thank you! Thank you! Thank you!

To my Aunt Beverly, thanks for coordinating the kids' schedules with my parents via email. Often it resulted in you taking the kids out to dinner whenever I just needed a moment to gather my thoughts. Your support of my journey is greatly appreciated.

To my Aunt Linda, thanks for the many prayers via the telephone when I was on my way to work and felt completely overwhelmed. You are forever loved and appreciated.

Moving to my work family, first I would like to thank my mentor, advisor, Dean, and friend, Dr. Sandra J. DeLoatch, as a mentor you encouraged me to pursue a terminal degree as it is required to have a career in academia at a time in my life when returning to school was farthest from my mind. As an advisor, you have seen many things that I have not seen in myself and you have shared your stories in an effort to encourage me to pursue my dreams. As my Dean, you provided financial support for classes and

travel to NewYork as well as release time at NSU so that I could have more time to study. As a friend, you have encouraged me to pursue even bigger dreams – some I am beginning to picture as well. For all that you have done and continue to do, I am eternally grateful.

To my buddy Marvin, you always have an ear for me to bend. You let me vent, gripe, and complain and then you tell me to get over it and do what I need to do. Thank you for your kind words of encouragement and having the courage to tell me when I am wrong. I really appreciate your friendship and your role as the big brother that I never had.

Thanks to my departmental chair, George Harrison, for scheduling my classes to teach on Tuesday and Thursday only to give me daylight hours to work on my assignments. Thanks to my colleagues, Jonathan, Cheryl, George and Nora for allowing me to conduct my research in your classes and for administering the plan as written. Thanks to my teaching assistants, Nikki, Michael, TJ, Wynton, Pam and Tom for grading the programming assignments and administering and collecting the surveys. Your participation was invaluable. Thank you, Yvette and Wallace, for your assistance in the creation of the online survey website.

Thanks to my Pace advisors, Joe and Fred, for your patience and incredible insight into the doctoral process. Both of you are extremely intelligent men with fluency in many topics. Joe, I especially appreciate all of the words you shared to keep me motivated and the words that you didn't when I procrastinated. You two are a wonderful good cop – bad cop team. To my statistical advisor, Dr. James Nolan, thanks for demystifying the world of research statistics at a time when my ignorance sent me to the ER for my one and only anxiety attack. Many thanks to Dr. Laurie Williams for the development of a process in which I was able to extend in research.

To my group 2 members, Ted and Dick, thanks for your assistance and participation in team assignments. A special thanks for agreeing to begin telephone group meetings at 9 pm which allowed me to get my kids into the bed to minimize distractions. To Mary, a special group 2 member, I really appreciate all of our conversations about any and everything – those that were grounding and encouraging and those that provided an "in" to the class of 2006 as a whole. Thanks for the rides to and from LaGuardia Airport and most importantly thanks for inviting me and my entire family over on graduation eve for dinner and entertainment. The girls on the violin were outstanding!

Overall, I must express that the experience I have gained, the things that I have learned, and the memories I have shared are PRICELESS. If it had not been for all of you and what you do and have done then I would not be the person that I am. Thanks to you all!

Dr. Aurelia Taylor Williams

(doesn't that sound AWESOME)

TABLE OF CONTENTS

LIST OF TABLES AND CHARTS

Chapter 1

1.0 Introduction

Pair programming is a software development practice in which two programmers work together at one computer, side by side collaborating on the same algorithm, code, or test [68]. One of the pair is identified as the "driver", who has control of the keyboard and actively types at the computer and the other is identified as the navigator, who constantly watches the work of the driver and identifies defects, thinks of alternatives, and asks questions. The two are constantly working as partners to develop code. In industry, pair programming has been used sporadically for decades. [70] Only in recent years, computer science educators have begun encouraging the use of pair programming in the classroom, mainly due to the fact of evidence from industry stating the major benefits of the practice. Prior to this time, such collaboration was considered cheating; educators were concerned about allowing pair programming because the approach had the potential for only one of the pair to learn while the other would passively allow their partner to complete the work. However, research has shown that pair programming in the computer science classroom is generally valuable [70] and the benefits associated with the student learning far outweigh the concern of whether pair programming is considered "authorized" cheating.

In the classroom environment, pairing students has many benefits. In industry, software developers generally spend 30% of their time working alone, 50% of their time working with one other person, and 20% of their time working with two or more people. [40] Pairing is one cooperative and collaborative pedagogy that introduces the students to the collaboration required of a student's future professional life [40]. Other benefits shown include more students passing the course, higher quality programs, less time to complete programming projects, increased student satisfaction, increased numbers of students continuing with a computer related major, and possibly better exam scores (when drop rates are factored in)[32]. Pairing a student with one partner for the entire semester is beneficial for both students and educators, but may not be optimal.[70] Some students may be concerned with the skill level and work ethic of their partner. Pair rotation may be a more viable solution. *Pair rotation* is defined as when students pair with different classmates throughout the semester. With each partner, each student assumes both the role of driver and navigator for varying periods of time in each programming session [70].

Pair programming also introduces other aspects that must be addressed when used in the academic environment. Peer evaluation and pair formation are two major concepts that affect the implementation of the pair programming pedagogy. *Pair formation* is defined as the method in which pairs are formed. Research has been performed that investigates various methods of pair formation and documented which methods work best at a students' level of proficiency in computer science [40]. *Peer evaluation* is defined as the mechanism used by students to evaluate their partner during a pair programming activity to maintain individual accountability. Pair evaluation has been used by many researchers in various forms and at many levels of the students' academic career. Peer evaluation's main goal is to give the students a voice in the assessment of their partner's performance.

This thesis combines the major aspects related to pair programming (pair formation, pair rotation, and peer evaluation) in the academic environment and reports on the practices used to facilitate students enrolled in CS1 courses at an HBCU.

This thesis is organized as follows: Chapter One gives an introduction to the research; Chapter Two provides the research motivation, background, definition, and related research activities for pair programming, pair rotation, pair formation, and pair evaluation; Chapter Three provides the methodology in which all aspects were used to provide quantitative data used for measurement; Chapter Four provides the observation, results and analysis of the data measured to suggest methods of pair formation that should be employed in CS1 courses when student satisfaction, retention of students, and maintaining individual academic performance are the main goals.

Chapter 2

2.0 Background and Related Work

2.1 University Setting (Background and Research Motivation)

Norfolk State University (NSU) is one of the largest predominately black institutions in the nation. "It is committed to pursuing its vital role of serving the people of the Hampton Roads area. NSU is a public, urban, comprehensive University offering programs at the undergraduate and graduate levels. NSU emphasizes teaching over research, and is classified as a Carnegie Mellon Foundation Master's/L: Master's College and University"[1]. Founded in 1935, NSU adheres to the "traditional purpose of the Historically Black College and University (HBCU) and espouses the tradition of service to its students, its alumni, the academy, the Commonwealth of VA, the nation and the world. NSU's primary mission is to provide an affordable high quality education for an ethnically and culturally diverse student population, equipping them with the capability to become productive citizens who continuously contribute to a global and rapidly changing society"[57].

As of fall 2005, NSU's total student population was reported at 6,096 students[44]. Of these students, 78% are full-time students, 87% are undergraduates, 64% are female, 86% are African-American and only 34% are residential. The mean (average) student age is 25. These numbers are especially significant to understand the student body that

NSU serves. Its students are generally older, full-time, non-residential students with full time jobs and other outside obligations such as families.

NSU is considered to be a liberal arts university with an emerging science and technology powerhouse. The School of Science and Technology (SST) is second only to the school of liberal arts. With its enrollment of 1720 students, 28% of the students enrolled at NSU are science majors.

The SST consists of nine departments; Allied Health, Biology, Chemistry, Computer Science, Engineering, Mathematics, Nursing, Physics, and Technology. It offers undergraduate degrees in all departments as well as masters level graduate degrees in electrical engineering, optical engineering, materials science, and computer science. The school also has the most funded grants of any school at NSU. Faculty contributions in research, grantsmanship, service, and mentoring are exceptional. Graduates secure outside jobs and earn advanced degrees in record numbers[15].

The Computer Science (CS) department at NSU currently has 220 students declared as a CS major. During the period from 1999-2004, the CS department has seen constant attrition that mimics that of the national average. However, at NSU, this rate has grown by double digits since 2001. The national attrition rate is especially high for computer science majors as opposed to other majors due to the unrealistic career preview. Traditionally students enrolled in computer science programs are educated and evaluated individually. Students have also been trained to believe that the career of a computer scientist does not include social aspects. Many factors have been considered while trying to pinpoint the reason for the extremely high attrition rate. Among the highest is the satisfaction of CS freshmen majors at the conclusion of their first year of studies. This transition shows the highest number of “drop-outs” many due to not really knowing what computer science is before electing it as a major.

Curriculum changes have been implemented to address the freshman CS experience. Those changes include the addition of programming laboratories and the inclusion of the pair programming practice used for software development. This research will investigate the feasibility of pair programming in a university setting in computer science education at a Historically Black College and University (HBCU) and the methods used to form a pair. Specifically, in this study pairs will be formed either by random assignment or by self-selection where each partner directly self-selects with whom they want to work.

This research has a great impact to the field of computer science education because all documented studies to date that measure the efficacy of pair programming in an academic environment have been completed at majority institutions. Another aspect related to pair programming that has been studied is the retention of underrepresented students (female and minorities) in the computer science discipline. It is important to note that those studies investigated retaining minorities and females enrolled at majority institutions, where the total number of those students rarely represented more than 5% of the students enrolled in their departments. In the CS department at NSU, virtually all, 98%, of the students fall into the category of underrepresented students.

2.2 Extreme Programming

eXtreme Programming (XP) is a software development method that favors informal and immediate communication over the detailed and specific work products required by many traditional design methods[68]. XP was produced by Kent Beck and Ward Cunningham over a fifteen year period with a team in mind[4]. Ron Jeffries defines extreme programming as a discipline of software development based on values: communication, simplicity, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.[23] The practices are: the planning game, small releases, metaphor, simple design, testing, refactoring, ***pair programming***, collective ownership, continuous integration, 40-hour week, on-site customer, and coding standards.[4] Beck and Cunningham believe that teams should put more weight on individuals and interactions over processes and tools and responding to change over following a plan. Pair programming fits well within XP for reasons ranging from quality and productivity to vocabulary development and cross training. XP relies on pair programming so heavily that it insists **all** production code be written by pairs[68].

XP's dozen practices are appropriate for small to midsize teams developing software with vague or changing requirements. The methodology copes with change by delivering software early and often and by absorbing feedback into the development culture and ultimately into the code.[68] The XP approach credits much of its success to the use of pair programming by all programmers, regardless of experience[68].

Perhaps the best example of XP success has been the Chrysler Comprehensive Compensation System first launched in May of 1997. During a period of five months of system development, the only code defects that made it through testing procedures were written by someone programming alone. There have even been claims of up to 1500 lines of code being written and run correctly the first time the program was executed, when programmed in pairs [67]. Industry programmers who work in pairs report higher job satisfaction than those who work alone (as many as 96% reported enjoying their job more when working in pairs [68]). It has even been reported that pairs can finish programming assignments in half the time as individuals [68]. The evidence is clear, programmers who develop code in pairs write better code faster than those who program alone.[10] For this reason, educators have begun to adopt the practice of pair programming as a collaborative learning tool for educating computer science (CS) majors and those students interested in working in the information technology (IT) field.

2.3 Pair Programming

Pair programming is defined as the practice of all production code being written with two people looking at one machine, with one keyboard and one mouse[4]. There are two roles in each pair. One of the pair, called the *driver,* types at the computer or writes down a design. The other partner, called the *navigator,* has the task of monitoring the driver's work for tactical and strategic defects. Tactical defects are local errors such as syntax errors, typos, or improper method calls. Strategic errors involve, as the name implies, problems in strategy, leading incorrect or inefficient solutions to programming problems. Because the navigator is not involved in the highly focused work of typing the code, they can afford to look ahead and monitor overall strategy. Still, the driver and navigator continuously spend time brainstorming and negotiating their course of action[60].

Ideally, communication between the driver and navigator is regular with the driver explaining what he or she is doing or asking for advice and/or confirmation of what he or she has done and the navigator providing feedback. The feedback can be pointing out simple errors, providing encouragement, or challenging the driver to explain what was done. An effective programming pair has an active relationship with both halves actively involved in the development of the code or design[4, 60]. Balance in the partnership is preserved by having the two switch roles on a regular basis.

Pair programming is not one person programming while another person watches. It is a dialog between two people trying to simultaneously program, analyze, design, test, and understand together how to program better.[4] Pairing is a dynamic process.

Pair programming profits from the fact that there is not one mind working but two. Two minds develop ideas around the same set of problems, but from a different kind of perspective[53]. It has many benefits and advantages. It has also been stated that if people choose to program solo they are more likely to make mistakes, more likely to overdesign, and more likely to dismiss the other practices of XP, particularly under pressure.[4] In industry, if two people pair in the morning, then they should be paired with another in the afternoon.[4]

Nosek ran an experiment [42] in which 15 full-time programmers were organized into five solo programmers and five groups of pair programmers to write a script that performed a database consistency check. Both solo and pair programmers were given 45 minutes to complete the task. Nosek measured the impact of pair programming on five independent variables: time, readability, functionality, programmer confidence, and enjoyment. Scores were better for pairs in all measures, but the differences in the time and readability scores were not statistically significant. The combined score of these four variables was about 35% better for pairs than for solo programmers. Nosek notes but does not elaborate, that "the scripts produced by some of the pairs were significantly better than the scripts they bought from expert consultants."

Williams and Kessler point out seven behaviors that happen naturally when programming in pairs [67] These behaviors make it possible for pair programmers to finish their tasks about twice as fast as solo programmers, with the use of approximately the same overall resource expenditure, and still generate higher quality products. These seven behaviors are:

1. *Pair pressure:* Williams and Kessler report that, in a junior-senior software development course, most students worked harder when paired because they did not want to let their partners down [65]. The students do not intentionally put pressure on their partners. Nevertheless, each feels the push to move forward.
2. *Pair negotiation:* The two programmers work together to solve a problem. They have different prior experiences but a common goal. The two have to negotiate to share a common approach.
3. *Pair courage:* The programmers give each other the courage to do something they might not do if working alone.
4. *Pair learning:* The programmers can learn from their partners' continual critique and review. Additionally, because the programmers work closely together, their knowledge, including programming tips, design skills, tool usage, is transferred between them constantly.
5. *Pair trust:* Pair programmers work in a collaborative fashion. They learn to trust their partners to get the work done.
6. *Pair review:* When working in a pair, both programmers review their joint product continuously. This review technique has been shown effective and enjoyable.
7. *Pair debugging:* Debugging is a tedious and laborious task. However, if we can discuss the problem with someone, we might find new ideas and solutions.

2.3.1 Advantages of pair programming

In industry, investigative studies [11] have shown that pair programming has inherent benefits, including:

- Continuous Reviews. Pair programming's shoulder-to-shoulder technique serves as a continual design and code review, leading to more efficient defect removal rates as compared to two solo programmers.
- Problem solving. Experienced pairs refer to the team's ability to solve "impossible" problems faster.
- Learning. Pair programmers repeatedly cite how much they learn from each other [4]
    - Team Building and Communication. Pair programmers become more familiar with each other, which serves to improve team communication and effectiveness.
    - Knowledge Management. Research shows that pair programming is an effective Knowledge Management technique [45]. In the course of pair programming, multiple programmers are exposed to each piece of code, reducing the impact of losing staff. When a pair is split, they both take domain and coding knowledge to other programmers that they pair with in the future. The continual interaction between pair programmers provides an environment that promotes knowledge sharing, and collaborative knowledge discovery [45].
    - Quality and Productivity. Studies have demonstrated through anecdotal, qualitative and quantitative evidence that incorporating pair programming into a software development process will help

yield software products of better quality in less time with happier, more confident programmers [27]. Developers are less likely to produce defective code, because a teammate is watching them. Developers are less likely to spend time on things other than work; therefore productivity increases [11].

Cockburn and Williams conducted an anonymous survey of professional pair programmers and student pair programmers to determine whether or not pair programming was more enjoyable than individual programming[11]. The results of the survey showed that more than 90% of those that practice pair programming enjoy their work more because of it. These results were also confirmed in a study conducted by Cliburn[10] at a small college where the vast majority of students are residential; he reported that the majority of students liked working in pairs on the projects (76.9%). It is also noted that the students who did not like pair programming performed well in the course (typically receiving an “A” or “B” final grade). That result is also consistent with [55] where the best students typically do not enjoy working in pairs since they could probably solve the problem alone.

Pair programming has been used in the undergraduate computer science classroom with more positive reports than negative reports[61, 64-66] Educators and students have both expressed satisfaction and enjoyment using pair programming to teach and understand computer science topics. Studies have identified the following advantages:

- *Pair programming improves quality, teamwork and communication*.[51] Researchers observed that pair programming conducted with senior-level undergraduates led to a 15% reduction in total number of observed defects with a 15% increase in total development time[62, 68]. It was also found that students working in pairs were more satisfied, solved tougher problems, faster, and had improved team communication and overall team effectiveness [32, 61, 68]. Pairs also spend less time working on assignments than individuals [8] and seem to keep each other on task while they work.[10]. These results are consistent with reports of pair programming used in industry.

- *Pair programming improves retention and confidence* [51]. Researchers demonstrated that pair programming students were significantly more likely to complete the course and achieve a grade of C or higher [33, 35, 40]. The pairing students demonstrated higher confidence in their project work in comparison to students who did not pair program [33, 35]. Additionally, the students who paired were more likely to take the next programming course than those who did not pair program[35, 40]
- *Pair programming leads to improved comprehension and learning* [51]. Students have reported that pair programming allowed them to understand the project better and improve their learning and comprehension of unfamiliar topics [51]. It also supported in studies via examinations scores that pair programming students perform better on individual tests [40].

Educators have also identified the following advantages:

- *Instructors feel more positive about the class*. Students are happier, and hand in assignments on time with higher quality. The students are more satisfied. [67]

- *Students are more self-sufficient.* Students no longer look to the teaching staff as their sole source of technical advice. In pairing, when one partner does not know or understand something, the other almost always does.[67]
- *Cheating is reduced.* Collaboration is legitimized. Pair pressure causes the students to start working on projects earlier and to budget their time more wisely. Students have a peer to ask for help, therefore do not feel as helpless.[67]
- *Grading is significantly reduced.* Two students submit one assignment and typically share the same grade[67] therefore reducing the workload[72].

Advantages specific to students are as follows:

- *Reduced frustration*. Small typos or a missing semi-colon can host hours of debugging time. Waiting, minutes, hours, or even a day to get help from a teacher is alleviated because the pair may be to provide an answer immediately.[67]
- *Negative self-talk is reduced[67]* Students have the conviction to ask for help. If they realize that they don't understand something and their partner doesn't understand it either, then it must hard. The students know that they are not the only ones.
- *Students get to know each other on a personal basis*. Students will not feel like a face in the crowd because they are 'forced" to work with a peer[67]. Pair rotation helps students get to know more of their peers.[67]

One researcher found that most of the students thought having a partner was beneficial to them (76.9%). Most students thought pair programming improved their grade (61.5%), while 30.8% thought there was no change [10].

2.3.2 Disadvantages of pair programming:

Students report only a few negative aspects to pair programming. *Pair programming could lead to schedule issues, pair incompatibility, and unequal participation* [8, 51]. Some students in a study opposed the pair programming practice due to pair incompatibility, schedule-related issues between pairs (finding times they can meet together to work on projects), and unequal participation by the individuals in a pair [8, 51].

Educators suggest a good solution to help students find times to meet is to have a lab associated with the course that students can use to work on assignments in pairs [33, 72]. This gives them one or two hours per week that they are guaranteed to find time to work together. For unreliable partners, a solution is to assign new pairs for each project. Thus, students will not have to work with the same poor partner for more than one assignment.

While pair programming is certainly not perfect for college courses, it seems to have many advantages that make its implementation worthwhile. [10]

2.3.3 Concerns with pair programming:

Pair incompatibility could be a major cause for concern. Several studies have shown that matching pairs based on skill level is beneficial for productivity [70]. Also, students with a higher self-reported skill level report the least satisfaction when they pair program with students of lesser self-reported skill level [55]. In the Cliburn study, one student, who said that pair programming lowered his grade mentioned, “it was more of a hassle to teach my partners the code they didn’t understand. Thus I had less time to focus on the projects.”[10] Researchers found that students produce their best work when they are paired with a partner of equal perceived skill level [56].

Educators have tried many methods to address the pair compatibility concern. The methods include pair formation, peer evaluation, and pair rotation; all will be addressed independently in the next sections.

2.4 Pair Rotation

In organizations where there is more than one pair, pairs often rotate partners after the completion of a task or subtask. Pair rotation allows the programmers to learn different areas of the system, and learn from the skills of different partners.[45]

The advantages of rotating the programmers are that they learn more about the whole product by pairing with many team members, they team with the person who can help them the most on a particular task, and communication and teamwork increases significantly [49] Rotating pairs offers knowledge management benefits because there are always at least two developers that know about a particular piece of code. Also, the more people that have paired on a particular piece of code, the better the quality will be and the easier it will be to discuss the code or to extend or change it [64].

In a course environment, pairing a student with one partner for the entire semester is beneficial for both students and educators, but may not be optimal. From an educational perspective, the term *pair rotation* [67] is used to denote when students pair with different classmates throughout the semester. With each partner, students assume both the roles of driver and navigator for varying periods of time in each programming session. Pair rotation was used in the freshman (CS 1) Introduction to Programming course at North Carolina State University (NCSU), and in the advanced undergraduate software engineering (SE) course at North Carolina State University (NCSU) during the Fall 2002 (SE) and Spring 2003 (CS 1) semesters.

When two programmers are first paired, there is often a “jelling period” during which time each is adjusting to the working habits of the other, learning the other’s strengths and weaknesses, and learning how to communicate effectively. While productivity is somewhat less during this period, it soon increases as the pair becomes comfortable working together. Anecdotal evidence suggest that this adjustment period varies from a few hours to several days, depending on the individuals.[62]

The idea of swapping partners has been used in computer science education. Educators and researchers have different reasons for trading partners and have documented students’ perceptions of the idea of partner trading. DeClue’s rationale for performing this switch included simulating the real-world work environment and investigating the possibility that switching partners might have a generalized positive impact on the production of quality code[14]. Specifically, after each of the three two-week phases, students were reassigned to new partners. The duration of a phase lasted approximately two weeks. One student stayed with the project he or she had been working on, while the second student was placed with a completely unfamiliar code set. He believes from a pedagogical standpoint, the forced trading of partners emphasized the practical importance of external design documents, test plans, and writing code for readability and he found that none of the students said they resented the fact that they had to switch partners.[14]

In another study when students were surveyed to identify whether they found pair rotation to be advantageous by answering the following question: “Do you think it was a good idea to change partners after each assignment?” [51] Research suggests that students who preferred pair rotation to those who did not are significantly greater than 2:1[51]. This perception was also identified in [10] where 76.9% of the students liked working with different partners. They mentioned reasons such as, “It exposed me to more people in my class” and “maybe a different partner knows a little more about a certain area, and when paired with them you can gather more.” However, in informal discussions with students, it was sensed that most would have preferred to work with the same partner each time, if they liked the person, their schedules were compatible, and the student was a hard worker. One student stated, “You’ll get to know him/her so it’d be easier to work (with them). If the partner is bad, it’s good that one can have another one.” The students who were not sure or thought it was bad to work with a different partner each time seemed to concur with this statement and added, “it is bad to work with different partners because it is hard to fit your schedule to theirs.” This seems to be one of the most difficult issues about pair programming for students, finding times outside of class for partners to meet[8, 10, 51].

2.4.1 Advantages of pair rotation

The benefits of pair rotation documented in industry have also been documented in computer science education. Pair rotation provides students with exposure to more classmates [51] The concept enabled the students to work with different classmates and learn new ways to solve problems[10, 51] and helped students get to know more of their peers.[67] Pair rotation also accommodates a student’s desire for a new partner [51]. It has permitted students to switch partners for future projects if they were not compatible with his/her current partner [51]. Since rotation is handled via regular classroom management it is believed that this aids in the student’s overall experience with pair programming. In one study [10], the overwhelming majority of students (84.6%) felt that their experience with pair programming would have been different if they had programmed with the same partner on each assignment. Without pair rotation, the instructor is presented with the problem of needing to break up another pair in order to separate and reassign the students in dysfunctional pairs [51]

Pair rotation has been found to be beneficial to educators also. Educators benefit from pair rotation by effectively dealing with dysfunctional pairs and inactive students via peer evaluation[51]. When pair rotation is implemented with peer evaluation, educators have multiple forms of feedback on an inactive student. This feedback conquers the "one word against another's" situation. Peer evaluation will be addressed in the next section.

2.4.2 Disadvantages of pair rotation

Pair rotation, just like pair programming, has its documented disadvantages as well. The main disadvantage that students have described is the need to re-adjust [51]. Pair rotation could be inefficient, as it requires the students to re-adjust to a different partner and his/her programming style at the beginning of every project [10, 51]. This adjustment period is often referred to as the "jelling" period Another disadvantage is the loss of a compatible partner [51]. If a student's current partner is highly compatible, pair rotation may lead to the loss of a perfectly good partner, and introduces the risk of not having a good partner for the next project [51]. The disadvantage to educators is the need to reassign pairs. Pair rotation requires teaching staff to assign/create pairs using their method of pair formation at each cycle.

Of more specific importance, however, is the role that changing partners had on the code production. Changing partners, as might have been predicted, did not prove to be a real difficulty for the students[14]. In DeClue's study, the aspect of pair rotation seemed to strengthen some of the more difficult lessons related to code readability, using a coherent design, and using appropriate code documentation.[14]

2.5 Peer Evaluation

When students work in pairs, the potential exists for one partner to do most or all of the work, while the credit is assigned equally to both partners in the pair.[51] Educators have implemented peer evaluation in classrooms where pair programming is used in the practice of software development. *Peer evaluation* is the practice of one partner evaluating the contribution of the other partner. Peer evaluation's intent is to ensure that both partners contribute equally to the product that is developed. The actual implementation has been used in many ways, some educators have used the results from the peer evaluation to influence the actual grade on the programming assignment[51], others have just required the evaluation as a means to check on the pair, and the actual evaluation score has no influence over the final score of the programming assignment but may influence the final score in the class. Peer evaluations have been collected in many ways as well. Some researchers have used web-based forms[51], others have required a simple email depicting the contribution of each of the pair[10], and one researcher actually used a quiz with specific questions regarding the program to check on the student's participation in the code development[14].

Srikanth, et al[51], used a web-based peer evaluation tool, pair rotation, and required students to assign 0 to 20 points on each of the following five questions, which gave the partner a score of 0%–100% on their peer evaluation: Students understood that the answers to the five questions on contribution could impact their grade. The students were asked to evaluate their perception of the partner’s skill level and their joint compatibility on an ordinal scale:

1. Did the student read the lab assignment and preparatory materials before coming to the scheduled lab?
2. Did the student do their fair share of work?
3. Did the student cooperatively follow the pair-programming model (rotating roles of driver and navigator)?
4. Did the student make contributions to the completion of the lab assignment?
5. Did the student cooperate?

In Srikanth’s study, students in all classes were aware that their grades could be significantly affected by their peer evaluation. However, peer evaluations were handled differently in the beginning computer science (CS1) class than in the software engineering (SE) class. In the CS1 class, the instructors examined the peer evaluations at the completion of the semester. In SE, a student was approached by the instructor as soon as he or she had two peer evaluation scores of 80% or less. Depending upon the severity of the situation, the student was either given a warning or his or her grade on the assignment was dropped. The grade was dropped to no lower than the earned grade multiplied by the peer evaluation score. The partner of the violating student’s score was then increased to (the earned score * (1 + (1-partner’s peer evaluation score))) but not exceeding a value of 100.

Additionally in the study, the researchers queried the students on their perception of the effectiveness of peer evaluation in providing feedback to teachers and in motivating students to improve contribution. The majority of the students in both CS1 (63%) and SE (65%) agreed that peer evaluation was an effective means of feedback to the teacher. However, a large number of students from CS1 (53%) believed that peer evaluation was not a good source of motivating students to contribute.[51] Nonetheless, the researchers do believe there is a need to use peer evaluation in pair programming to improve the productivity of all students, both strong and weak.

In Cliburn's study [10], the researcher paired students with similar course grades. The first project was fairly simple and he used the grade on that project and the first exam as a basis for assigning partners in the future. Peer evaluations were added to the course to comprise 10% of the student's grade therefore reducing the influence of homework and projects to compensate for the addition of peer evaluations. The researcher now thought that students would be held accountable to each other, and if one student does all the work on an assignment, the student would indicate that in a peer evaluation and their partner would not receive the same score on the project. The researcher knew that students typically do not like to evaluate each other, so the evaluations were very short and confidential. After each assignment, students were required to send an e-mail to the instructor stating what percentage each of them contributed to the assignment. It is written that most e-mail evaluations looked something like:

```
Bob  50%
Bill 50%
```

The researcher, Cliburn, told the students that "ideally they will report a 50/50 split on the work yet their evaluation should not consider who typed what amount of code, but how much each contributed to the final project." [10]

Cliburn also experienced similar results to Srikanth, et.al when surveying the CS1 students regarding peer evaluations. When asked “did peer evaluations make you feel accountable to your partner,” for the most part (69.2%), students did feel that peer evaluations made them accountable to their partner. The students mentioned things like peer evaluations “encouraged you to put at least equal effort if not more” and “it protected me from having a partner not do anything.” However, almost one third (30.8%) did not feel the peer evaluations encouraged them to be accountable to their partner. One student said “I wouldn’t give my partner a bad evaluation if they did nothing.” Another student agreed saying, “Most of the time peer evaluations are garbage. If I would not have ever seen my partner and did the whole thing myself I would have still given them full credit.” Another surprising result the researcher found was that students did not feel that meeting regularly with their partner was necessary to earn a good peer evaluation. Students responded with statements like, “I feel I did most of the work therefore it was not necessary to (attend) all the meetings.” Another mentioned, “I met to get the work done. It really wasn’t for the evaluation.”[10]

Cliburn reports that “it seems that you cannot force the students to always give an honest and objective evaluation, but at least the mechanism is there to protect the students who want to be protected from a “parasite” (the term for students who do no work, relying entirely on their partner to complete assignments) partner”. [10]

The researcher also concluded that since peer evaluations have been added and students have been paired based on similar abilities, the quality of the students who finish the course has seemed to improve and there has never been a student to complain that their partner was a “parasite.” However, retention is lower (23.5% dropped the course during the first semester using peer evaluation).[10]

DeClue [14] used a peer evaluation similar to Cliburn for the first phase. DeClue required one student to remain with the current project he or she had been working on and the other student to be placed with a completely unfamiliar code set for the next phase. He required each student to assess either him or herself with regard to contribution to the pair in a particular phase. The question used was “Provide the name of your partner and state in percentage terms how much of Phase <n> your partner was responsible for. Give yourself a percentage as well. These two numbers must add up to 100%.” For the last two phases, he also administered a quiz that could only be successfully completed if the student had been highly involved with the production of the code. Two example questions were “what is the name of the class which contains your hash method?” and “what is the name of the text area object you used in your program?” He concluded that pair programming does challenge the assessor, however. When two students work as closely together as is called for by generally accepted guidelines for pair programming, the code produced must be credited to both students. Switching partners at the end of each phase helped defeat the feeling that once you were assigned a partner you were "stuck" so to speak. The quizzes at the end of phases two and three helped verify that each partner was contributing to the effort of code production. In DeClue’s opinion, this aspect of pair programming - difficulty in assessment - had less effect than might have been anticipated.[14]

The main advantage of peer evaluation is that it provides multiple forms of feedback. When a student is paired with only one student for the entire semester, it can be difficult to handle the situation in which one student is saying he or she did all the work while the other insists that he or she has contributed. To overcome this situation, instructors can apply pair rotation to get the input of several partners for each student and confidently deal with an ineffective student [51]. Peer evaluation also allows help with the handling of dysfunctional pairs. Since some researchers have included peer evaluation, they have never had a student to complain that their partner did absolutely none of the work.[10]

The greatest disadvantage of peer evaluation is the need for peer evaluation. When two students work as closely together as is called for by generally accepted guidelines for pair programming, the code produced must be credited to both students. Switching partners at the end of each phase (or programming assignment) helped defeat the feeling that once you were assigned a partner you were "stuck" so to speak[14]. Pair rotation requires the teaching staff to consider the peer evaluation factor and its resulting issues. When all students work alone, the grade determination process is based on student contribution and is less complicated[51]. Also, there is a need to monitor that one partner does not dominate the pair or that one partner is burdened with the entire workload[40]. There must be some mechanism to ensure that students are contributing fairly.

A major concern of peer evaluation is the reliability of student responses. Most students do not want to evaluate their partners and many will not “turn in” their partner[40]. Students will often evaluate the performance of the pair with equal distribution of effort even when the partner has contributed minimally[10]. One can not force a student to give an honest and objective evaluation. Peer evaluation seems to work better in upper-level (advanced) undergraduate courses than in CS1 courses. This could be contributed to the maturity of the student but it has not been proven.

2.6 Pair Formation

Pair formation has been discussed in many research studies however, there is no clear method that accommodates all students. Some studies have suggested various ways to pair studies but there is a distinction in which methods work best for students learning to program (CS1) and advanced undergraduate students (3rd and 4th year students mostly in the software engineering (SE) course). CS1 students are typically 1st year freshmen students who have not had exposure to any program languages previous to their enrollment in the course. However in some cases these students may have been exposed to a software development methodology and high- level language in high school via an advanced placement (AP) course. These students are the exception at Norfolk State University (NSU) and are usually scholarship students so they are high achievers and perform well in most academic environments.

Self-selection pairing is defined as the student choosing his or her partner in which to pair. McDowell, et. al [33], allowed students to choose their own partners and remain with the same partner for the entire semester unless unforeseen circumstances occurred (scheduling problems, a student stopped showing up, etc.). This practice is also used indirectly in many research studies in which the students identify who they *do not* want to pair with [22, 51, 73]. It is then explained that the students are not assigned any student on their “do not pair with” list.

Random pairing is defined as the instructor identifying and assigning all pairing groups “ad hoc” that is without regard to factors that some research suggests increases pair compatibility such as personality type, skill level, self esteem, and self-assessment [67].

This practice was used in several studies including Nagappan, et. al [40], where a software program was used to make random partner assignments with pair rotation for each assignment. In a small percentage of cases, the random pairing led to incompatible partners, which led to conflicts during working.

Studies have formed pairs based on self-assessment. Thomas, et al. [55], had students rate themselves on a scale of 1 to 9 in terms of their programming ability and then made partner assignment based on these ratings. In some projects, students who rated themselves highly (7-9) were paired with students who rated themselves poorly (1-3). In other projects, students were paired with those who rated themselves similarly.

Williams and Kessler have also formed pairs based on personalities (whether someone is an extrovert or introvert) [67] in their research study using the Myers Briggs Personality Type Indicator. The indicator is often used in studies to classify people into one of the sixteen personality types based on the dimensions such as introversion, intuition, feeling and judgment. While the indicator may be a factor in determining productive pairs with professional programmers, it has been found that students' personality type and self-esteem are not critical for their compatibility.[25]

A particular study at Hanover College [10] in the CS1 level course tried a different approach to form pairs by grouping students from different cultural or ethnic backgrounds, thinking that it would be a good experience for college freshman to work with people from places different than their home. The algorithm required grouping students by priorities; first, by different cultural and ethnic background and secondly, pairing upperclassmen with lowerclassmen students. This pairing scheme was used on programming assignments one, two, and three with pair rotation. This study yielded surprising results, the individual exams showed that many of the students that scored highly on the programming assignments seem to have no idea how to do any programming on the exams.[10] Based on these results, the pairing scheme was abandoned for one that paired students of similar attributes. The study suggests that assigning partners so that students always work with someone at or near the same skill level seems to be the best strategy for ensuring that students do not get though the course only on the strengths of their partners[10] in the introductory course.

Another study conducted by Katira, et al.[26] aimed specifically at understanding compatibility found that:

- Pairs are more compatible if students with different personality types are grouped together in CS1
- Pairs are ***not*** more compatible if students with similar actual skill level are grouped together in CS1.
- Pairs are more compatible if students with similar perceived skill level are grouped together in CS1.
- Pairs are ***not*** more compatible if students with similar self-esteem are grouped together in CS1.

These results are not the same for the advanced undergraduate student. The study states the most significant result is that students prefer to pair with someone they perceive to be of similar technical competence[26]. This was found to be true in students learning to program as well as advanced undergraduate students. However, educators cannot predict this perception nor can pairs be formed based on this fact. The data suggests that pair compatibility in beginning courses may increase if pairs are formed by joining students of dissimilar personality type.[26]

This research uses a combination of all of the previous sections. Its focus investigates if self-selection or random pair formation with pair rotation and peer evaluations leads to increased satisfaction of the CS1 student enrolled at an HBCU.

2.7 Pair Programming in Laboratories

The greatest concern of implementing pair programming into the CS curricula that students have identified is the ability to coordinate schedules to find a time for both partners to work together[8, 10, 51]. Researchers have suggested including a lab associated with the class to assist with this problem[10, 33, 72]. This gives them one or two hours per week that they are guaranteed to find time to work together.

Wiebe, et al[60], in a study investigating the differences between paired and solo labs, found that in general, students in the pair-programming lab sessions showed a high level of interaction with each other. Students discussed issues related to the programming assignment on a consistent basis. Students questioned, directed, and guided each other throughout the lab session. When student pairs could not seem to answer questions on their own, they would ask the instructor; but the interaction with the instructor was usually very brief (less than five minutes). Overall the paired labs seemed to be much more efficient than the unpaired labs. Although there was a lot of dialogue during the class it never seemed chaotic, as students seemed to stay focused on their tasks. The paired labs provided structure to the communication between students, which made it more effective than the unpaired labs.

They also noted that a lot of students' questions to instructors were of a logistical rather than conceptual nature. On a very frequent basis, pairs resolved their own problems without the instructor's help. Overall, instructors spent very little time answering questions. Most instructor-student interactions seemed to take the form of extended discussions. Students would want to know how to apply what they were doing to another scenario. Hypothetical discussions of applications showed evidence of higher-level thinking processes that went beyond the scope of the programming assignment. They observed self-management of groups and the way they were often able to overcome their own programming difficulties.[60]

The main impact of pair programming in the CS1 course is seen in the lab[60]. The lab work is meant to reinforce lecture topics and the material found on the exams. Results indicate that student pair programmers are more self-sufficient in lab and are more likely to complete the class with a grade of C or better[60]. It is possible that pair programming in the labs provided enough of an attitudinal boost to motivate weaker students to continue working on passing the class when they might otherwise give up.

Closed labs are excellent for controlled use of pair programming [8]. The instructor can ensure that people are, indeed, working in pairs at one computer and ensure that the roles of driver and navigator are rotated periodically[40]. However, many courses have programming assignments that require development outside of a closed lab and it would be difficult to enforce pair programming in this case. It is the hope that a majority of students will come to fully appreciate the benefits of pair programming and will seek to work with a partner outside of class[60]. Others will choose to work alone, whether they prefer pair programming or not, often because they would prefer to work on homework in the comfort of their dorm room at any hour of the day or night when collocation may not be feasible. Educators have suggested the use of distributed software development packages[18, 21] that have tools that simulate human actions such as gesturing built in to overcome this concern. Pairing outside of lab requires effort in planning and coordination; some students view the added coordination as a hassle[60].

Chapter 3

3.0 Methodology

In the 2005-6 academic year, an experiment was conducted to investigate the benefits of self-selection and random pairing with pair rotation by monitoring students enrolled in the undergraduate introductory programming, CS1, courses taught at Norfolk State University (NSU). The introductory course used a breadth first approach to teach the C++ programming language. The course included typical introductory content: types, loops, conditionals, scope, arrays, searching, sorting, and static variables. The course also emphasized software engineering practices regarding testing, debugging, and documentation.

The courses were taught with either three 50-minute or two 75 minute lectures and one two-hour lab each week. The lecture course, CSC 170: Computer Programming I, and the lab course, CSC 170L: Computer Programming I Laboratory, were two separate courses; therefore students received two separate grades, one for the lecture course (CSC 170) and one for the lab course (CSC 170L). Each course will be addressed individually. Students attended lectures in groups of not more than 27 and labs in groups of not more than 40 with others from any lecture section.

Computer Programming I

The lecture course is a service course and is therefore taken by many students throughout the School of Science and Technology. Most students are from the computer science department and are either freshmen or sophomores. However, students of all undergraduate and graduate levels may take the course.

Student grades in the lecture course were based on three exams, one final exam, quizzes, lab assignments, a written assignment and programming projects that were completed outside of the closed lab and the lecture course class meetings. The programming projects are generative, that is the students start the project from scratch without any structure imposed by the instructor. Students were only allowed to complete the programming projects as a pair, all other assessments were measured individually.

Computer Programming I Laboratory

The closed labs were a new addition to the computer science department at NSU; therefore the lab was required for all newly entering freshmen CS majors only. Any other students, non-majors or CS majors repeating the course, were not required to also enroll in the closed labs due to their previously chosen curriculum and catalog year. However, their enrollment into the labs was highly encouraged.

The lab period was run as a closed lab where students were given a weekly assignment to complete during the allotted time. Lab assignments were “completion” assignments where students were required to fill in the body of methods in a skeleton of a program prepared and provided by the instructor.

Students’ grades in the lab course were based on the lab assignments. Twenty percent of each weekly lab assignment was based on the peer evaluation submitted by the pairing partner. There were no individual assignments in the lab course. All students were expected to use pair programming and complete all lab assignments in a pair.

All students participated in both self-selection and random pairing. The students were required to evaluate their experience with their partner at the conclusion of each programming assignment and at each rotation in the laboratory course.

Pair rotation was used to facilitate a number of reasons listed in Chapter 2, specifically to expose the students to more students currently enrolled in the class than they would normally meet and to give the students the comfort in knowing that they do not have to work with the same partner for the entire semester. An additional reason to implement was to aid in handling dysfunctional pairs. Peer evaluation was used to obtain multiple forms of feedback regarding each one of the pair. This feedback allowed the researcher to get data on a particular student's participation with each pair programming exercises.

The researcher decided to only use two methods of pair formation; self-selection and random pairing. Random pairing is commonly used in CS education. Self-selection is not as common at the CS1 level. Self-selection was used because the researcher believed that giving the student the option to choose their partner on occasion would aid in the satisfaction and enjoyment of the CS1 course and that it would not yield any adverse results. In fact, at the CS1 level, the researcher believed that the students do not really know the background or work ethic of other students in the class therefore the self-selection of a partner is effectively a random choice.

The researcher and two other independent faculty members, with no interest in the research, taught all sections of the course and laboratories. All faculty members are seasoned in the instruction of programming development courses and were trained to use pair programming at the CS1 level course. One teaching assistant for all sections of each course assisted with the grading of all assignments and therefore eliminated any bias of the researcher on completed assignments. Students were required to attain a grade of C or better to pass each class.

3.1 Sample

All students enrolled in the lecture and laboratory courses were observed for the study. Students may consist of any classification level and any major enrolled in the lecture course. The laboratory students were expected to be all entering freshmen computer science majors.

3.2 Instrumentation

Two instruments were employed for data collection in this research. They include the demographic data sheet and the peer evaluation form (PEF). Both instruments were designed by the author for this particular study. A description of both instruments follows.

3.2.1 Demographic Data Sheet

This form was designed and populated by the author. The information contained in the data sheet was obtained from the student information database at NSU. The form contains the following list of information for each student. *(see Appendix D)*

1. Scholastic Aptitude Test (SAT) math score
2. High school GPA
3. College GPA (if a transfer student)
4. Hometown City
5. Hometown State
6. Gender

7. Major

3.2.2 Peer Evaluation Form (PEF)

This form was designed to collect student reflections of their experience with pair programming and working with their partner. The PEF was based on the peer evaluation assessment tool developed and used by Dr. Laurie Williams in many of her research studies[67]. *(see Appendix B)*

The peer evaluation form was administered in class at the completion of every laboratory and programming assignment in each course. The survey consisted of questions related to the students' perceptions of pair programming, questions for evaluating their partner in relation to their activity with pair programming, and free-response questions to provide anecdotal statements regarding the students' experience. Students were asked for their name so that a classification and gender relationship could be established. Students ranked the following statements with strongly disagree, disagree, neutral, agree, and strongly agree:

- (Q1) My partner made a serious effort to complete the assigned work.
- (Q2) My partner contributed to team efforts.
- (Q3) My partner was technically capable of completing this week's lab exercise.
- (Q4) My partner and I were compatible when we worked together on this exercise.
- (Q5) My partner helped me to complete this exercise.
- (Q6) I enjoyed working with my partner on this exercise.
- (Q7) My partner improved my ability to complete this exercise.
- (Q8) The quality of this programming exercise was improved because my partner and I worked together.
- (Q9) My partner treated me with respect while working on this exercise.
- (Q10) I was more confident completing this exercise because my partner worked

with me.

All other questions were free-response questions. Students were asked if they selected their partner and if not, would they have rather had the option to select their partner and why. Students were also queried on how long they have known their partner and to list other courses that both were currently enrolled in during that particular semester. The final 3 free response questions on the PEF asked did they enjoy the pair programming experience with this partner, what was the partner's most significant contribution and was most irritating about working with this partner.

3.3 Procedure

Participants were observed in their classes, where all instructors agreed to administer the research plan *(see Appendix A)* and surveys. Participants were not told the specific objective of this research but were advised by a statement included on the syllabus that they were participating in a study aimed at increasing student retention. A statement was also read aloud to the class that informed the students of their right to not participate without penalty although their participation was highly encouraged.

The lecture course was taught as it had been in many years but now included the addition of several key techniques. Pair programming was implemented for the software development of all programming projects. Peer evaluation was used to assure faculty members that all students were contributing to the development of the programming assignments. Pair rotation was employed to allow students to meet more students in the classroom and to decline the chances of pairing with a parasitic partner. Pair formation consisted of either random pairing or self-selected pairing. Random pairs were formed based on computer generated software. The software assigned a number to each student enrolled in the class and then assigned pairs based on those numbers. Students were not allowed to pair with someone that they had previously paired. All students in both the lecture and the laboratory courses were expected to participate in pair programming, pair rotation, peer evaluation, and prescribed methods of pair formation.

All students were given a mini-lecture on pair programming. The students were also placed in pairs to solve a simple problem during a regularly scheduled lecture meeting. This allowed the students to get a sense of what pair programming is before having to complete assignments outside of the classroom.

In the lecture course, the students were required to complete six programming assignments administered during the semester in addition to other instruments, quizzes and tests, used to assess the individual learning of the course material (which is outside the scope of this research). There is a plethora of documented research that suggests that pair programming does not hinder individual student learning [32, 33, 35, 40, 61, 64, 65, 67].

Specifically, five pair rotations occurred during the semester in the lecture course and weekly (13) pair rotations occurred in the lab. All students were not permitted to pair with a previous partner within the same class unless it could not be avoided in the laboratory course due to low enrollment. Students completed peer evaluations at the conclusion of each programming or laboratory assignment and then rotated to a new partner.

All programming assignments were graded by each instructor or TA based on the requirements of the program. The peer evaluation forms (PEF) submitted by each student's partner comprised 20% of the final grade for each programming or laboratory assignment. The PEFs were administered at the conclusion of each programming assignment and every lab assignment. The students rated their partner's contribution to the pair programming activity. Specifically, students answered all questions located on the peer evaluation form, however only a subset of the questions was used to measure the partner's performance. The following table shows the questions the students answered to determine the partner evaluation points given to their partner.

Table 1: Questions used to Evaluate Partner

	Strongly Disagree	Disagree	Neither agree nor disagree	Agree	Strongly Agree
1. My partner made a serious effort to complete the assigned work.					
2. My partner contributed to team efforts.					
3. My partner was technically capable of completing this week's lab exercise.					
4. My partner helped me to complete this exercise.					
5. My partner treated me with respect while working on this exercise.					

The answers were scored from 0 (Strongly Disagree) to 4 (Strongly Agree). The maximum score that a student could receive from their partner was a score of 20 points received from the peer evaluation. The remaining 80 points were obtained from the actual merit and implementation of the programming solution. Those assignments were graded as a pair, in that both students of the pair received the same score for the eighty percent of the programming solution. The other 20% allowed students within a pair to have varying final scores for each programming assignment.

Only programming and lab assignments were graded as a pair. All other assignments, tests, quizzes, and final exam, were tested individually. These grades measured the individual learning of each student. In addition, common questions within the quizzes and exams were distributed to all sections to assess common material within all sections. A common final exam was given to all students taking the CS1 lecture programming course.

In the lab course, students participated in weekly completion assignments and worked as a pair using pair programming to complete the assignments. Each assignment's grade was composed of 80% merited toward the actual programming development solution and 20% awarded based on the peer evaluation. There were no other assignments that contributed to the final grade awarded in the lab course and no assignments that measured the individual learning of the students. That aspect seemed to be unnecessary since all students enrolled in the lab course were also enrolled in the lecture course and there were many instruments used to measure individual student learning in the lecture course.

Students in the lab environment were paired two different ways. During all even weeks the lab was in session, students were randomly assigned a partner by the teaching assistant (TA) at the beginning of the lab section using a random pair generator. If one of the pair did not attend, the partner was reassigned to another single partner or a pair was made into a triple, no one worked alone. During odd weeks, students had the opportunity to self-select their partners – as long as it was not someone they had worked with previously. Students dissatisfied with their pair were not reassigned. Careful preparation was made to ensure that the lab assignments given to the students would meet (not exceed) the two-hour time requirement, and that there was enough work for two to three students to complete. At the conclusion of each lab, all students were required to fill out a peer evaluation form in order to; (1) ensure one peer is not completing more of the assignment than the other, (2) have students learn the evaluation process early, and (3) to create a paper trail of measurements in the form of student responses about how effective their pair experiences were as a first time programmer.

Students rated their experience based on enjoyment while working in a pair to complete the assignment and whether they would rather self-select their partner and if they found a difference in working with a partner they self-selected versus a partner assigned to work with them.

3.3.1Baseline data

The class results were measured against students that were enrolled in the same course during the fall 2004 semester. During that semester, no students participated in pair programming therefore all items used to measure performance were tested individually. The math score for the Scholastic Aptitude Test (SAT-M), and high school grade point average (HS-GPA, required of all new freshmen for entrance into the university) or college grade point average, (C-GPA, required of all transfer students) were used to determine if the students enrolled in both semesters were students with equivalent aptitude and academic preparation to succeed in computer science. These metrics were used in a similar study completed at North Carolina State University to show that students were not uniformly distributed across all groups [17]. The mean SAT-M for the students enrolled during the Fall 2004 semester was 461.67 with a standard deviation of 85.46 while the Spring 2006 students has a mean score of 463 with a 77.67 SD. The mean GPA for the students enrolled during the Fall 2004 semester was 2.48 with a standard deviation of 0.48 while the Spring 2006 students had a mean score of 2.537 with a 0.51 SD The analysis on the SAT-M and the GPA using the One-Way ANOVA showed that the students enrolled during the Fall 2004 semester were intellectually comparable to the students enrolled during the Spring 2006 semester. The differences are not statistically significant.

Table 2: Students' SAT-M and GPA Information

	SAT-M	**Std. Deviation**	**GPA**	**Std. Deviation**
Fall 2004	461.67	85.46	2.48	0.48
Spring 2006	463	77.67	2.537	0.51

3.4 Hypotheses

To investigate the effects of self-selection pair formation on CS1 students and its relationships to pair programming and retention, the researcher developed three hypotheses that follow along with its' rationale

Hypothesis 1: *Satisfaction for pair programming is higher in CS1 when pairs self-select their partners.*

Alternative: *Satisfaction for pair programming is not increased when students self-select their partner.*

Rationale: This hypothesis was developed to investigate satisfaction when CS1 students select their partners.

Resolution: To investigate H1, **an** ANOVA was used to determine when students self-select their partners by selecting yes to Q12, "Did you select your partner?", do more students answer the questions, "Q6: I enjoyed working with my partner on this exercise.", and "Q7: My partner improved my ability to complete this exercise" at higher rates than when students partners are assigned.

Comments to questions "Q14: Did you enjoy the pair programming experience with this partner?" were coded into positive (1) and negative (0) responses. Those values were evaluated and compared based on the individual response to "Q12, Did you select your partner?" Responses to the questions "Q15: What was the partner's most significant contribution?", and "Q16: What was the most irritating about working with this partner?" were used to supply anecdotal data.

Hypothesis 2: *Self-selection pairs perform as well as or better than randomly assigned pairs in CS1 on individual assignments.*

Alternative: *Self-selection pairs do not perform as well as or better than randomly assigned pairs in CS1 on individual assignments.*

Rationale: This hypothesis was developed to show if there is a relationship between self-selection pairs and individual assignments. The research intends to prove that allowing a student to self-select their partner does not hinder a student's performance on individual assignments.

Resolution: To investigate H2, an ANCOVA will be used to determine if students self-select their partners by answering yes to Q12, "Did you select your partner?," do more students receive equivalent scores on individual assignments.

A Chi-Square test will be used to determine the statistical difference in success rates on exam grades when a student participated in self-selection pair formation and random pair formation, when answering yes to Q12, “Did you select your partner?”

Hypothesis 3: *Pair programming in CS1 increases retention of CS majors.*

Alternative: *Pair programming does not increase the retention of CS majors.*

Rationale: This hypothesis was developed to explore a relationship between pair programming and CS major retention.

Resolution: Retention will be analyzed by comparing all computer science majors enrolled in the lecture course in fall 2004 (baseline data), fall 2005 (1st implementation of research measures) and spring 2006. The students of each semester will be categorized by major, gender and successful completion of the first CS course. Students for each semester will be identified first by successful completion. The researcher will assume that a student who does not pass the class will interpret the failure as a negative experience and therefore will be more likely to change their major therefore the first goal of increasing the retention rate is to increase the passing rate. The students will then be identified by major. The researcher’s goals are to retain students who have identified computer science as their major. Genders of these students will also be analyzed to identify if the research measures improved the female retention rate.

The CS majors of fall 04 will be divided into 3 categories and each category will then be divided by gender to reveal retention statistics by gender. The first category (1) will contain students who successfully passed the course and enrolled in the next course during the next semester; these students will also be identified if they remain a CS major one-year later (this is especially important because the highest attrition occurs at the conclusion of the first year of study.) The second category (2) will contain students who do not successfully pass the first course, but remain in the major by repeating the course in the ensuing semester. The percentage of these students will also be analyzed to investigate if they remained a CS major one-year later. Both categories one and two will be considered retained students. The third category (3) will identify students that change their major at the conclusion of the first semester or at the end of the first year regardless of completion of the course. These students will be considered to be non-retained at each intersession.

All of the categories will yield percentages to which fall 2005 and spring 2006 data can be compared. Retention statistics will be reported in percentages relative to the original data at the beginning of the fall 2005 semester since it is the first semester with implemented measures to increase retention. A Chi-square test will be used to report statistical significance.

3.5 Data Analysis

All analyses will be conducted via SPSS Statistical software and Microsoft Excel. All statistical measurements will be analyzed at the 95% confidence level. The following list identifies all preliminary statistics used to show that the students are of equal caliber. All other statistics used are identified in the Hypotheses section of this chapter.

- The mean SAT-M and HSGPA for 2 semesters, F04, &, S06. These tests were used to identify that the students are of equal caliber.
- A one way ANOVA (T-test) on SAT-M and HSGPA was used to further show that the students are comparable to the students enrolled in Fall 04.

- Final grades were used to determine passing percentages and were compared from fall 2004 to spring 2006. The Chi-Square test was used to confirm previous studies that pair programming increases learning and that the statistical difference is significant.
- Questions 1 through 10 on the PEF were coded Strongly Disagree (1) to Strongly Agree (5) so that quantitative statistics could be performed and compared to basic percentages.

Chapter 4

4.0 Observations

A formal experiment was implemented at Norfolk State University (NSU) to investigate the methods to form pairs at the CS1 level that influence satisfaction within the course and to validate the effectiveness of pair rotation and its relationship to pair formation in CS1. In the fall of 2005, introductory programming was taught to 78 students who were mainly freshmen and sophomore undergraduate students of various science majors. A subset of the group, 40 freshmen CS majors, was also enrolled in an introductory programming laboratory course. The students formed pairs by self-selecting their partners or by having their partner randomly assigned to them to complete the majority of the programming assignments. The purpose of the class was to pilot the research plan before implementing the formal experiment.

4.1 Pilot Study

The students enrolled in the lab course switched partners weekly. On the odd week, the pairs were formed by self-selection and on the even week, the pairs were formed randomly. The students completed a paper peer evaluation each week and were told that the rating given to them by their partner comprised 20% of their weekly lab score as well as their programming assignment grade in the lecture course. There were many problems that arose.

The first problem was that first-year students are inexperienced in evaluation techniques. Peer evaluation was initially used in fall 2005 to facilitate individual accountability in paired programming assignments; however, studies suggest that "peer evaluation is not a good tool for motivating students"[10, 51]. When dealing with students in their 1st CS course, motivation must be considered since most students are unsure of their academic major election. Attrition in CS is a nationwide problem, so the researcher decided to reduce as many antagonistic points as possible.

Furthermore, comments from students during the fall semester suggested that students were not evaluating their partners based on their performance because they did not want to be responsible for decreasing their partner's final score. Students also asked the researcher, "How did I get a lower grade than my partner?" When told it was due to the peer evaluation, the students would then mentally recount the activities and suggest they did more work than their partner. At the end of the fall 2005 semester, the evaluations were reviewed and further supported the students' comments. Many students elected to give the highest scores because they did not want to "bring anyone else down." This caused the data to be extremely skewed and did not represent whether or not the student actually participated in the pair programming process.

Analysis of the data from fall 2005 showed most students were evaluated with the highest scores. Only in the case where a student did absolutely nothing, did the partner evaluate the student with low marks. Table 3 shows the percentages of answers provided by the students during the fall 2005 semester.

Table 3: Response Distribution for Fall 2005

	Strongly Disagree	Disagree	Neither agree nor disagree	Agree	Strongly Agree
1. My partner made a serious effort to complete the assigned work.	0.73%	0.36%	1.10%	13.6%	84.1%
2. My partner contributed to team efforts.	0.73%	0.36%	1.47%	11.3%	86%
3. My partner was technically capable of completing this week's lab exercise.	0.73%	0.36%	1.83%	13.2%	83.8%
4. My partner and I were compatible when we worked together on this exercise.	0.73%	0.36%	1.83%	11.3%	85.6%
5. My partner helped me to complete this exercise.	0.36%	0.73%	1.83%	11.3%	85.6%
6. I enjoyed working with my partner on this exercise.	0.36%	0.73%	3.30%	11.7%	83.8%
7. My partner improved my ability to complete this exercise.	0.36%	0.73%	2.94%	13.6%	82.3%
8. The quality of this programming exercise was improved because my partner and I worked together.	0.36%	0.73%	2.20%	11.7%	84.9%
9. My partner treated me with respect while working on this exercise.	0.36%	0.73%	1.10%	10.6%	87.2%
10. I was more confident completing this exercise because my partner worked with me.	0.73%	0.36%	3.30%	12.8%	82.7%

This phenomenon suggested a need for change to obtain accurate data. During the spring 2006 semester, the researcher decided to use the peer evaluation as a means to evaluate the students' experience with pair programming with a particular partner working on one assignment for 2-3 weeks in the lecture course or with multiple assignments for 2-3 weeks in the lab course. The students were advised to give their honest evaluation of their experience and that their responses would not influence the partner's grade. Students were ***not*** asked to evaluate their partner but to evaluate their experience with pair programming. Teaching assistants were used to evaluate the students' performance in the laboratory course. The online PEF responses for spring 2006 suggested that this particular change was very beneficial. Analysis of the responses showed a better distribution across all evaluation categories and suggests that the problem was corrected in the spring 2006 semester.

The second problem that contributed to the skewed data was the paper surveys. Since the surveys were filled in by hand and expected to be given to the instructor before leaving the classroom, some students would wait to see what their partner would give them and then assess their partner with the same scores. Students did not give their honest assessments for fear of retribution by their partner. This problem was corrected by building the survey into an online system. The students were then required to complete the online survey within one week of completion of the assignment. The students could then wait to complete the survey at their convenience at any computer with an internet connection. It appeared that students gave more honest reflections of their experiences because their partners were not around.

The third problem was the frequency in which the survey was administered. The survey was administered every week in the lab course and five times during the semester in the lecture course. Therefore a student enrolled in the lecture and lab course was exposed to the survey as many as 17 times within the semester. This exposure was too often for these students. By the mid-term advisory period, students began to just check the items without reading the questions. This problem was corrected in the lab course by having the student complete the online survey after working with a particular partner for 3-4 weeks. This reduced the exposure of the survey for a student enrolled in the lecture and lab course to a maximum of 9 times within the semester.

Another problem was the frequency in the partner rotation in the lab course. During the fall semester, students rotated partners weekly in the lab course. The students knew that they only had to work with a particular partner for a maximum of 2 hours per week therefore whether they selected their partner or the partner was assigned to them had no influence on their experience. This problem was addressed by requiring lab students to work with a single partner for 3-4 weeks and complete 3-4 assignments, and then evaluate their experience with pair programming as it related to working with a single partner on multiple assignments.

The final problem that was addressed was whether or not the students were actually performing true pair programming as defined by Extreme Programming in the lecture course. This problem manifested in many ways, some faculty members had to be convinced by the researcher that pair programming is not one person working on partial code and sending it to their partner via email and in essence that no code should be changed without both partners being present. To address this problem, a question was added to the survey to determine the actual work methodology employed by the students. Students could select one of 4 choices; whether they worked alone, programmed alone and shared code via email or the equivalent, programmed as a pair by sharing one computer for the entire assignment, or other, in this case the student could describe how the pair completed the assignments. The students were told to complete the survey accurately and that there would be no retribution grade wise. Spring 2006 faculty members teaching the CS1 courses were provided training on the practice of pair programming and were told to highly encourage the true practice for all assignments. Real Virtual Networking Computer (VNC) was also introduced to the students in the researcher’s class to aid in the collocation requirement of pair programming.

4.2 Official Experiment

The official experiment was implemented in the spring of 2006. The students consisted of 52 freshmen and sophomore undergraduate students of various science majors enrolled in two sections of the CS1 course and a subset of 25 freshmen CS majors enrolled in two sections of the introductory programming lab course that accompanies the lecture course. All of the students were considered to be novice programmers and for most students these courses were their first introduction to program development using a high level programming language. In the lecture course, the students completed six programming assignments during the semester. The students selected their partner for assignments one and four, were assigned partners for assignments two and five, completed assignment six individually, and were given the choice to self-select their partner or work individually to complete assignment three.

In the lab course, the students completed a new lab assignment weekly therefore completing fourteen lab assignments during the semester. The students were allowed to self-select and work with the same partner for three weeks. At the conclusion of the three weeks, a new partner was then assigned to the student and the pair worked together for the next three week period (completing three lab assignments during the three week period). The students alternated self-selection and assigned pairing throughout the entire semester effectively completing four rotations during the semester. At each rotation, the student obtained a partner that they had not previously worked with.

The lecture and lab courses were structured quite differently. The pairs in the lecture class worked together on one assignment for a period of two-three weeks. The pairs in the lab course worked together on three different assignments during their 2-3 week timeframe. It is important to note that when determining whether a pair had worked together previously, the students were only bound by the particular class. For example, if a student had worked with a particular partner in the lecture course, that did not exclude the same pair from forming in the lab course.

4.2.1 Sample

Two sub-groups of undergraduate students were observed for the study. The first group represented 28 students enrolled in section 01 (LEC1) and 24 students enrolled in section 02 (LEC2) of the CS1 programming lecture course. The second group represented 16 students enrolled in laboratory section 01 (LAB1) and 9 students enrolled in laboratory section 02 (LAB2). The researcher only taught LEC2, all other courses were taught by the independent faculty members. The fifty-two students represented 37 males, 15 females, 27 CS majors, 25 non-majors, 18 male CS majors and 9 female CS majors.

4.2.2 New Data Collection Instrument (Spring 2006): Online Peer Evaluation Form (OPEF)

The Online Peer Evaluation Form was the peer evaluation form built into an online database system. The OPEF consisted of the same questions on the PEF but also included an additional question on the work methodology of the pair. *(See Appendix C.)*

Students used their NSU email addresses and ID numbers as their username/password combination when submitting online survey responses. The surveys could be completed from any computer with internet access. The participants were asked to complete the OPEF within one week of completing the assignment. In most cases, completion of the survey occurred within one week of the assignment; however, one of the faculty members allowed the students lecture class time to complete the survey once for those students who had not completed the survey. Individual surveys could not be edited by anyone once they were completed. The researcher and the students were able to view only the completed surveys once submitted.

All programming assignments were graded by each instructor or TA based on the requirements of the program. The OPEF surveys were administered at the conclusion of each programming assignment and every third lab assignment where the students rated their experience with pair programming. Specifically, students answered all questions located on the online peer evaluation form. Students rated their experience based on enjoyment while working in a pair to complete the assignment and whether they would rather self-select their partner and if they found a difference in working with a partner they self-selected versus a partner assigned to work with them.

4.3 Achievement

Achievement was measured by the number of students that successfully completed the course with a grade of C or better versus those that received a D, F, or withdrew during the semester. The study found that during the Fall 2005 and Spring 2006 semesters where all students participated in pair programming, 66% and 58%, respectively, of the students successfully completed the course with a grade of C or better. Comparatively, during the fall 2004 semester where no students participated in pair programming, only 42% of the students successfully completed the course. A Chi-Square test comparing fall 2004 to fall 2005 revealed that the difference in achievement rates is statistically significant, p =.048. This result is consistent with a similar study completed at North Carolina State University [17] which concludes that pair programming improves a student learning to program in a CS1 level course. The addition of pair programming shows that it helps students pass the class with a grade of C or better regardless of the method of pair formation.

The Chi-Square comparing fall 2004 (baseline data) to spring 06 does not reveal a statistically significant increase, p= .327, which is consistent with a similar study at the University of California at Santa Cruz [34, 35] that measured the passing rates of pairers and non-pairers that actually took the final exam. Fall 2005 students consisted of 63% computer science majors therefore the researcher suggests that those students have a different motivation for passing than the students that have not chosen computer science as their academic discipline. Table 4 shows the mean GPA, SAT-M, percentages of majors, pass rates, and probability of the Chi-Square test for each semester.

Table 4: Achievement Factors by Semester

	GPA	SAT-M	CS Major Percentage	Pass Rate	Chi-Square results (p)
Fall 2004 (baseline data)	2.527	460	29%	48%	
Fall 2005	2.579	453	63%	66%	.048
Spring 2006	2.518	463	52%	58%	.327

4.4 Results

The data was collected and analyzed in many ways. When observing all of the data, regardless of pair formation methodology, the researcher found that the students overall had positive experiences with the methodology implemented. Table 5 shows the students' OPEF responses to questions 1 through 10 for spring 2006 in percentages. The normalized distribution of student responses suggests that the students provided more honest reflections of their pair programming experience.

Table 5: Response Distribution for Spring 2006

	Strongly Disagree	Disagree	Neither agree nor disagree	Agree	Strongly Agree
1. My partner made a serious effort to complete the assigned work.	6.25%	4.68%	15.6%	36.7%	36.7%
2. My partner contributed to team efforts.	5.42%	5.42%	13.9%	37.2%	37.9%
3. My partner was technically capable of completing this week's lab exercise.	4.68%	3.90%	18.7%	39.8%	32.8%
4. My partner and I were compatible when we worked together on this exercise.	4.68%	3.90%	15.6%	42.9%	32.8%
5. My partner helped me to complete this exercise.	4.65%	4.65%	17.8%	38.7%	34.1%
6. I enjoyed working with my partner on this exercise.	7.03%	5.46%	17.1%	36.7%	33.5%
7. My partner improved my ability to complete this exercise.	6.92%	6.92%	18.4%	33.0%	34.6%
8. The quality of this programming exercise was improved because my partner and I worked together.	6.97%	6.20%	17.0%	37.9%	31.7%
9. My partner treated me with respect while working on this exercise.	3.10%	1.55%	17.0%	43.4%	34.8%
10. I was more confident completing this exercise because my partner worked with me.	6.97%	5.42%	18.6%	34.1%	34.8%

These responses suggest that the majority of CS1 students are satisfied with their experience when placed in an academic environment that uses pair programming, pair evaluation, pair rotation, and two methods of pair formation.

4.5 Hypotheses Analysis

The main objective of the research is to prove that self-selection pairing is a beneficial pedagogical technique that increases the satisfaction of CS1 students that ultimately leads to greater retention of CS majors. To investigate the objective several hypotheses were explored in part:

Hypothesis One: *Satisfaction for pair programming is higher in CS1 when pairs self-select their partners.*

Alternative: *Satisfaction for pair programming is not increased when students self-select their partner.*

Resolution: All questions were coded one (1) Strongly Disagree to five (5) Strongly Agree so that quantitative analysis could be performed. An ANOVA was used to determine when students self-select their partners by selecting yes to Q12, "Did you select your partner?", do more students answer the questions, "Q6: I enjoyed working with my partner on this exercise.", and "Q7: My partner improved my ability to complete this exercise" at higher rates than when students partners are assigned.

In the lab course for Q6, the mean rating students gave when self-selecting (answering yes to Q12) their partner was very high, 4.800 and the mean rating when the partners were assigned randomly was 4.042. In the lecture course for Q6, the mean rating students gave when self-selecting was 4.111 and was 3.500 when the partners were assigned. Overall for both the lab and lecture, the average for Q6 was 4.226 when the pairs used self-selection to implement pair formation and 3.688 when the pairs were randomly assigned. Charts 2 and 3 display the graphical representations of the mean scores.

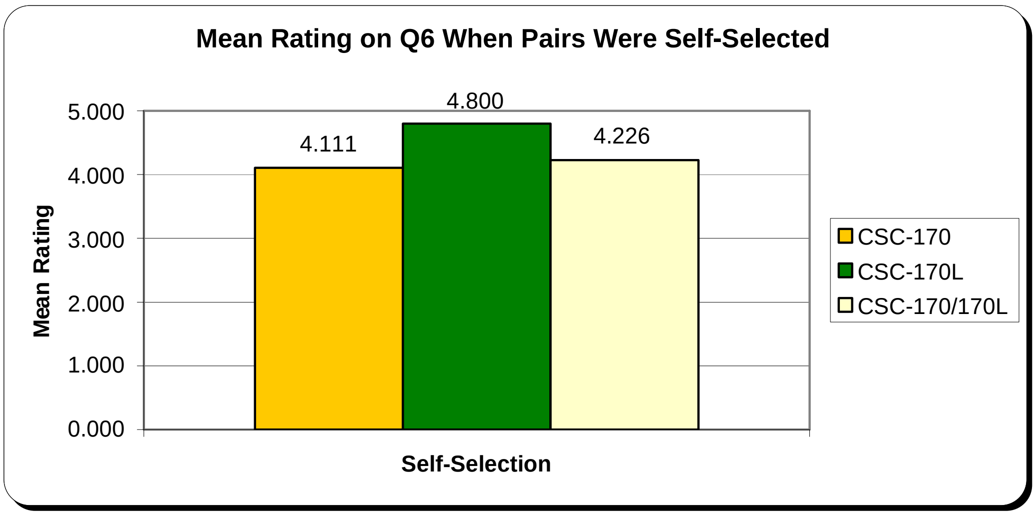


Chart 1 : Mean Self-selection Ratings to Q6

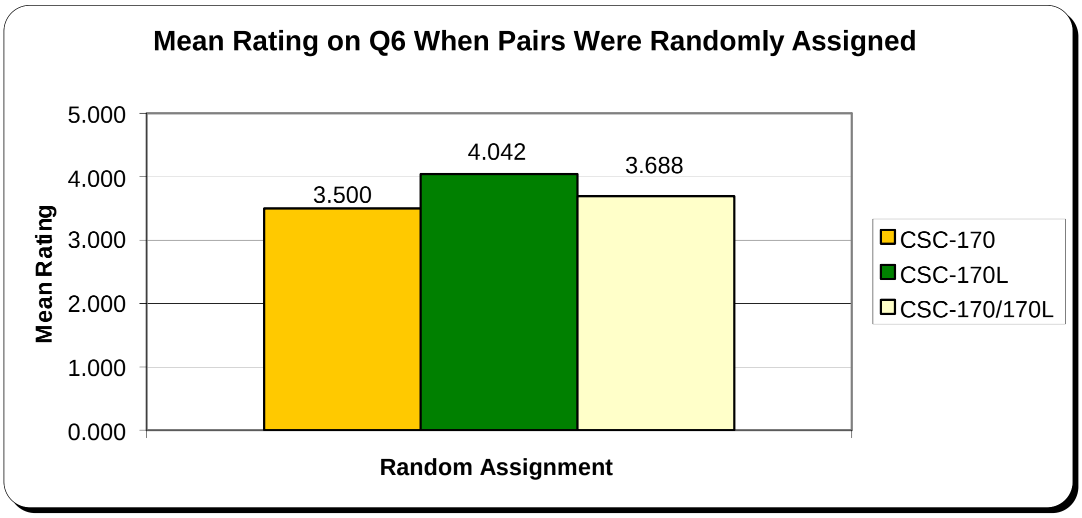


Chart 2 : Mean Randomly Assigned Ratings to Q6

The ANOVA for Q6, $p < 0.05$, shows that the increase is statistically significant for all comparisons.

In the lab course for Q7, the mean rating students gave when self-selecting their partner was also high, 4.600, and the mean rating when the partners were assigned randomly was 3.917. In the lecture course for Q7, the mean rating students gave when self-selecting was 4.000 and was 3.519 when the partners were assigned. The combined lab and lecture self-selection paired average for Q7 was 4.113 and was 3.663 when the pairs were randomly assigned. Charts 3 and 4 show the graphical representations of the mean scores.

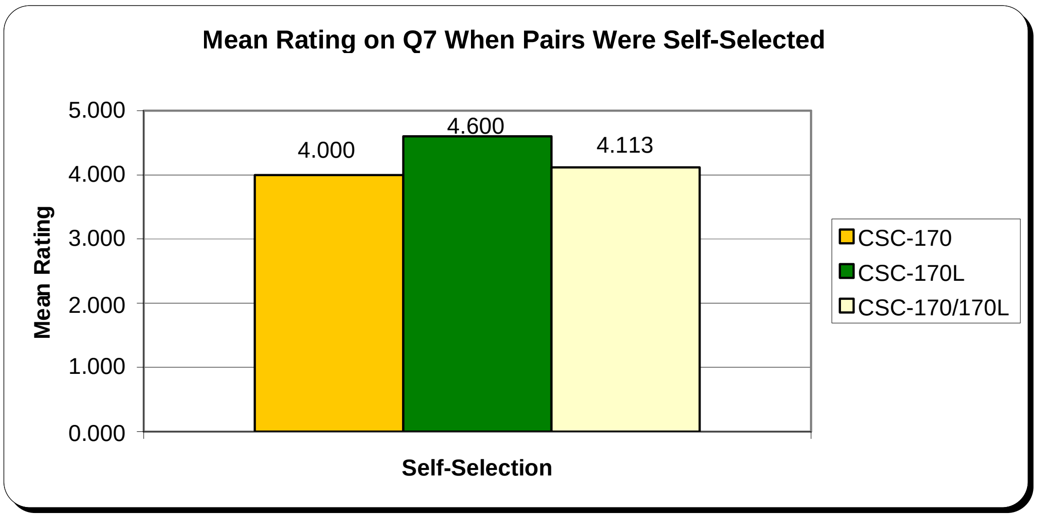


Chart 3: Self-Selection Mean Rating to Q7

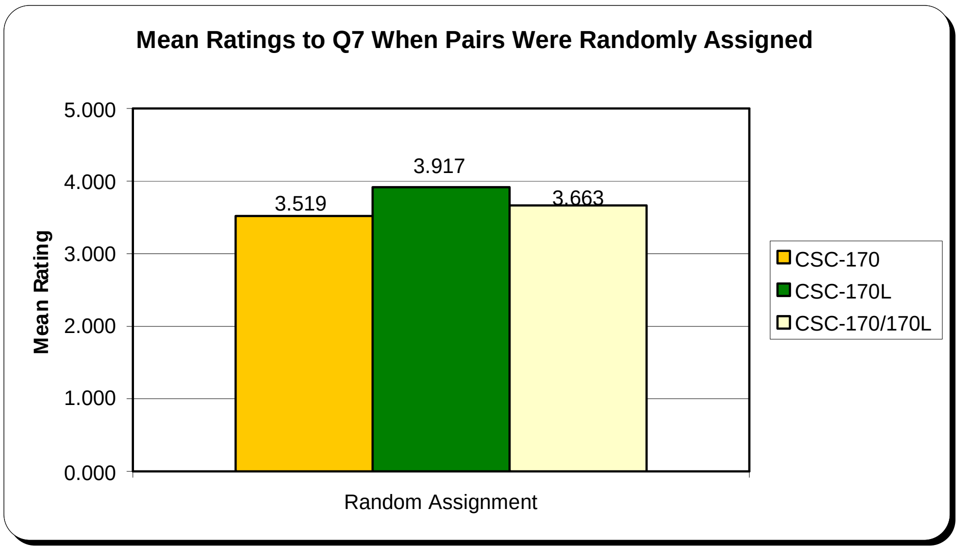


Chart 4: Random Mean Rating to Q7

The ANOVA for Q7, $p < 0.05$, shows that the increase is statistically significant across all comparisons.

Comments to questions "Q14: Did you enjoy the pair programming experience with this partner?" were coded into positive (1) and negative (0) responses. Those values were evaluated and compared based on the individual response to "Q12, Did you select your partner?" Responses to the questions "Q15: What was the partner's most significant contribution?", and "Q16: What was the most irritating about working with this partner?" were used to supply anecdotal data.

In response to Q14:"Did you enjoy the pair programming experience with this partner? the following results were discovered; students overall had positive experiences whether their partner was self-selected or the partner was assigned to them. In CSC170, when the student self-selected their partner, 91% of the student responses were positive compared to 69% positive responses when the pairs were assigned randomly. This represents a 22% increase in the enjoyment of the student when they self-select their partner compared to having the partner assigned to them. In CSC170L, the lab course, the difference is not as large but there is still an increase. When the students self-selected their partner, 100% of the students enjoyed the programming experience, comparatively when the partners were assigned, 89% of the students' responses were positive.

Table 6 : Positive Student Responses to Q14

	CSC 170	*CSC170L*
Randomly Assigned	69%	89%
Selection	91%	100%
% Increase of Self-Selection Pairing Over Randomly Assigned Pairing	+22%	+11%

The data suggests that students overwhelmingly favored self-selecting their partner (92%) when compared to having the partner assigned to them (76%) when satisfaction is a goal.

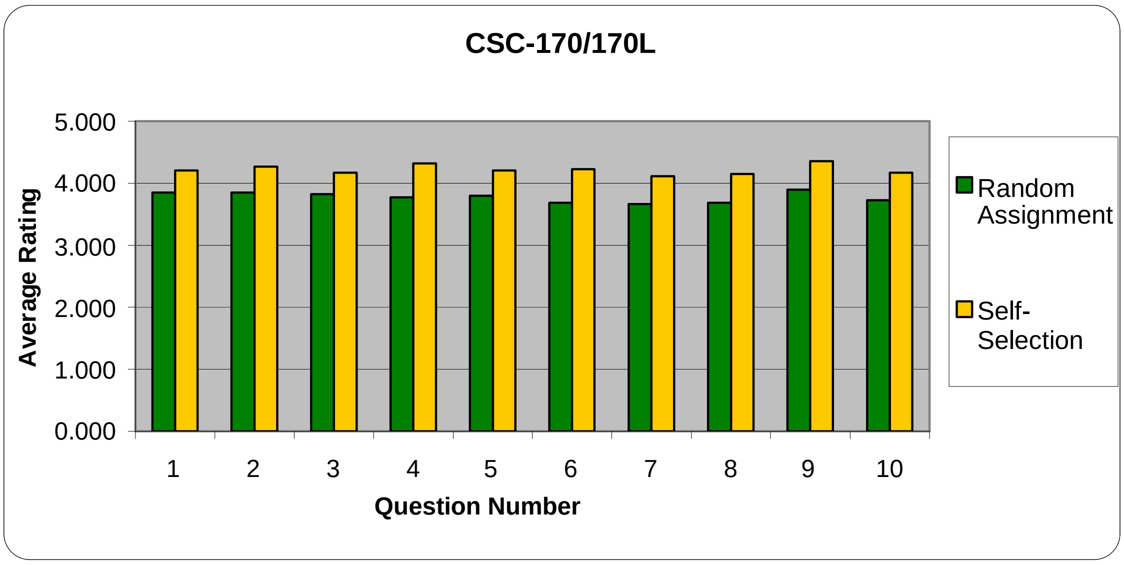


Chart 5: Mean Ratings by Question (All Courses)

Chart 5 displays that students rated all questions higher (mean score) when self-selecting their partner over having their partner assigned to them randomly. The data shows that students preferred to select their partner. Anecdotal evidence lists the following student responses as reasons to self-select their partner, “the partner that I would have picked we would have done a better job, and more effort would have been put into the project,” and “more comfortable with my last partner because I knew him longer.”

Based on these findings, the research supports the hypotheses that satisfaction for pair programming is higher in CS1 when pairs self-select their partners.

Hypothesis 2: *Self-selection pairs perform as well as or better than randomly assigned pairs in CS1 on individual assignments.*

Alternative: *Self-selection pairs do not perform as well as or better than randomly assigned pairs in CS1 on individual assignments.*

Resolution: An ANCOVA was used to determine if when students self-select their partners by answering yes to Q12, “Did you select your partner?,” do more students receive equivalent scores on individual assignments.

A Chi-Square test was used to determine the statistical difference in success rates on exam grades when a student participated in self-selection pair formation and random pair formation, when answering yes to Q12, Did you select your partner?

This hypothesis was only tested in the CSC170 lecture course since no individual assessments were conducted in the lab course, CSC170L. Three exams and the final exam were used to measure how students performed individually. The exams were administered during weeks 4, 8, 12, and 16. Exam 1 and the final exam were common across all sections. The final exam was the only exam considered to be fully comprehensive however each exam was incremental in nature, and students were expected to know any information that had been covered up to the exam date including material tested on previous exams. Students participated in self-selection pair formation leading up to exams 1 and 3 and random pairing prior to exam 2 and the final exam. Any grade of zero on an exam is not calculated in the average, the researcher assumes that any student that actually took the exam would score above a numerical score of 0.

On average, when students participated in self-selection pairing they performed better on their exams than on the exams taken after participating in random pairing, as shown in Table 7.

Table 7: Performance on Exam Scores by Pair Formation Method

Exam	Pairing Method		Mean	Standard Deviation
	Self-Selection	Random		
Exam 1	X		66.66	24.77
Exam 2		X	54.272	26.302
Exam 3	X		57.28	30.12
Final Exam		X	53.7	18.6

In addition, when students were given the choice to self-select their partner or have a partner assigned to them for programming assignment three, 76% of the students elected to self-select their partner.

These exam averages show that students performed highest on the exams when they participated in self-selection. The Chi-Square test, p = .0495, suggests the difference in the number of students who pass the exams (receive a score of 70 or higher) is statistically significant when the students participate in self-selection pairing.

After evaluating student success by gender on individual assessments (intermediate exams and the final exam) regardless of partner selection and overall class success; the t-test shows that there is no significant difference in any of the variables. Group statistics are shown in Table 8. The results of the t-test are shown in Table 9.

Table 8: Group Statistics by Gender

Pair Formation	Assessment	Student's Gender	N	Mean	Std. Deviation	Std. Error Mean
Self-Selection	**Exam1**	**Male**	37	63.11	25.583	4.206
		Female	15	66.53	32.656	8.432
Assigned	**Exam2**	**Male**	37	46.770	30.9230	5.0837
		Female	15	51.067	29.4515	7.6043
Self-Selection	**Exam3**	**Male**	37	43.24	35.812	5.888
		Female	15	46.07	37.240	9.615
Assigned	**Final Exam**	**Male**	37	43.84	25.967	4.269
		Female	15	45.87	28.943	7.473
Pass		**Male**	37	.57	.502	.083
		Female	15	.60	.507	.131

Table 9: Independent Samples Test by Gender

Partner Formation	Assessment		Levene's Test for Equality of Variances		t-test for Equality of Means						
			F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
										Lower	Upper
Self Selection	Exam1	Equal variances assumed	.250	.619	-.403	50	**.688**	-3.425	8.493	-20.484	13.633
		Equal variances not assumed			-.364	21.320	**.720**	-3.425	9.422	-23.002	16.152
Assigned	Exam2	Equal variances assumed	.000	.997	-.460	50	**.648**	-4.2964	9.3414	-23.0592	14.4664
		Equal variances not assumed			-.470	27.198	**.642**	-4.2964	9.1471	-23.0584	14.4656
Self Selection	Exam3	Equal variances assumed	.022	.882	-.255	50	**.800**	-2.823	11.086	-25.090	19.443
		Equal variances not assumed			-.250	25.094	**.804**	-2.823	11.275	-26.039	20.392
Assigned	Final Exam	Equal variances assumed	.139	.711	-.247	50	**.806**	-2.029	8.214	-18.527	14.469
		Equal variances not assumed			-.236	23.649	**.816**	-2.029	8.606	-19.806	15.748
Course Success		Equal variances assumed	.203	.654	-.210	50	**.834**	-.032	.154	-.342	.277
		Equal variances not assumed			-.210	25.765	**.836**	-.032	.155	-.351	.286

A One-way ANOVA was used at each assessment, holding the gender variable constant, revealed there were no significant differences when student self-selected their partner vs having the partner assigned as shown in Table 10.

Table 10: One-way Analysis of Variance by Gender

Pair Formation	**Assessment**		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Self Selection	**Exam 1**	**Between Groups**	125.218	1	125.218	.163	**.688**
		Within Groups	38491.301	50	769.826		
		Total	38616.519	51			
Assigned	**Exam 2**	**Between Groups**	197.015	1	197.015	.212	**.648**
		Within Groups	46567.731	50	931.355		
		Total	46764.745	51			
Self Selection	**Exam 3**	**Between Groups**	85.083	1	85.083	.065	**.800**
		Within Groups	65585.744	50	1311.715		
		Total	65670.827	51			
Assigned	**Final Exam**	**Between Groups**	43.932	1	43.932	.061	**.806**
		Within Groups	36002.760	50	720.055		
		Total	36046.692	51			
Successful Course Completion		**Between Groups**	.011	1	.011	.044	**.834**
		Within Groups	12.681	50	.254		
		Total	12.692	51			

A One-way ANOVA was performed at each assessment, holding the pre-registration variable constant and identified there is no significant difference between students who pre-registered and those who did not when students self-selected their partner vs having the partner assigned as shown in Table 11.

Table 11: Analysis of Variance by Pre-Registration

Pair Formation	Assessment		Sum of Squares	df	Mean Square	F	Sig.
Self Selection	Exam 1	Between Groups	37.011	1	37.011	.048	.828
		Within Groups	38579.508	50	771.590		
		Total	38616.519	51			
Assigned	Exam 2	Between Groups	737.759	1	737.759	.801	.375
		Within Groups	46026.987	50	920.540		
		Total	46764.745	51			
Self Selection	Exam 3	Between Groups	26.185	1	26.185	.020	.888
		Within Groups	65644.642	50	1312.893		
		Total	65670.827	51			
Assigned	Final Exam	Between Groups	15.874	1	15.874	.022	.883
		Within Groups	36030.819	50	720.616		
		Total	36046.692	51			
Pass Course		Between Groups	.401	1	.401	1.633	.207
		Within Groups	12.291	50	.246		
		Total	12.692	51			

After evaluating student success by pre-registration and their performance on individual assessments (intermediate exams and the final exam) regardless of partner selection and overall class success; the t-test shows that there is no significant difference in any group performance. Group statistics are shown in Table 12. The results of the t-test are shown in Table 13.

Table 12: Group Statistics by Pre-registration

Pair Formation	Assessment	Preregistered	N	Mean	Std. Deviation	Std. Error Mean
Self-Selection	**Exam1**	**Yes**	29	63.34	29.382	5.456
		No	23	65.04	25.590	5.336
Assigned	**Exam2**	**Yes**	29	44.655	31.0098	5.7584
		No	23	52.239	29.4664	6.1442
Self-Selection	**Exam 3**	**Yes**	29	44.69	35.852	6.658
		No	23	43.26	36.714	7.655
Assigned	**Final Exam**	**Yes**	29	43.93	25.285	4.695
		No	23	45.04	28.706	5.986
Successful Course Completion		**Yes**	29	.66	.484	.090
		No	23	.48	.511	.106

Table 13: Independent Samples Test by Pre-registration

Partner Formation	Assessment		Levene's Test for Equality of Variances		t-test for Equality of Means						
			F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
										Lower	Upper
Self Selection	Exam1	Equal variances assumed	.244	.623	-.219	50	**.828**	-1.699	7.756	-17.277	13.880
		Equal variances not assumed			-.223	49.520	**.825**	-1.699	7.632	-17.031	13.634
Assigned	Exam2	Equal variances assumed	.215	.645	-.895	50	**.375**	-7.5840	8.4715	-24.5995	9.4315
		Equal variances not assumed			-.901	48.326	**.372**	-7.5840	8.4208	-24.5121	9.3442
Self Selection	Exam3	Equal variances assumed	.000	.987	.141	50	**.888**	1.429	10.117	-18.892	21.749
		Equal variances not assumed			.141	46.819	**.889**	1.429	10.145	-18.983	21.841
Assigned	Final Exam	Equal variances assumed	.433	.513	-.148	50	**.883**	-1.112	7.495	-16.167	13.942
		Equal variances not assumed			-.146	44.243	**.884**	-1.112	7.608	-16.442	14.217
Successful Course Completion		Equal variances assumed	2.227	.142	1.278	50	**.207**	.177	.138	-.101	.455
		Equal variances not assumed			1.270	46.103	**.211**	.177	.139	-.104	.457

A One-way ANOVA was performed at each assessment, holding the major variable constant and identified there is no significant difference between students who are declared computer science as their major and those who did not when students self-selected their partner vs having the partner assigned as shown in Table 14.

Table 14: Analysis of Variance by Major

			Sum of Squares	df	Mean Square	F	Sig.
Self Selection	**Exam1**	**Between Groups**	745.974	1	745.974	.985	**.326**
		Within Groups	37870.545	50	757.411		
		Total	38616.519	51			
Assigned	**Exam2**	**Between Groups**	63.719	1	63.719	.068	**.795**
		Within Groups	46701.027	50	934.021		
		Total	46764.745	51			
Self Selection	**Exam3**	**Between Groups**	858.380	1	858.380	.662	**.420**
		Within Groups	64812.447	50	1296.249		
		Total	65670.827	51			
Assigned	**Final Exam**	**Between Groups**	38.734	1	38.734	.054	**.818**
		Within Groups	36007.959	50	720.159		
		Total	36046.692	51			
Successful Course Completion		**Between Groups**	.026	1	.026	.101	**.752**
		Within Groups	12.667	50	.253		
		Total	12.692	51			

Regression analyses (analysis of covariance) based on gender and whether the student preregistered revealed there is no significant difference based on pair formation that predicts success on exam one after selecting the partner (.897), success on exam two after assignment the partner (.596), success on exam three after selecting the partner (.961), success on the final exam (.958), nor whether the students passes the course successfully (.450) as shown in Table 15.

Table 15: Analysis of Covariance

Predictors: (Constant Variables) Pre-registration, Student's Gender

Model	Dependent Variables		Sum of Squares	df	Mean Square	F	Sig.
1	Self Selection Exam1	Regression	170.114	2	85.057	.108	**.897**
		Residual	38446.405	49	784.621		
		Total	38616.519	51			
2	Assigned Exam2	Regression	979.009	2	489.505	.524	**.596**
		Residual	45785.736	49	934.403		
		Total	46764.745	51			
3	Self-Selection Exam3	Regression	106.461	2	53.231	.040	**.961**
		Residual	65564.366	49	1338.048		
		Total	65670.827	51			
4	Assigned Final Exam	Regression	62.855	2	31.428	.043	**.958**
		Residual	35983.837	49	734.364		
		Total	36046.692	51			
5	Successful Course Completion	Regression	.407	2	.203	.811	**.450**
		Residual	12.286	49	.251		
		Total	12.692	51			

The ANOVAs do not reveal any significant differences between gender, pre-registration, pair formation and any exam scores. The majority of students received equivalent scores on their exams. In addition, the ANCOVA does not reveal any predictors that suggest a major contributor to class success. The research supports the hypotheses that self-selection pairs perform as well as or better than randomly assigned pairs.

Hypothesis 3: *Pair programming in CS1 increases retention of CS majors.*

Alternative: *Pair programming in CS1 does not increase retention of CS majors.*

Resolution: Retention was analyzed by comparing all computer science majors enrolled in the lecture course in fall 2004 (baseline data), fall 2005 (pilot study) and spring 2006 (formal study). The students of each semester were analyzed by major, gender and successful completion of the first CS course. Students for each semester were identified first by successful completion. The researcher assumed that a student who does not pass the class will interpret the failure as a negative experience and therefore will be more likely to change their major therefore the first goal of increasing the retention rate is to increase the passing rate. The students were then identified by major. The researcher's goals were to retain students who have identified computer science as their major. Genders of the students were also analyzed to identify if the research measures improved the female retention rate.

The CS majors of fall 04 were divided into 3 categories and each category was then divided by gender to reveal retention statistics by gender. The first category (1) contained students who passed the course and enrolled in the next course during the next semester; these students were also identified if they remained as a CS major one-year later (this is especially important because the highest attrition occurs at the conclusion of the first year of study.) The second category (2) contained students who did not pass the first course, but remained in the major by repeating the course in the next semester. The percentage of these students was also analyzed to investigate if they remained a CS major one-year later. These same intercessions were evaluated in a study conducted at the University of Santa Cruz[34, 35] to identify persistence in the computer science major. Both categories one and two are considered to be retained students. The third category (3) identified students that changed their major at the conclusion of the first semester and at the end of the first year regardless of completion of the course. These students are considered to be non-retained. All of the categories yield percentages to which fall 2005 and spring 2006 data can be compared. A Chi-square test was used to report statistical significance, if any.

The baseline data of fall 2004 yields the following composition as displayed in chart 7; 14 CS majors, 34 non-CS majors, 26 males, 22 females, 23 students passed, and 25 did not pass. Of the CS majors, 8 were male and 6 were female.

Summarized in Table 16, category one shows 35.7% of CS majors passed and were retained at the completion of one semester, 28.6% of the CS majors, as shown in category two, did not pass the course but remained a CS major at the conclusion of the first semester. Summing the percentages yields 64.3% of the CS majors that began in fall 2004 were retained at the end of the first semester. Of those same students, 57.1% of the students were retained at the conclusion of the first academic year regardless of success or non-success in the 2nd enrolled course. All 14 CS majors from the fall 2004 semester were accounted for in the data.

Table 16: Fall 2004 CS Majors Retention Data

N=14	One Semester	Percentages	One Year	Percentages
Cat 1 - A	5	35.7%	8	57.1%
Cat 1 - Males	4	50.0%	6	75.0%
Cat 1 - Females	1	16.7%	2	33.3%
Cat 2	4	28.6%		
Cat 2 - Males	2	25.0%		
Cat 2 - Females	2	33.3%		
Cat 3	5	35.7%	1	7.1%
Cat 3 - Males	2	25.0%	0	0.0%
Cat 3 - Females	3	50.0%	1	16.7%
Total	14	100%	9	64.3%
Retained	9	64.3%	8	57.1%
Did Not Retain	5	35.7%	1	7.1%
M Retained	6	75.0%	6	75.0%
F Retained	3	50.0%	2	33.3%

Table 8 shows that at the conclusion of one year, 75% of the male CS majors were retained and only 33% of the female CS majors were retained. These statistics mirror the national computer science retention rates.

The data from fall 2005 yields the following composition as displayed in chart 8; 49 CS majors, 29 non-CS majors, 56 males, 22 females, 51 students passed, 25 did not pass and 2 students received an incomplete for the final grade. Of the CS majors, 34 were male and 15 were female.

Summarized in Table 17, Category one shows 61.2% of CS majors passed and were retained at the completion of one semester, 18.4% of the CS majors, as shown in category two, did not pass the course but remained a CS major at the conclusion of the first semester. Summing the percentages shows that 79.6% of the CS majors that began in fall 2005 were retained at the end of the first semester. Of those same students, 34.7% of the students were retained at the conclusion of the first academic year regardless of success or non-success in the 2nd enrolled course. All 49 CS majors from the fall 2005 semester were accounted for in the data.

Table 17: Fall 2005 CS Major Retention Data

N=49	One Semester	Percentages	One Year	Percentages
Cat 1 - A	31	63.3%	30	61.2%
Cat 1 - Males	21	61.8%	21	61.8%
Cat 1 - Females	10	66.7%	9	60.0%
Cat 2	9	18.4%		
Cat 2 - Males	7	20.6%		
Cat 2 - Females	2	13.3%		
Cat 3	9	18.4%	10	20.4%
Cat 3 - Males	6	17.6%	7	20.6%
Cat 3 - Females	3	20.0%	3	20.0%
Total	49	100%	40	81.6%
Retained	40	81.6%	30	61.2%
Did Not Retain	9	18.4%	10	20.4%
M Retained	28	82.4%	21	61.8%
F Retained	12	80.0%	9	60.0%

The baseline data of spring 2006 yields the following composition; 27 CS majors, 25 non-CS majors, 37 males, 15 females, 30 students passed, 21 students did not pass and 1 student received an incomplete.

Summarized in Table 18, Category one shows 48.1% of CS majors passed and were retained at the completion of one semester, 14.8% of the CS majors, as shown in category two, did not pass the course but remained a CS major at the conclusion of the first semester. Summing the percentages shows that 63.0% of the CS majors that began in spring 2006 were retained at the end of the first semester.

Table 18: Spring 2006 CS Major Retention Data

N=27	One Semester	Percentages
Cat 1 - A	13	48.1%
Cat 1 - Males	10	55.6%
Cat 1 - Females	3	33.3%
Cat 2	4	14.8%
Cat 2 - Males	2	11.1%
Cat 2 - Females	2	22.2%
Cat 3	10	37.0%
Cat 3 - Males	0	0.0%
Cat 3 - Females	4	44.4%
Total	27	100%
Retained	17	63.0%
Did Not Retain	10	37.0%
Males Retained	12	66.7%
Females Retained	5	55.6%

Table 18 shows that at the conclusion of one semester, 66.7% of the male CS majors were retained and 55.6 % of the female CS majors were retained.

Chart 10 shows the fluctuation in retention of all CS majors and by gender. Analysis of the fluctuation suggests that more data per semester is needed to establish a true trend and suggestion for the retention of CS majors at NSU to reduce the number of anomalies that may influence the retention statistics.

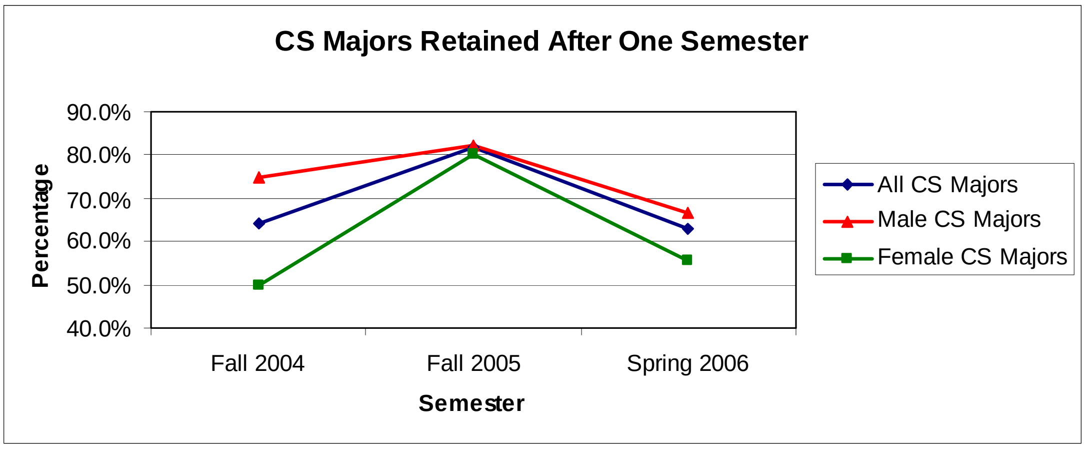


Chart 6: CS Majors Retained After One Semester

When comparing the number of CS majors retained after one semester, the difference is statistically significant from the fall 2004 semester (64.3%) to the fall 2005 semester (81.6%), p = 0.005 using the Chi-Square statistical measurement. However the difference from fall 2004 to spring 2006 (63%) is not statistically significant (p=0.848) and the percentages show a slight decrease in the percentage of CS majors retained at the conclusion of one semester.

Examining the retention rates by gender shows that from fall 2004 to fall 2005, the increase in the retention of male CS majors is slight (7.4%) and the difference is not significantly different (p=0.201). However, the difference in the number of female CS majors retained is vast (30%) and that difference is significantly different using the Chi-Square test, $p = 8.69 \times 10^{-6}$.

Again the differences in gender from fall 2004 to spring 2006 do not reveal any statistical significance, p=0.196 for males. The percentage of males actually retained shows a decrease of 8.3% however, the data shows no support for the hypothesis in this group. The percentage of females only increased slightly (5.6%) and is not statistically significant using the Chi-square test (p= 0.427).

Retention of CS majors enrolled one year later was analyzed only on the students beginning during the fall 2005 semester and compared to the students enrolled during the fall 2004 semester. Majors from spring 2006 enrolled one year later can not be measured or forecasted as the spring 2007 semester is in the future. A longitudinal study will be implemented and reported in future work.

The difference in the retention of CS Majors for fall 2004 (57.1%) and fall 2005 (61.2%) is only slightly improved and statistically non-significant. The differences by gender however, are very interesting. The increase in females retained from fall 2004 (33.3%) to fall 2005 (60%) reports statistical significance, p=0.000154, using the Chi-Square test. The retention of males one year later from fall 2004 was 75%, comparatively, the retention of males one year later from fall 2005 was 61.8%. Chart 11 shows each representation by percentage.

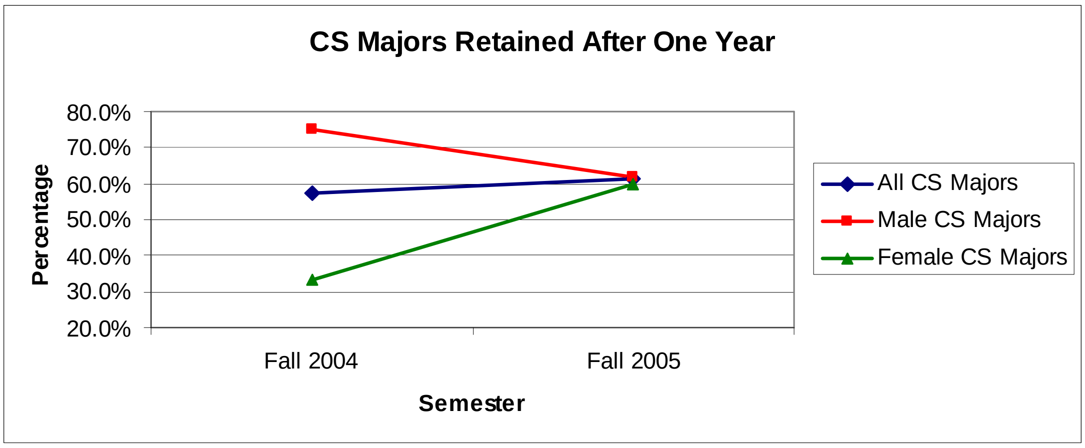


Chart 7: CS Majors Retained After One Year

The researcher offers the following as possible rationale for decrease in retention of male CS majors. During the fall 2004 semester there were only 8 male CS majors compared to the 34 male CS majors enrolled during the fall 2005 semester, a more than 300% increase in the male CS major population. Also, the possibility that research methods designed to improve the retention of females may have an adverse reaction on the retention of males. This issue needs to be addressed and will be included as a topic for future work.

The research does not support the hypothesis that pair programming in CS1 increases the retention of all CS majors; however it does suggest that pair programming in CS1 increases the retention of female CS majors at the completion of one semester and one year. There is no data to support the hypothesis among male CS majors. A longitudinal study is needed to accurately suggest a trend in the retention of all CS majors regardless of gender.

Chapter 5

5.0 Conclusions

There are two major implementations that affected the spring 2006 students that the fall 2004 (baseline) students did not use. Those implementations were 1) the addition of programming laboratory course and 2) the inclusion of the pair programming practice to solve all software development assignments.

The laboratory was introduced to give CS majors a dedicated time when they can practice coding techniques under the supervision of a faculty member and/or teaching assistant. The faculty member or TA is present to assist with syntactical errors that often frustrate beginning programmers. The frustration is a large contributor in a student's decision to elect or maintain computer science as an academic major. As shown in Chart 8, lab enrollment only of the student was not a major contributor to student success in the CSC 170 lecture course. Individual assessment analysis shows that there is no significant improvement of students enrolled in both the lab and the lecture course over those enrolled in the lecture course only. The results show that those students enrolled in the lecture only performed consistently equitably with those students enrolled in the lab and the lecture course.

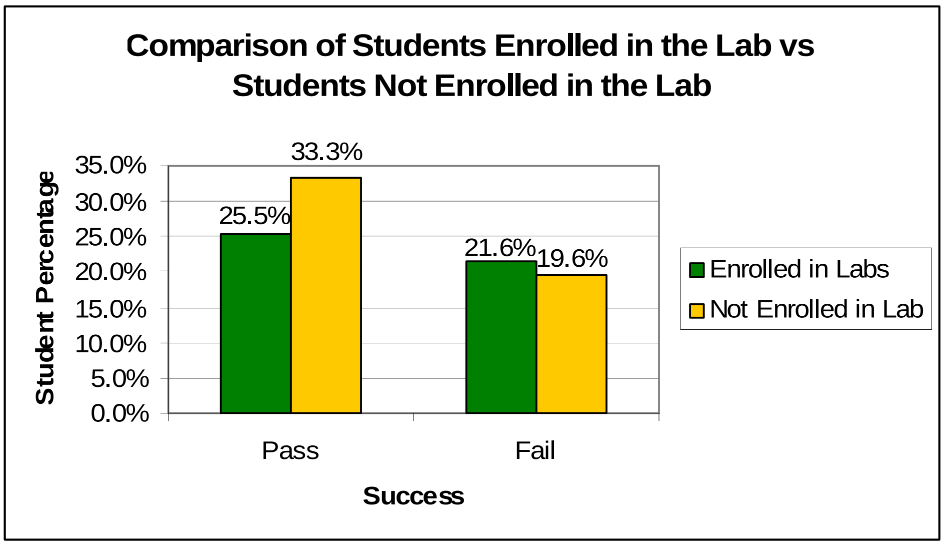


Chart 8: Comparison of Student Success based on Lab Enrollment

Of those students enrolled during the spring 2006 semester, 58.8% passed the lecture course, and of those, 33.3% of those students were not enrolled in the lab. The laboratory was only required of CS majors. Other majors, such as mathematics majors have the necessary logical skills to do well in beginning computer science courses and will find the necessary time to apply the lecture with hands on activities.

The inclusion of the pair programming practice to solve all software development assignments allowed the researcher to investigate the problem of whether pair formation influences student satisfaction and therefore their retention as a computer science major. Several ideas were explored in part; 1) Satisfaction of CS1 students is increased when engaged in self-selection pair programming, 2) Self-selection pairing is a beneficial pedagogical technique and 3) Self-selection pairing leads to the retention of CS1 students.

The researcher concludes that satisfaction of CS1 students is increased when engaged in self-selection pair programming based on the statistical significance shown in tests that compare satisfaction between students engaging in self-selection pairing and randomly assigned pairing. The researcher also concludes that self-selection pairing is a beneficial pedagogical technique based on the documented research that suggests pair programming increases the ability of students learning to program regardless of pair-formation methodology and the results of the individual assessments measured in this study suggests that self-selection of partners does not decrease the overall effects of pair programming in CS1 students. In addition, statistical measurements did not show that other factors contributed significantly to the success of the students in enrolled in the courses.

Finally, the research does not support the hypotheses that pair programming in CS1 increases the retention of all CS majors; however it does support the idea that pair programming in CS1 increases the retention of female CS majors. Retention was measured at various points of attrition, the highest being at the conclusion of the first year of study. Documented research suggests that one of the largest influences on retention is the satisfaction of the CS1 student at the conclusion of year one. This study showed an increase in female retention at the end of the course and at the end of the first year.

Documented research also suggests that pair programming increases a student's confidence and reduces his/her frustration level. All of these factors influence a student's decision on whether to maintain computer science as a major or not. The researcher's premise is to increase the number of factors that influence a student's decision to retain computer science as a major. The major goal was to show data that suggests that satisfaction is increased when students participate in self-selection pairing and therefore increases the retention rate of computer science majors. Students expressed more positive feelings of satisfaction with the pair programming process when the students self-selected their partner. Anecdotal data suggests that having the choice to self-select the partner allows the students to work harder at arranging to meet for pair programming. In all statistical measurements, the students consistently preferred to choose their partner over having the partner assigned to them. In a CS1 course, the election to implement self-selection pair formation or random pair formation is trivial and students will perform equally whether the pairs are assigned or self-selected, however self-selection aids in the satisfaction of a student and therefore impacts their decision to retain computer science as their major. Moreover, the results of the research were very successful and the implementation of the programming labs and the use of self-selection pair programming continue to be employed in the Computer Science department at Norfolk State University.

5.1 Future Work

A vast amount of research has been conducted regarding pair programming in the academic environment. Some research has shown that different implementations work with different sets of group while there is one common theme: pair programming in the academic environment works. This research has shown that allowing CS1 students to select their partner does not have an adverse effect on the learning of students and often increases the likelihood that the student will enjoy the process although there are more avenues to explore. The following list represents some of those possibilities for additional research.

1. Implement self-selection pair programming over an extended period of time to analyze its effects on long-term retention. More data is needed to accurately suggest

an improvement in retention based on the self-selection pairing. Many things can affect the retention of CS majors and only continued implementation of the technique will allow a researcher to establish a trend line across semesters to suggest that retention has improved.

2. Identify and test pedagogy techniques that can be used to reduce the parasitic behavior in freshmen and non-interested students. Those techniques may include topic delivery methods and/or software that can be used to keep the student focused and enjoying the lecture course.
3. Later research has identified many techniques used to increase the retention of females. Many of those techniques suggest creating environments that are more social and inclusive. The techniques for creating the environment should be analyzed to establish that those techniques used to increase the retention of female CS majors do not cause a subsequent adverse reaction on the retention of male CS majors. This study should look at those methods that are nurturing for females but could be considered condescending for males.
4. Implement the research methodology in separate course sections; one for self-selection pairing and the other for random pairing. Evaluate the measures over at least two semesters to investigate whether self-selection pairing increases the retention of CS majors.

APPENDIX A: Spring 2006 Execution Plan

CSC 170/170L
Spring 2006

Week/Date	**Lab Class (170L)**	**Lecture Class (170)**
1/ Jan 9	Introduction	Q1
2/ Jan 16	L1, Individual, Environment Intro	Q2
3/ Jan 23	L2, Selection	Q3
4/ Jan 30	L3, Selection	T1, PA1-Selection, Assessment using OPEF
5/ Feb 6	L4, Selection, Assessment of L2-L4 using OPEF	Q4
6/ Feb 13	L5, Assigned	Q5, PA2, Assigned, Assessment using OPEF
7/ Feb 20	L6, Assigned	Q6
8/ Feb 27	L7- Assigned	Q7
9/ Mar 6	L8-Assigned, Assessment of L5-L8 using OPEF	T2, PA3, Choice of Selection or Assigned, Assessment using OPEF
10/ Mar 13	**Spring Break**	
11/ Mar 20	L9, Selection	Q8
12/ Mar 27	L10, Selection	Q9, PA4, Selection, Assessment using OPEF
13/ Apr 3	L11, Selection, Assessment of L9-L11 using OPEF	Q10
14/ Apr 10	L12, Assigned	T3, PA5, Assigned, Assessment using OPEF
15/ Apr 17	L13, Assigned	Q11
16/ Apr 24	L14, Assigned, Assessment of L12-L14 using OPEF.	Q12, PA6, Individual, Assessment using OPEF
17/May 1	**Exam Week**	

L-Lab
Q-Quiz
T-Test
PA-Programming Assignment

170 – measures experience with one partner with one assignment over time period of 2-3 weeks
170L – measures experience with one partner with 3-4 assignments over time period of 3-4 weeks
Assigned or Selection indicates the method in which the pair should be created. All assigned pairs should be created without regard to factors such as personality type, skill level, self esteem, and self-assessment.

APPENDIX B: Peer Evaluation Form (Fall 05)

Laboratory # ____

Your Name ____________________________ Partner Name ___________________________

Mark an X in the box that represents the extent to which you agree with each statement

	Strongly Disagree	Disagree	Neither agree nor disagree	Agree	Strongly Agree
11. My partner made a serious effort to complete the assigned work.					
12. My partner contributed to team efforts.					
13. My partner was technically capable of completing this week's lab exercise.					
14. My partner and I were compatible when we worked together on this exercise.					
15. My partner helped me to complete this exercise.					
16. I enjoyed working with my partner on this exercise.					
17. My partner improved my ability to complete this exercise.					
18. The quality of this programming exercise was improved because my partner and I worked together.					
19. My partner treated me with respect while working on this exercise.					
20. I was more confident completing this exercise because my partner worked with me.					

Did you select your partner? Yes ○ No ○

If No, would you rather select your partner? Yes ○ No ○ Doesn't matter ○

If Yes, Why?____

How long have your known your partner?

○ Just met this semester (JMS).

○ JMS, we are in more than one class together. Which courses?

____

○ I knew my partner prior to the Fall 2005 semester

Did you enjoy the pair programming experience with this partner?

____

____

What was the partner's most significant contribution?

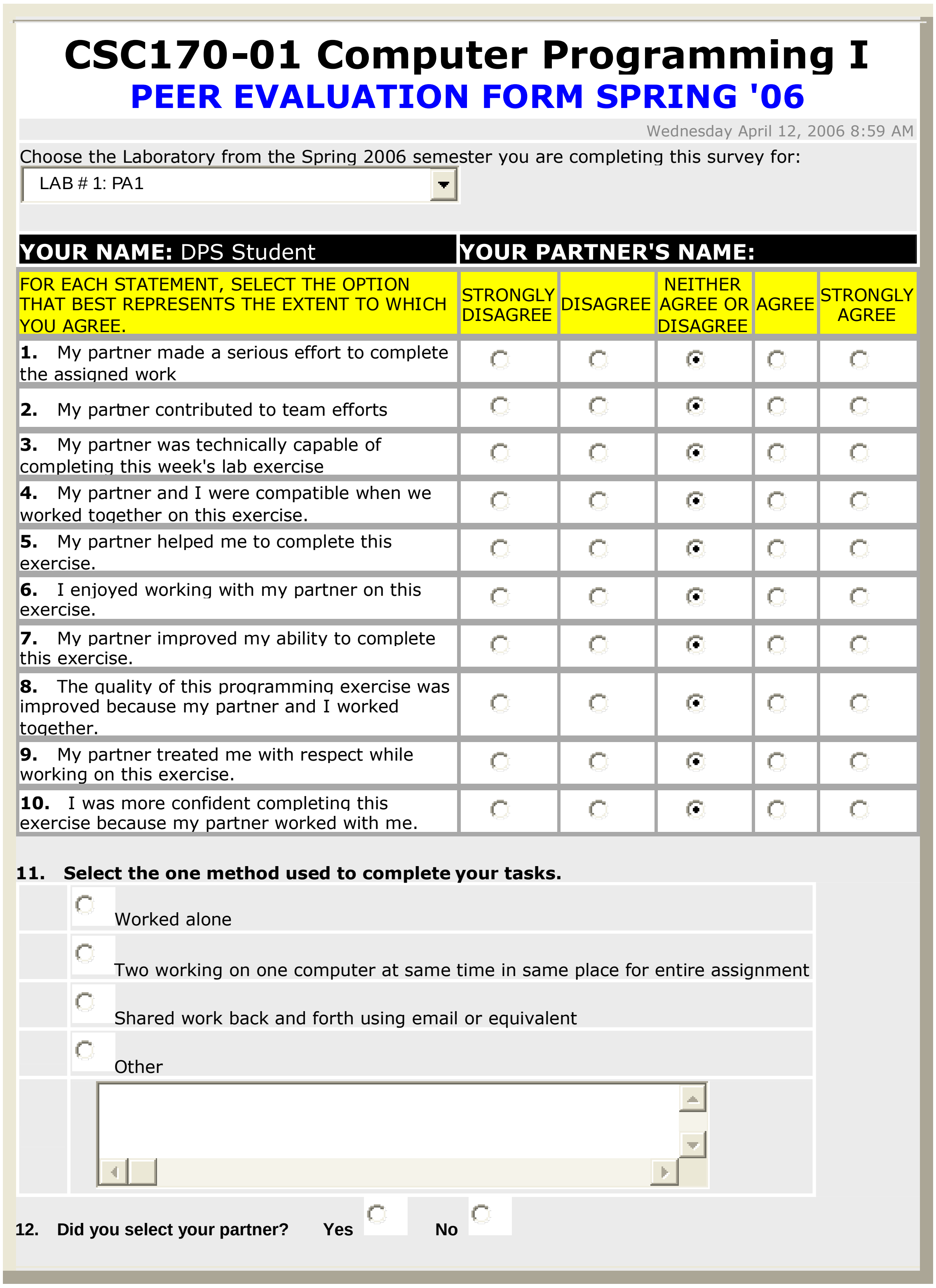
____

____

What was most irritating about working with this partner?

____

APPENDIX C: Online Peer Evaluation Form



CSC170-01 Computer Programming I
PEER EVALUATION FORM SPRING '06

Wednesday April 12, 2006 8:59 AM

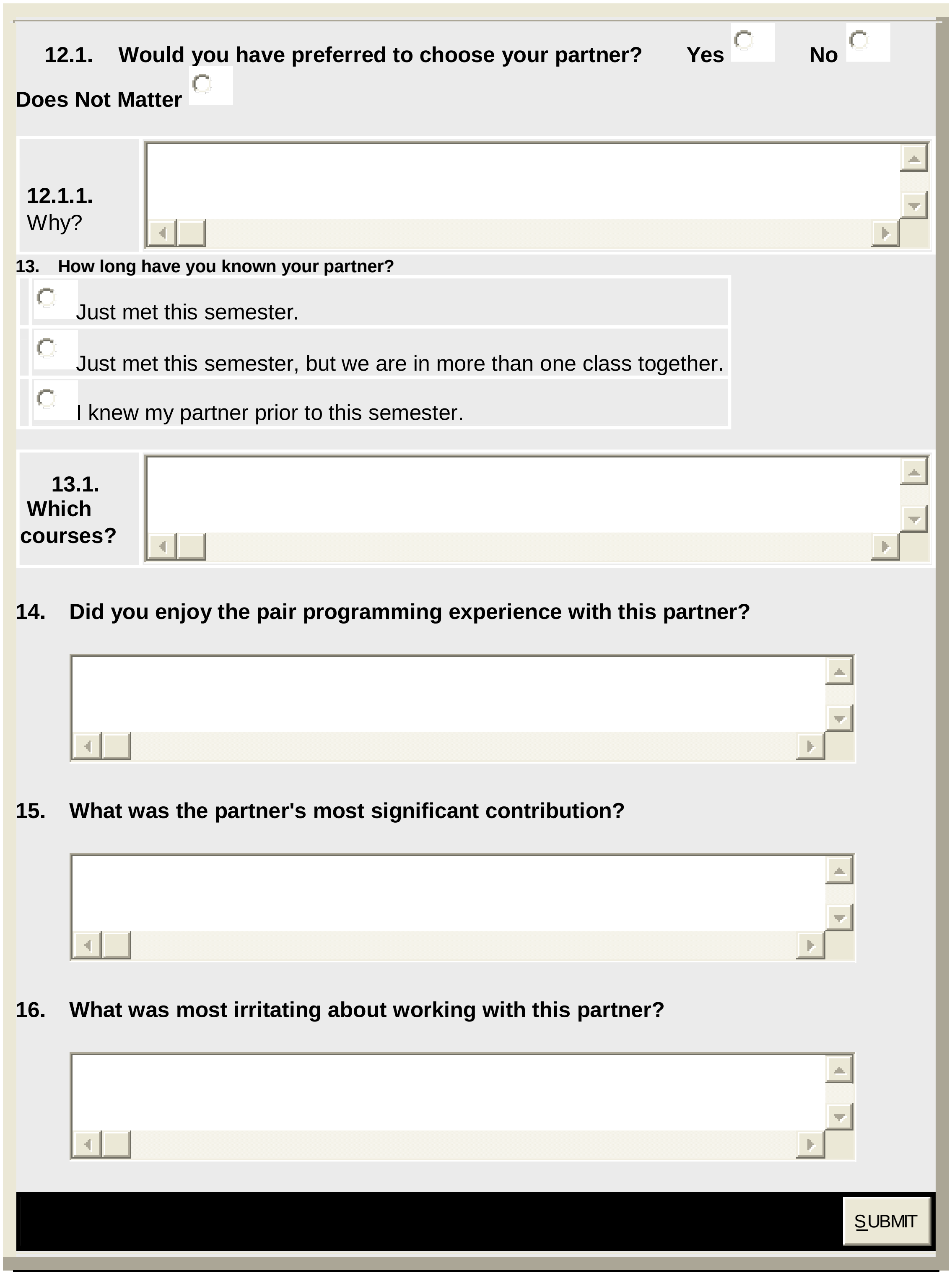
Choose the Laboratory from the Spring 2006 semester you are completing this survey for:
LAB # 1: PA1

YOUR NAME: DPS Student	YOUR PARTNER'S NAME:				
FOR EACH STATEMENT, SELECT THE OPTION THAT BEST REPRESENTS THE EXTENT TO WHICH YOU AGREE.	STRONGLY DISAGREE	DISAGREE	NEITHER AGREE OR DISAGREE	AGREE	STRONGLY AGREE
1. My partner made a serious effort to complete the assigned work	○	○	●	○	○
2. My partner contributed to team efforts	○	○	●	○	○
3. My partner was technically capable of completing this week's lab exercise	○	○	●	○	○
4. My partner and I were compatible when we worked together on this exercise.	○	○	●	○	○
5. My partner helped me to complete this exercise.	○	○	●	○	○
6. I enjoyed working with my partner on this exercise.	○	○	●	○	○
7. My partner improved my ability to complete this exercise.	○	○	●	○	○
8. The quality of this programming exercise was improved because my partner and I worked together.	○	○	●	○	○
9. My partner treated me with respect while working on this exercise.	○	○	●	○	○
10. I was more confident completing this exercise because my partner worked with me.	○	○	●	○	○

11. Select the one method used to complete your tasks.

○ Worked alone
○ Two working on one computer at same time in same place for entire assignment
○ Shared work back and forth using email or equivalent
○ Other

12. Did you select your partner? Yes ○ **No** ○



12.1. Would you have preferred to choose your partner? **Yes** ○ **No** ○

Does Not Matter ○

12.1.1. Why?

13. How long have you known your partner?

○ Just met this semester.

○ Just met this semester, but we are in more than one class together.

○ I knew my partner prior to this semester.

13.1. Which courses?

14. Did you enjoy the pair programming experience with this partner?

15. What was the partner's most significant contribution?

16. What was most irritating about working with this partner?

APPENDIX D: Demographic Data Sheet

ID – Data:										
CS Major:	Y	N								
Active Major:										
Gender:	M	F								
Ethnicity:	B	W	A	MX						
Pass CSC 170:		Y	N							
Final Grade:	A	A-	B+	B	B-	C+	C	C-	D	F
SAT-M:										
HS-GPA										
C-GPA										
Hometown City:										
Hometown State:										
Hometown Zip:										

Acronyms and Glossary

CS	Computer Science
CS1 or CSC170	Computer Programming I course
CS1-L, CS2-L	Computer Programming Laboratory course
HBCU	Historically Black College and University
NSU	Norfolk State University
SST	School of Science and Technology
IT	Information Technology
NCSU	North Carolina State University
NSF	National Science Foundation
XP	Extreme Programming
SE	Software Engineering
Pair Programming	Two programmers working side-by-side at the same designs, algorithms, codes, and tests
Extreme Programming	A discipline of software development based on values, communication, simplicity, feedback, courage, and respect
Peer Evaluation	The mechanism used by students to evaluate their partner during a pair programming activity to maintain individual accountability
Pair Formation	The method in which students are partnered
Pair Rotation	Students pair with different classmates throughout the semester
Driver	One of the pair that types at the computer or writes down a design
Navigator	One of the pair that has the task of monitoring the driver's work for tactical and strategic defects
Random Pairing	The method of pair formation where the instructor identifies and assigns all pairing groups without regard to factors that some research suggests increases pair compatibility
Self-Selection Pairing	The method of pair formation in which a partner directly self-selects with whom they want to work

References

[1] "The Carnegie Classification of Institutions of Higher Education". 2005, The Carnegie Foundation for the Advancement of Teaching, http://www.carnegiefoundation.org/classifications/index.asp.

[2] Baheti, P., E. Gehringer, and D. Stotts, "Exploring the Efficacy of Distributed Pair Programming", in *Extreme Programming/ Agile Universe*. 2002, Springer: Chicago, IL.

[3] Baheti, P., et al. "Exploring Pair Programming in Distrubuted Object-Oriented Team Projects". in *OOPSLA Educator's Symposium*. 2002. Seattle, WA.

[4] Beck, K., "Extreme Programming Explained: Embrace Change". The XP Series. 2000: Addison- Wesley.

[5] Beck, L.L., A.W. Chizhik, and A.C. McElroy, "Cooperative learning techniques in CS1: design and experimental evaluation", in *Proceedings of the 36th SIGCSE technical symposium on Computer science education*. 2005, ACM Press: St. Louis, Missouri, USA.

[6] Berenson, S.B., et al., "Voices of women in a software engineering course: reflections on collaboration". J. Educ. Resour. Comput., 2004. **4**(1): p. 3.

[7] Bergin, J., et al., "Teaching software development methods: the case of extreme programming", in *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 2004, ACM Press: Norfolk, Virginia, USA.

[8] Bevan, J., L. Werner, and C. McDowell. "Guidelines for the Use of Pair Programming in a Freshman Programming Class". in *Conference on Software Engineering and Training*. 2002.

[9] Beyer, S., et al., "Gender differences in computer science students". SIGCSE Bull., 2003. **35**(1): p. 49-53.

[10] Cliburn, D., "Experiences with pair programming at a small college". The Journal of Computing in Small Colleges, 2003. **19**(10): p. 20-29.

[11] Cockburn, A. and L. Williams, "The Costs and Benefits of Pair Programming", in *Extreme Programming Examined*. 2002, Addison-Wesley: Boston.

[12] Cohoon, J.M., "Must there be so few?: including women in CS", in *Proceedings of the 25th International Conference on Software Engineering*. 2003, IEEE Computer Society: Portland, Oregon.

[13] Cohoon, J.M., "Toward improving female retention in the computer science major". Commun. ACM, 2001. **44**(5): p. 108-114.

[14] DeClue, T., "Pair programming and pair trading: effects on learning and motivation in a CS2 course". The Journal of Computing in Small Colleges, 2003. **18**(5): p. 49-56.

[15] DeLoatch, S., "Our Profile: School of Science and Technology ". 2005, http://sst.nsu.edu/ourprofile.php.

[16] Domino, M.A., et al., "Conflict in collaborative software development", in *Proceedings of the 2003 SIGMIS conference on Computer personnel research: Freedom in Philadelphia--leveraging differences and diversity in the IT workforce*. 2003, ACM Press: Philadelphia, Pennsylvania.

[17] Freeman, S., B. Jaeger, and J. Brougham. "Pair Programming: More Learning and Less Anxiety in a First Programming Course". in *ASEE Annual Conference & Exposition: Staying in Tune with Engineering Education*. 2003. Nashville, TN: Northeastern University Monash University.

[18] Hanks, B., "Student performance in CS1 with distributed pair programming", in *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. 2005, ACM Press: Caparica, Portugal.

[19] Hanks, B. and C. McDowell. "Program Quality with Pair Programming in CS1". in *9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 2004. Leeds, Unikted Kingdom.

[20] Hans, G., A. Erik, and D. Tore. "An Initial Framework for Research on Pair Programming". 2003.

[21] Hickey, T.J., J. Langton, and R. Alterman, "Enhancing CS programming lab courses using collaborative editors". J. Comput. Small Coll., 2005. **20**(3): p. 157-167.

[22] Ho, C.-w., et al., "Examining the Impact of Pair Programming on Female Students", in *NCSU Technical Report*. 2004.

[23] Jefferies, R., "What is XP?" in *XProgramming.com an Agile Software Development Resource*. 2001.

[24] Jepson, A. and T. Perl, "Priming the pipeline". SIGCSE Bull., 2002. **34**(2): p. 36-39.

[25] Katira, N., L. Williams, and J. Osborne. "Towards Increasing the Compatibility of Student Pair Programmers". in *International Conference on Software Engineering (ICSE) 2005*. 2005.

[26] Katira, N., et al. "On Understanding Compatibility of Student Pair Programmers". in *ACM Technical Symposium on Computer Science Education, SIGCSE 2004*. 2004. Norfolk, VA.

[27] Kist, S., "Project Management and Mythical Manmonths". 2001, Korean Advanced Institute of Science and Technology

[28] Loftus, C. and M. Ratcliffe, "Extreme programming promotes extreme learning?" in *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. 2005, ACM Press: Caparica, Portugal.

[29] Margolis, J. and A. Fisher. "Geek Mythology and Attracting Undergraduate Women to Computer Science". in *Joint National Conference of th eWomen in Engineering Program Advocates Network and the National Association of Minority Engineering Program Administrators*. 1997.

[30] Margolis, J. and A. Fisher, "Unlocking the Clubhouse: Womed in Computing". 2002, Cambridge, Massachusetts: MIT Press.

[31] Matthias, M.M. and P. Frank. "An Empirical Study about the Feelgood Factor in Pair Programming". 2004.

[32] McDowell, C., B. Hanks, and L. Werner. "Experimenting with Pair Programming in the Classroom". in *ITiCSE*. 2003. Thessaloniki, Greece.

[33] McDowell, C., et al. "The Effects of Pair Programming on Performance in an Introductory Programming Course". in *SIGCSE '02*. 2002. Northern Kentucky.

[34] McDowell, C., et al., "Pair Programming Improves Student Retention, Confidence, and Program Quality". Commun. ACM, 2006. **49**(8): p. 90-95.

[35] McDowell, C., et al. "The Impact of Pair Programming on Student Performance, Perception and Persistence". in *25th International Conference on Software Engineering*. 2003.

[36] McKinney, D. and L.F. Denton, "Affective assessment of team skills in agile CS1 labs: the good, the bad, and the ugly", in *Proceedings of the 36th SIGCSE technical symposium on Computer science education*. 2005, ACM Press: St. Louis, Missouri, USA.

[37] McKinney, D. and L.F. Denton, "Developing collaborative skills early in the CS curriculum in a laboratory environment", in *Proceedings of the 37th SIGCSE technical symposium on Computer science education*. 2006, ACM Press: Houston, Texas, USA.

[38] Melnik, G. and F. Maurer, "Perceptions of Agile Practices: A Student Survey", in *Extreme Programming/ Agile Universe*. 2002: Chicago, IL.

[39] Muller, M.M. and W.F. Tichy, "Case study: extreme programming in a university environment", in *Proceedings of the 23rd International Conference on Software Engineering*. 2001, IEEE Computer Society: Toronto, Ontario, Canada.

[40] Nagappan, N., et al. "Improving the CS1 Experience with Pair Programming". in *ACM Technical Symposium on Computer Science Education, SIGCSE 2003.* 2003.

[41] Nawrocki, J. and A. Wojciechowski. "Experimental Evaluation of Pair Programming". in *European Software Control and Metrics*. 2001. London, England.

[42] Nosek, J.T., "The case for collaborative programming". Commun. ACM, 1998. **41**(3): p. 105-108.

[43] NSU, O.o.A.A., "About NSU", http://www.nsu.edu/about/.

[44] NSU, O.o.E.M., "Norfolk State Univesity Fall 2005 Facts". 2005, http://www.nsu.edu/enrollmentmanagement/IRfactsheet.html.

[45] Palmieri, D.W., "Knowledge management through pair programming". 2002. p. vi, 80 p.

[46] Preston, D., "Pair Programming as a Model of Collaborative Learning: A Review of the Research". Journal of Computing Sciences in Colleges, 2005. **20**(4).

[47] Preston, D., "Using collaborative learning research to enhance pair programming pedagogy". SIGITE Newsl., 2006. **3**(1): p. 16-21.

[48] Sanders, D. "Student Perceptions of the Suitability of Extreme and Pair Programmng". in *XP Agile Universe*. 2001.

[49] Shukla, A., "Pair programming and the factors affecting Brooks' Law". 2002. p. vii, 75 p.

[50] Slaten, K., et al., "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction". Agile 2005, 2005: p. 323-330.

[51] Srikanth, H., et al. "On Pair Rotation in the Computer Science Course". in *Conference on Software Engineering Education and Training*. 2004.

[52] Stotts, D., et al., "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming", in *Extreme Programming and Agile Methods - XP/Agile Universe 2003* 2003, Springer Berlin / Heidelberg New Orleans. p. 129 - 141

[53] Succi, G. and M. Marchesi, "Extreme Programming Examined". The XP Series. 2001: Addison Wesley.

[54] Sullivan, S.L., "Reciprocal peer reviews", in *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education.* 1994, ACM Press: Phoenix, Arizona, United States.

[55] Thomas, L., M. Ratcliffe, and A. Robertson, "Code warriors and code-a-phobes: a study in attitude and pair programming", in *Proceedings of the 34th SIGCSE technical symposium on Computer science education*. 2003, ACM Press: Reno, Navada, USA.

[56] Thonas, L., M. Ratcliffe, and A. Robertson. "Code Warrior and Code-a-Phobes: A Study in Attitude and Pair Programming". in *SIGCSE 2003*. 2003. Reno, NN.

[57] University, N.S., "About NSU", http://www.nsu.edu/about/.

[58] VanDeGrift, T., "Coupling pair programming and writing: learning about students' perceptions and processes", in *Proceedings of the 35th SIGCSE technical symposium on Computer science education.* 2004, ACM Press: Norfolk, Virginia, USA.

[59] Werner, L.L., B. Hanks, and C. McDowell, "Pair-programming helps female computer science students". J. Educ. Resour. Comput., 2004. **4**(1): p. 4.

[60] Wiebe, E., et al. "Pair Programming in Introductory Programming Labs". in *American Society for Engineering Education Annual Conference & Exposition.* 2003.

[61] Williams, L. "But, isn't that cheating? Collaborative programming." in *29th Annual Frontiers in Education Conference.* 1999. Champaign, IL.

[62] Williams, L., "The Collaborative Software Process", in *Computer Science*. 2000, Univeristy of Utah: Utah. p. 206.

[63] Williams, L., "Debunking the Nerd Stereotype with Pair Programming". 2006. p. 83-85.

[64] Williams, L. and R. Kessler, "All I Really Need to Know about Pair Programming I Learned In Kindergarten". Communications of the ACM, 2000.

[65] Williams, L. and R. Kessler. "The effects of "pair pressure" and "pair-learning" on software engineering education." in *Software Engineering Education and Training*. 2000. Los Alamitos, CA.

[66] Williams, L. and R. Kessler, "Experimenting with Industry's "Pair-Programming" Model in the Computer Science Classroom". Computer Science Education, 2001.

[67] Williams, L. and R. Kessler, "Pair Programming Illuminated". 1st ed. 2003: Addison Wesley. 288.

[68] Williams, L., et al., "Strengthening the Case for Pair-Programming"". IEEE Software 2000, 2000. **17**: p. 19-35.

[69] Williams, L., et al. "Building Pair Programming Knowledge through a Family of Experiments". in *IEEE International Symposium on Empirical Software Engineering (ISESE) 2003.* 2003.

[70] Williams, L. and R.L. Upchurch. "In Support of Student Pair Programming". in *SIGCSE Conference on Computer Science Education.* 2001. Charlotte, NC.

[71] Williams, L., et al., "In Support of Pair Programming in the Introductory Computer Science Course". Computer Science Education, 2002.

[72] Williams, L., et al. "Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations". in *OOPSLA Educator's Symposium 2002.* 2002.

[73] Wilson, D.G., "An empirical study of the tacit knowledge management potential of pair programming". 2004. p. vi, 40 p.

[74] Wilson, J.D., N. Hoskin, and J.T. Nosek, "The benefits of collaboration for student programmers", in *Proceedings of the twenty-fourth SIGCSE technical symposium on Computer science education.* 1993, ACM Press: Indianapolis, Indiana, United States.