The Impact of Remote Pair Programming in an Upper-Level CS Course

Zachariah J. Beasley
arjohns2@usf.edu
University of South Florida
Tampa, Florida, USA

Ayesha R. Johnson
arjohns2@usf.edu
University of South Florida
Tampa, Florida, USA

ABSTRACT

Pair programming is an active learning technique with several benefits to students, including increasing participation and improving outcomes, particularly for female computer science students. However, most of the literature highlights the effects of pair programming in introductory courses, where students have different prior programming experience and thus may experience group issues. This work analyzes the effect of pair programming in an upper-level computer science course, where students have a more consistent background education, particularly in languages learned and coding best practices. Secondly, the effect of remote pair programming on student outcomes is still an open question of increasing importance with the advent of Covid-19. This work utilized split sections with a control and treatment group in a large, public university. In addition to comparing pair programming to individual programming, results were analyzed by modality (remote vs. in-person) and by gender, focusing on how pair programming benefits female computer science students in confidence, persistence in the major, and outcomes. We found that pair programming groups scored higher on assignments and exams, that remote pair programming groups performed as well as in-person groups, and that female students increased their confidence in asking questions in class and scored 12% higher in the course when utilizing pair programming.

CCS CONCEPTS

• Applied computing → Collaborative learning; • Social and professional topics → Computing education.

KEYWORDS

Pair programming; active learning; programming experience; computer science; collaborative learning; student outcomes; experimental research; randomized controlled trial.

1 INTRODUCTION

Pair programming, where two students share one screen while working on a programming assignment, has often been heralded as an effective active learning technique to increase participation and improve outcomes in the classroom, particularly when working on computer science. One student, the driver, writes the program, while the other, the navigator, provides direction, encouragement, and debugging in real-time. Student benefits from this collaborative learning approach include 1) the opportunity for "continuous review" where defects are corrected as they arise, 2) greater satisfaction in the course and learning from peers, 3) team building, communication, and improvement of other soft skills, and 4) greater industry and productivity [8, 17]. Researchers have discovered a 95% increase in confidence in the final product and found that "pair programming is 40-50% faster than programming alone" [12]. Others have noted that "programming assignment grades, exam scores, and persistence in computer programming courses" improve if pair programming is done properly [16]. For instructors, pair programming can provide a barrier to academic dishonesty, promote healthy social interaction in a post-pandemic environment of relative isolation, and allow students to coach and learn from one another [17].

For all its benefits, there are still several open questions in pair programming. One is the effect of remote pair programming on student outcomes, i.e., whether students must be physically together to experience the full benefits. This question is of particular importance post-pandemic as students are more comfortable with online work. As our university transitioned back to the in-person modality, we found that most students (95%) preferred to work on assignments remotely. A second consideration is the optimal pairing of programming partners. For example, Bowman et al. found a negative effect when pairing students with significantly different programming ability [5]. Since most of the literature highlights the effect of pair programming in introductory courses, partners can have a wide divergence in ability and thus experience problems with one partner leaving the other behind. Relatively little work has studied pair programming in upper-level computer science courses where students have a more consistent background education. Thus, the goals of this work are to examine the effect of pair programming 1) where students could work remotely on assignments and 2) on an upper-level, CS-3 course where students should exhibit greater consistency in education, style, and languages learned. This work utilized split sections with a control and treatment group at a large, public university. Results were analyzed with a particular focus on how pair programming benefits female computer science students in confidence, persistence in the major, and outcomes.
2 PREVIOUS RESEARCH

Much of the prior work on pair programming in computer science education has studied its effects in introductory courses, where most students have little or no formal exposure to programming languages and paradigms. Furthermore, those students who do have prior experience come from a variety of backgrounds with various levels of competency and languages learned. Students in a pair programming group in such an introductory course can experience mixed results, often related to the differing levels of prior experience. For example, one large scale study conducted over two years in three different introductory courses (CS, information science, and humanities) at the University of Iowa found that "students who worked with a more experienced partner actually had poorer outcomes, including lower effort exerted on the assignment, perceptions that their partner gave more effort than they did, less time in the driving role (i.e., typing out the assignment), lower understanding of concepts from lab, and less interest in computer science overall" [5]. Another study found that women in an introductory CS course who were randomly paired on assignments found it beneficial to have "someone to ask questions and discuss ideas [with]", especially before they would approach a TA, and that it "improves understanding" [19]. But several women in the same study also reported a feeling that they were "burdening their partner" due to less programming experience. Finally, the unknowns in how to best assign pair programming partners and how to navigate complex partner interaction has led some to propose automating the pair programming partner as a conversational agent that adapts to the user’s skill level [14]. While an exciting research direction, the ability of such an agent to solve complex problems as either the driver or the navigator is currently infeasible.

In regards to remote pair programming, a study of distributed team programming (with two to four students per team) in an upper level CS course between two universities, University of North Carolina and North Carolina State University, found no statistically significant productivity or quality differences between geographically distributed programming teams and in person teams [2]. A more recent work has studied the effects of leadership style and “pair jelling” in a remote and part-time educational setting, with a class of size 10, but did not investigate remote pair programming’s effect on grades or student outcomes [1]. Several recent small-scale studies have weighed in on remote pair programming. One, limited to six pairs, found that women use more non-verbal cues and prefer co-located over remote pair programming to feel connected [11]. In another, with four participants paired by a research team according to perceived confidence and experience, remote pair programming was “associated positively with feelings about communication skills and team-working, which may enhance employability” [10]. However, no quantitative work on the effects of pair programming was performed. In general, it has been suggested that much work remains to establish the effectiveness of remote pair programming on student outcomes, especially since it is increasing in utilization in both academia and industry [9].

3 COURSE DESCRIPTION

Data Structures is a 400-level course offered in the Department of Computer Science and Engineering at a large, public university. The course textbook was Data Structures and Algorithms in C++ (2nd Edition) by Goodrich, Tamassia, and Mount. Course materials were taught primarily through live coding, lectures, and small group discussion. Students received 150 minutes of instruction per week, with access to the professor, teaching assistants, and peer leaders outside of contact hours. Data Structures was offered in person, with code and lecture recordings available for those who were unable to attend due to quarantine or other health-related reasons.

During the fall semester two identical sections of Data Structures were taught by the first author, one of which utilized pair programming (section 1) and a control group which did not (section 2). The pair programming section had 62 students (20 of whom were women) who were randomly assigned the same pair for all four graded assignments. The second section had 45 students (4 of whom were women) who completed each assignment individually.

During the second week of the semester, before any programming projects were assigned, the instructor taught object-oriented design concepts through a week-long live coding project. In this way, a form of pair programming was demonstrated to students, with the instructor as the driver and the class collectively as the navigator. In pair programming, the driver types the code while the navigator develops or modifies the flow of logic, suggests corrective, perfective, or refactorative maintenance, and encourages. Both the driver and the navigator participate real-time by watching the same screen, in person or remotely via screen sharing. Prior to beginning the first assignment, students in section 1 were instructed to read a document on pair programming best practices [18].

4 METHODS

4.1 Research Design and Participants

Participants included all Data Structures students. Data was collected from student records, including student examination scores and assignment completion. In addition, we conducted a pre- and post-course survey asking students to self-evaluate the effectiveness of active learning practices (pair programming in detail) and to report on confidence, persistence in the major, and preference. The 4-minute surveys were voluntary and were sent by another instructor to mitigate perceived coercion. The survey instruments were piloted before the semester. Each survey had an 8-day window for completion, with students receiving extra credit on the final exam for successful completion of both surveys. Alternative means of obtaining equivalent extra credit was provided. The pre-course survey had a completion rate of 87% and the post-course survey had a completion rate of 92%.

4.2 Measures

Students completed the first survey (described above) at the beginning of the semester to gather demographic information and capture attitudes toward computer science. At the end of the semester students completed the second survey which included additional questions related to their experience in the course. The pair programming section was also asked questions related to their experiences using pair programming.

Students were given 4 programming assignments (weighted 40%) and 3 exams (weighted 60%) throughout the semester, with other practice assignments not counting toward final course grade. All
exams were completed individually. A final score for the class was given based on a composite score of all course work and exams.

4.3 Data Analysis
The primary outcome explored was students’ final score in the class. This score was compared between the pair programming section and the section working individually using an independent samples t-test. Separate repeated measures ANOVA models were estimated to explore exam scores over time, and scores on assignments over time across the two groups. Differences found were further explored using post hoc mean comparisons. The type I error rate in post hoc analyses were adjusted using Bonferroni’s adjustment. Additionally, gender differences and differences between pairs working remotely versus in person were explored.

5 RESULTS
A total of 88 students completed both surveys: 64 male students and 24 female students. Pair programming (n=56) and individual sections (n=32) were similar at both at baseline and end of the term with respect to responses given on the survey with the exception of gender. There were more women in the pair programming section than in the individual programming section (20 versus 4).

5.1 Individual vs. Pair Programming
In Table 1, repeated measures ANOVA on programming assignments showed both a within-subject (p = .003) and a between subject effect (p = .001). The pair programming group consistently scored higher than the individuals over time, with the difference ranging from 8–14% (Figure 1). Additionally, the pair programming group scored nearly perfectly on assignments 3 and 4.

Repeated measures ANOVA on exam score showed a within-subject effect (p < .001) but no between subject effect. The pair programming group scored higher than the individual programming group as the semester progressed (Figure 2). Final scores revealed a 7% higher score among students in the pair programming group (87.4 vs. 80.4; p = .013) compared to students in the individual programming group (Table 1).

5.2 In Person vs. Remote
Students in the pair programming group were asked to respond to the statement “my pair programming partner and I always worked remotely” on the end of semester survey. Overall, 34 (59%) strongly agreed, 21 (36%) somewhat agreed, 2 (3%) somewhat disagreed, and 1 (2%) strongly disagreed. It appears one pair had a student who
Table 1: Effect of Pair Programming on Scores

| Section        | Pair Mean (SD) | Individual Mean (SD) | Test           | Effect          | P-value |
|----------------|----------------|----------------------|----------------|-----------------|---------|
| Assignment 1   | 91.3 (19.3)    | 82.7 (32.1)          | RM-ANOVA Group | 0.001           |
| Assignment 2   | 88.9 (27.6)    | 75.1 (34.9)          | Time           | 0.003           |
| Assignment 3   | 99.8 (1.1)     | 85.8 (31.6)          | Group-Time     | 0.733           |
| Assignment 4   | 97.8 (13.0)    | 87.3 (32.5)          |                |                 |
| Exam 1         | 88.2 (12.3)    | 90.0 (8.7)           | RM-ANOVA Group | 0.37            |
| Exam 2         | 78.1 (23.0)    | 74.2 (20.9)          | Time           | <.001           |
| Exam 3         | 75.6 (17.4)    | 70.3 (22.6)          | Group-Time     | 0.306           |
| Final Score    | 87.4 (9.7)     | 80.4 (16.8)          | T-test         | 0.013           |
| Final Score (Women) | 83.0 (10.4) | 71.4 (21.0)          | T-test         | 0.091           |

Figure 4: Pair versus Individual Confidence Asking Questions in Class

somewhat agreed with the statement, while the other somewhat disagreed. Ultimately, while the course itself was offered in person, most students chose to pair program remotely. While we did not ask the students why they chose to do so, we suspect it was due to convenience (several students mentioned something similar to struggling with finding a common time to meet) or safety concerns (e.g., to protect against the transmission of Covid-19).

Programming assignments and final scores were compared between those who agreed or strongly agreed to working remotely and those who disagreed or strongly disagreed (i.e., worked in person). An interesting trend emerged (Table 2). The T-tests showed no statistically significant difference between both groups on any assessment, except for assignment 4, in which the group that worked remotely did better (by 2.9 points) than those that worked in person ($p = .012$). In addition, the actual difference between remote and in person groups narrowed over time, suggesting that those who pair programmed remotely found a routine that worked as the semester progressed. This means that no adverse effects of remote pair programming on student outcomes were observed in this study.

5.3 Gender Differences

Women in the pair programming group scored almost 12% higher than women in the individual programming group on their final score in the course. While this difference did not achieve statistical significance ($p = .091$) due to the low sample size of women in the individual (non-pair) programming group, a 12% difference is practically significant in that it could result in a difference of 2 letter grades (e.g., from a 79 to a 91).

There were also several gender differences in responses on the end of semester survey from students in the pair programming section. First, women reported an increase in comfort asking questions in class, when compared to men (Figure 5). At the beginning of the semester, only 15% strongly agreed with the statement, “I am comfortable asking questions in class”. At the end of the semester, 40% of women students agreed with the statement, and fewer women than men disagreed with the statement. This is important, because it is an indicator of confidence and implies the ability to handle graceful failure in front of peers. By participating in classroom discussion, there is greater opportunity for learning.

Both men and women saw a similar decrease in confidence in their programming abilities by the end of the semester (a reduction of 10% in "strongly agree", Figure 6). In contrast to a first year course, where many students are programming for the first time and seeing marked strides in learning a new language, students in a CS-3 course like Data Structures have prior experience, but are then exposed to significantly difficult assignments. The learning curve is much higher. Thus, it is not surprising that some students adjusted their confidence after experiencing the rigor of solving large, real-world problems.

Table 2: Effect of Pair Programming Remotely on Scores

| Section        | Remote Mean (SD) | In Person Mean (SD) | Test   | P-value |
|----------------|------------------|---------------------|--------|---------|
| Assignment 1   | 91.8 (19.0)      | 100 (0.0)           | T-test | 0.463   |
| Assignment 2   | 88.3 (28.7)      | 95.8 (7.2)          | T-test | 0.655   |
| Assignment 3   | 99.8 (1.1)       | 100 (0.0)           | T-test | 0.725   |
| Assignment 4   | 99.6 (1.9)       | 96.7 (2.9)          | T-test | 0.012   |
| Final Score    | 87.9 (9.3)       | 89.6 (6.9)          | T-test | 0.768   |
No male student indicated that he had a definitive plan to pursue a career outside of computer science (Figure 7). Some women, however, noted definitively that they did not intend to continue work in computer science. A representative comment that explains the sentiment behind taking Data Structures, but not desiring to continue in the field is the following from a woman in the first section: “I feel as though I am a people person and I would not enjoy sitting behind a computer computing/programming all day. I’d rather sell the product [of] the computation to people, but have the understanding of what it is doing.”

Finally, when asked to respond to the statement “I prefer pair programming to working individually” women had more extreme responses than men. Almost one-quarter (23.8%) of female students strongly agreed compared to 8.1% of male students. Similarly, one-third of women (33.3%) strongly disagreed compared to 18.9% of men. The differences in responses to this question were marginally significant with p = .052.

6 DISCUSSION

Over the semester, we noted several observed benefits from the instructor and TA perspective. First, pair programming promotes social intelligence and teamwork capability, two important skills for programmers today. Pair programming more closely models work in industry, where most students will work in teams. While there were a few groups with issues (mainly related to a failure to properly plan ahead for meetings), pair programming afforded the opportunity for students to communicate and resolve differences. Many students mentioned in the survey enjoying this aspect of pair programming. Particularly as our institution shifted to offering in person courses in the wake of Covid-19, students were eager for interaction and social benefits provided by peer review [6].

Pair programming may reduce instances of plagiarism and promote academic integrity because a student must 1) decide to cheat and 2) convince the pair programming partner to become complicit. The social stigma against suggesting cheating is a useful barrier. In addition, if a pair programming group is comprised of at least one student who plans and works ahead, then it can reduce procrastination which can lead to cheating out of desperation because of an impending deadline.

We also observed that pair programming reduces the number of questions asked on assignments, suggesting that students successfully use one another as a resource. Each Data Structures course had a separate Piazza page (crowd-sourced communication tool for professors, TAs, and even other students to provide clarification or help to students’ questions). Students were encouraged to post there first for a faster response (both sections had a 9-minute average response time at the end of the semester). In the Data Structures 1 (with pair programming) students posted an average of 1.05 questions per student, while students in Data Structures 2 (without pair programming) posted an average of 1.72 questions per student. This suggests another intangible benefit to pair programming: that students can, and will, use their partner as a primary resource for clarifying questions.

Finally, several other active learning techniques were utilized in addition to pair programming on assignments in both sections of Data Structures: in-class small group discussion questions modeled after peer instruction [13], peer leaders [4], and live coding [15]. Figure 8 shows how helpful the students found each active learning technique in the post-course survey. The men appeared to prefer peer leaders and live coding, while the women preferred the assignments and live coding examples. In-class group discussion questions appears to be the least helpful to all students. Note that this does not mean the practice is not beneficial to students, as perceived helpfulness is not always correlated to actual helpfulness.
7 CONCLUSION AND FUTURE WORK

Pair programming provides an advantage over individual coding in overall performance in the classroom, even when pairs collaborate remotely. This was demonstrated by the pair programming groups’ better performance on exams and on their final course score. The exam score differences between the two groups appeared to increase over time. Pair programming also demonstrated an advantage on programming assignments. Women in the pair programming group had an almost 12 percentage point advantage over their individually programming counterparts. They also experienced an increase in comfort asking questions in class over the course of the semester to more closely match that of the men. Thus, we have found on several fronts that pair programming is one technique that can be utilized to support the goal of increasing the participation of women in STEM professions.

For future work, we intend to 1) delve more deeply into students who do well on paired assignments, but poorly on individual exams. Such students may simply do better on programming projects rather than conceptual questions, or it may be a result of team breakdown. Thus, we further plan to 2) study the optimal makeup of teams, based on gender, prior ability, and working styles (e.g., supporter versus leader or early submitter versus late submitter). We will also study what makes an ideal pair programming partner and analyze the results with a natural language processing algorithm to identify common trends (aspects with correlated sentiment) [3]. In doing so, we will better understand the team dynamics and optimal pairing techniques to allow computer science students to realize the full benefits of pair programming.

ACKNOWLEDGMENTS

The authors are grateful to Dr. Jing Wang for her support. This material is based upon work supported by USF STEER, USF CITL, and the National Center for Women & Information Technology. We would also like to thank the reviewers for their helpful comments and suggestions.

REFERENCES

[1] Adeola Adeluyi, Michel Wermelinger, Karen Kear, and Jon Rosewell. 2021. Investigating Remote Pair Programming In Part-Time Distance Education. In United Kingdom and Ireland Computing Education Research conference. 1–7.

[2] Prashant Baheti, Edward Gehringer, and David Stotts. 2002. Exploring the efficacy of distributed pair programming. In Conference on Extreme Programming and Agile Methods. Springer, 208–220.

[3] Zachariah Beasley, Les A Piagel, and Paul Rosen. 2021. Polarity in the Classroom: A Case Study Leveraging Peer Sentiment Toward Scalable Assessment. IEEE Transactions on Learning Technologies (2021).

[4] Bethany Bowling. 2015. Professionalizing the role of peer leaders in STEM. Journal of STEM Education 16, 2 (2015).

[5] Nicholas A Bowman, Lindsay Jarrett, KC Culver, and Alberto Maria Segre. 2019. How Prior Programming Experience Affects Students’ Pair Programming Experiences and Outcomes. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education. 170–175.

[6] Mehmet Celepikolu and Kristy Elizabeth Boyer. 2018. Thematic analysis of students’ reflections on pair programming in cs1. In Proceedings of the 46th ACM technical symposium on computer science education. 771–776.

[7] Aaron Clark and Jeffrey R Raker. 2020. Peer-Leaders’ Perceived Roles: An Explanatory Study in a Postsecondary Organic Chemistry Course. International Journal of Teaching and Learning in Higher Education 32, 2 (2020), 180–189.

[8] Alistair Cockburn and Laurie Williams. 2000. The costs and benefits of pair programming. Extreme programming examined 8 (2000), 223–247.

[9] Bernardo José da Silva Estacio and Rafael Prikudzicki. 2015. Distributed pair programming: A systematic literature review. Information and Software Technology 63 (2015), 1–10.

[10] Janet Hughes, Ann Walsh, Bobby Law, and Brendan Murphy. 2020. Remote pair programming. In 12th International Conference on Computer Supported Education. SciTePress, 476–483.

[11] Sandeep Kaur Kuttal, Kevin Gerstner, and Alexandra Bejarano. 2019. Remote pair programming in online cs education: Investigating through a gender lens. In 2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE, 75–85.

[12] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. In Proceedings of the 33rd SIGCSE technical symposium on Computer science education. 38–42.

[13] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving fail rates using peer instruction: a study of four computer science courses. In Proceedings of the 44th ACM technical symposium on Computer science education. 177–182.

[14] PETER ROBE and SANDEEP KAUR KUTTAL. 2021. Designing PairBuddy--A Conversational Agent for Pair Programming. (2021).

[15] Marc J Rubin. 2013. The effectiveness of live-coding to teach introductory programming. In Proceeding of the 44th ACM technical symposium on Computer science education. 651–656.

[16] Karthikeyan Umapathy and Albert D Ritzhaupt. 2017. A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. ACM Transactions on Computing Education (TOCE) 17, 4 (2017), 1–13.

[17] Laurie Williams and Richard I Upchurch. 2001. In support of student pair-programming. ACM SIGCSE Bulletin 33, 1 (2001), 327–331.

[18] Laurie A Williams and Robert R Kessler. 2000. All I really need to know about pair programming I learned in kindergarten. Comm. ACM 43, 5 (2000), 108–114.

[19] Kimberly Michelle Ying, Lydia G Pezzullo, Mohona Ahmed, Kassandra Crompton, Jeremiah Blanchard, and Kristy Elizabeth Boyer. 2019. In their own words: Gender differences in student perceptions of pair programming. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education. 1053–1059.