Strategies to Optimize Student Success in Pair Programming Teams

Dr. Ayesha Johnson, University of South Florida, College of Nursing

I am an assistant professor of statistics in the College of Nursing at the University of South Florida. My research interests include educational methods, and health equity. I have experience in data analysis for various types of research designs.

Dr. Zachariah J Beasley P.E., University of South Florida

Dr. Zachariah Beasley received his Ph.D. in Computer Science and Engineering from the University of South Florida with a focus on sentiment analysis in peer review. He is the first author of nine peer-reviewed papers and a reviewer of three software engineering and natural language processing textbooks. Dr. Beasley has received the ASEE State of Engineering Education in 25 Years Award and USF Spirit of Innovation Award. He plays the guitar at his church and has spent five summers as a volunteer English teacher in Taiwan. Dr. Beasley joined the University of South Florida in August 2020 as an Assistant Professor of Instruction and is a USF STEER STEM Scholar.

©American Society for Engineering Education, 2023

Strategies to Optimize Student Success in Pair Programming Teams

1. Introduction

Pair programming is a software development paradigm used both in industry and in the classroom to increase productivity and reduce defects [11, 27]. In pair programming, two students share a single screen (virtually or in person) while working on a project synchronously. One student uses the keyboard to write the code (the driver), while another observes, offering feedback, and suggesting alternate courses of action (the navigator). Pair programming has been highlighted in the literature as an active learning technique that benefits computer science (CS) students in several ways: it allows “continuous review" where defects are corrected as they arise, it increases confidence in the final product, and it is “40-50% faster than programming alone" [20]. If done properly, students experience an improvement in “programming assignment grades, exam scores, and persistence in computer programming courses" [25]. There are several distinct social benefits for students when collaborating with a peer. Not only does it more closely model the post-university software development work environment, where team projects are normative, but it allows students to help one another without risking academic integrity violations. Finally, it provides an opportunity to improve communication techniques and can inspire healthy social interaction in a post-pandemic era [27].

For its many benefits, the full potential of pair programming is undercut when pair breakdown occurs. The optimal pairing of students to alleviate pair breakdown is still an open and elusive question. Researchers have found a net negative effect in effort spent on the assignment, balance of workload, splitting time as the driver, understanding of lab concepts, and interest in continuing in computer science when a student with less experience is paired with a stronger partner (i.e., when programming ability differs significantly) [7]. Most of the literature examines pair programming in introductory courses which may contribute to this effect, since students can have significantly different programming experience in high school. Ultimately, there is a need to identify and examine exactly which student factors correlate to a group’s success and mitigate pair breakdown so that the full benefits of pair programming might be realized. Our work seeks to answer the question what factors promote success in pair programming? We analyze key factors—gender, prior programming experience, confidence in programming, as well as preferences toward deadlines, communication, and leadership. We then provide several best practice suggestions toward the optimization of pair programming.

2. Related Work/Background

Researchers have generally assessed pair programming to be positive for both in person [11, 20, 25, 27] and remote [1, 3, 5] modalities. In one meta-analysis of 18 studies, positive effects found included decreased time spent on low-complexity programming projects and increased quality of code for high-complexity programming projects [15]. However, there is reason to be cautious in

thinking of out-of-the-box pair programming as a panacea for quickly scaffolding students' competency in programming projects. Paired students' ability to work together is key. Optimizing matching of individuals (e.g., social matching or social expert recommender systems) has a wide variety of applications across academia [30], industry [17, 19], online dating sites [8, 13, 31], and social networks in general [16, 18, 24].

Specifically in the CS education community, there is little agreement on which factors should be utilized to pair students. One often-mentioned factor is the difference in prior programming experience. A large-scale study conducted in computer science, humanities, and information science courses at the University of Iowa over two years noted 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" [7]. Another study analyzed a post-course reflection essay for sentiment towards pair programming. In the "social" dimension they found that "students also reported that their partner can define the success of the pair programming activity: too much difference in knowledge level demotivates high-achieving students by increasing their workload, distracting them from the activity and making them slower, while low-achieving students cannot keep up with the high-achieving students' pace and feel marginalized in the problem-solving activity. This may be why, with few exceptions, most students preferred to work with a similarly skilled partner" [9]. Thus, they point to prior programming experience as an important factor for a student enjoying, and benefiting from pair programming. Others have discovered a similar result when analyzing genders separately. In one such study, researchers found that women reported a feeling of "burdening their partner" when they were randomly paired with a partner who had greater programming experience, although they noted several benefits as a whole to the practice, finding it "improves understanding" and is convenient to have "someone [with whom] to ask questions and discuss ideas" before approaching a TA [29].

Prior programming experience is not the only factor researchers point to when analyzing team success. A literature review on distributed pair programming (DPP) identified a need to "explore the effects of coordination, communication and cultural diversity in DPP" [12]. This is an important finding since an increasing number of students are voluntarily pair programming remotely (including 69% of the students in our study). One small scale study that investigated remote pair "jelling" (ability to work well together) identified communication and leadership style as factors that had a greater effect than prior programming ability, however, the study did not report statistical significance due to only having 5 pair programming groups [1]. Another study drilled down to examine communication via written text (in Slack and GroupMe) between freshman Honors College group members using several features and machine learning algorithms. They analyzed potential determining factors between high- and low- performing

students, finding that sentence length and the number of words (rather than tone, large words, or analytical communication) best predicted performance [2]. Still others have found shared passion to be a key trait in enabling “pro-amateurs" (those with moderate ability) to effectively work on a team and recommend a dating-app-like process to match those with shared passion in the game development community [14].

Finally, a lack of clear agreement on the key factors as well as an absence of best practice recommendations in assigning programming partners due to the complexity of studying partner interaction has led some to propose completely automating the pair programming partner. One such study imagines, but did not implement, a conversational agent that adapts to the user’s skill level to avoid altogether the pairing of partners with different prior programming experience [22]. While this would mitigate conflicting schedules, communication issues, or interpersonal conflict, it is possible that removing the human component of pair programming would have some deleterious effect.

3. Methods

This study seeks to identify the factors that promote success in pair programming. Data from students in an upper-level computer science class was analyzed using linear models.

3.1 Course Description

Data Structures is a sophomore and junior-level course offered by the Department of Computer Science and Engineering at a large, public university in the southeast United States. The course is the last in a chain of three core courses required for a bachelor’s degree in computer science or computer engineering. Data Structures and Algorithms in C++ (2nd Edition) by Goodrich, Tamassia, and Mount was the primary textbook for the course. In addition to utilizing traditional lectures, Data Structures was recently redesigned to include several active learning techniques: live coding [23], small group discussion questions that modeled peer instruction [21], proactive (intrusive) advising [26], and pair programming. Students had access to several teaching assistants and peer leaders [10] outside of class. Data Structures was offered in person, with course material available on the Canvas learning management system.

During the spring of 2022, two sections of Data Structures were taught by the second author, both of which utilized pair programming. In pair programming, the driver types the code while the navigator contributes to the flow of logic and suggests corrective, perfective, or refactorative maintenance. Both the driver and the navigator participate in real-time by utilizing the same screen. This may occur either in person, or remotely via screen sharing; students were allowed to choose the modality of their preference. The first section of Data Structures had 79 students, while the second section had 59. Students were assigned to pair programming groups based on

factors gathered in a pre-course survey. Details on the process to create pairs are provided in subsection 3.3.

Students were provided instruction on how to appropriately pair program via three different methods before any projects were assigned. First, the instructor described in class how to appropriately pair program and the benefits students would receive from following the process correctly. Second, the instructor took the second week of class to cover 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. Finally, prior to beginning the first assignment, students in both sections were instructed to read a document on pair programming best practices [28] as suggested by [25] in their meta-analysis of 18 studies of pair programming. Approximately 41% of Data Structures students had experience with pair programming in a prior course.

3.2 Participants and Research Design

Participants included all Data Structures students. In total, there were 104 (79%) male students and 26 (20%) female students across both sections. The average number of years of prior programming experience was 2.89 (standard deviation of 1.52) with a maximum of 8 and a minimum of 0. Primary outcomes examined were students' examination and assignment scores. Pre- and post-course surveys were conducted which asked students to report their individual preferences on factors potentially important to pair success, and to self-evaluate the effectiveness of active learning practices (pair programming in detail). Factors were rated on a 4-point Likert scale with ratings of "Strongly agree", "Somewhat agree", "Somewhat disagree", and "Strongly disagree". The 4-minute surveys were voluntary and were sent by another instructor to mitigate any 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 were provided. The pre-course survey had a completion rate of 100% and the post-course survey had a completion rate of 93%. This study was approved by the university's IRB (IRB Study #: STUDY002899).

3.3 Pairing Students

Prior to completing any course assignments, students filled out the pre-course survey described above that collected several factors potentially relevant to a pair programming group's success. When creating pairs, we focused on deadlines, communication, and leadership style. Student responses were grouped by whether they were similar (either strongly or somewhat) or different for a statement. In total, 116 students were similar, and 23 students were different for the statement "I prefer to work far in advance of deadlines", while 118 students were similar and 21 were different for the statement "I prefer a high level of interaction/communication in my work

environment." Finally, 76 students were similar, and 63 students were different for the statement "I prefer to support someone else, rather than to take the lead on projects." The finding on the last factor was rather surprising, as we expected a majority of computer science students to prefer taking the lead on coding projects. Results from the survey were used to shuffle student pairings until a balance between the three primary factors was achieved without distributing pairs across course sections (Table 1). Since students mostly preferred to work far in advance of deadlines and to have a high level of communication, fewer pairs had different answers to those questions. Leading vs. supporting had a better balance, with closer to half of the class answering each way. For example, in the leadership category approximately 52% of pairs in section 1 were similar, while 55% of pairs in section 2 were similar. Student demographics such as age and sex were not used as pairing factors.

Table 1. Distribution of Primary Pairing Factors by Section

	Section 1		**Section 2**	
	Similar	**Different**	**Similar**	**Different**
Deadlines	18	9	26	12
Communication	18	9	29	9
Leadership	14	13	21	17

3.4 Primary Measures

Students were given 4 programming project assignments to work on in pairs (weighted 40% of their final grade), and 3 exams to complete individually (weighted 60% of their final grade). There were two small-scale practice assignments that did not count toward a student's final grade that were submitted individually to ensure that every student had the programming language and environment working on their computer. Each student's final score was based on all coursework and exams.

Secondly, students completed two surveys during the course (subsection 3.1). The first survey was completed at the beginning of the semester to gather baseline and demographic data and to assign pairs. Students completed the second survey at the end of the semester which gathered additional data on students' experience in the course, attitude toward the various active learning techniques, and detailed comments toward pair programming.

3.5 Data Analysis & Definitions

Assignment score was explored as pairs, while exam score was explored individually. Factors were analyzed as the same or different depending on whether both students had the same response to each of the factors: gender, programming experience, confidence in programming skills, working with respect to deadlines, communication style, or preferring others to lead. Programming experience was defined as the same if the difference in students' programming

experience did not exceed 2 years. All factors except gender and programming experience were also analyzed as being similar if both students had the same or similar responses. Responses were considered to be the same if both students in the pair had the same response. Responses were considered to be similar if both students expressed a positive sentiment (strongly / somewhat agree), or both students expressed a negative sentiment (strongly / somewhat disagree).

Table 2 shows a summary of the pairs' responses with the additional factors for analysis once the data had been cleaned (e.g., groups of 3 and students who dropped the course were removed). To determine which factors were most important in facilitating pairs working well on assignments, backward stepwise linear regression models were estimated for assignment scores (see subsection 3.5). Exam scores were explored in the same way to elucidate if coding in pairs also impacted individual work. Post-hoc analyses were conducted to determine if factors were different for male and female students, and to identify the impact of student perceptions.

Table 2. Summary of Sameness / Similarity in Pairs

N = 65	**Same n (%)**	**Similar n (%)**	**Different n (%)**
Gender	44 (68)		21 (32)
Programming Experience	15 (23)		50 (77)
Confidence	28 (43)	50 (77)	15 (23)
Deadlines	31 (48)	44 (68)	21 (32)
Communication	24 (37)	47 (72)	18 (28)
Leadership	27 (42)	35 (54)	30 (46)

3.6 Analysis: Linear Regression

Linear Regression estimates a linear relationship between a dependent (outcome) variable and a set of predictor (explanatory) variables. The line estimated is the least squares regression line, which minimizes the total error, or the difference between the actual value of the outcome and the value estimated by the line. The equation of the line is given by:

$$Y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

With the following terms:

Y: The mean value of the outcome for a given set of values of x,

n: The number of predictor variables,

β_0: The mean value of the outcome when all x_i = 0,

β_i: The increase in the mean value of the outcome when x_i changes from 0 (difference in preference) to 1 (similar or same preference).

To find the best model to explain the outcome, we chose to use backward elimination. In this process, all predictors under investigation are entered into the regression model. Through a process of elimination, the least significant predictors are removed until a pre-specified stopping rule is reached. For our model, we used a stopping rule of α=.15. When a parameter is estimated, there is variance in the estimate due to sampling variation. This leads to having a confidence interval, or a range of values for the estimated parameter. Typically set at 95%, a confidence interval provides a range of values within which there is 95% confidence that the true value of the parameter falls. For this paper, a p–value of .05 was considered statistically significant. Factors with p–values between .05 and .10 are discussed. However, suggestions for best practices are solely based on factors which are statistically significant.

4. Results

In total, there were 104 (79%) male students and 26 (20%) female students across both sections. The average number of years of prior programming experience was 2.89 (standard deviation of 1.52) with a maximum of 8 and a minimum of 0. Table 3 shows the complete results of the regression models predicting assignment and exam scores. Preliminary analysis (conducted before exam 3 results were finalized) indicated the predictors shown in Models 1 (assignments) and 3 (exams). These models were then run on the final data. Model 1 can be written as:

Mean Assignment Score = 96.7 - 6.9 (deadlines) - 7.3 (comm style) + 7.3 (prog conf)

This indicates that the mean assignment score for students with different deadline preference, communication style preference, and programming confidence is 96.7%. The difference between the mean scores of pairs who have similar communication style preference and those who do not is 7.3%, with similar communication style preferences having a lower score. We discuss and interpret each model separately in the following sections.

Table 3. Regression Models Predicting Assignment and Exam Scores

	Coeff. (95% Conf. Int.)	**P-value**
Model 1. Assignments		
Similar: Deadlines	-6.9 (-14.2, 0.3)	0.061
Similar: Communication Style	-7.3 (-14.8, 0.2)	0.056
Similar: Programming Confidence	7.3 (-0.6, 15.1)	0.069
Constant	96.7 (87.4, 106.1)	<.001

Model 2. Assignments (at least 1 female student)		
Similar: Deadlines	-5.8 (-19.2, 7.6)	0.376
Similar: Communication Style	-14.1 (-23.7, -4.5)	0.006
Similar: Programming Confidence	11.0 (0.8, 21.1)	0.036
Constant	96.8 (82.7, 110.9)	<.001
Model 3. Exams		
Same: Gender	4.8 (-0.4, 9.9)	0.069
Same: Deadlines	2.7 (-2.3, 7.7)	0.295
Others leading		
Similar but not the same[a]	9.7 (2.0, 17.3)	0.014
Same[b]	2.5 (-2.8, 7.8)	0.354
Constant	72.4 (66.9, 77.9)	<.001
Model 4. Exams (Male Students)		
Same: Gender	8.3 (1.5, 15.0)	0.017
Same: Deadlines	1.0 (-4.7, 6.7)	0.734
Others leading		
Similar but not the same[a]	11.0 (2.1, 19.9)	0.016
Same[b]	2.3 (-3.8, 8.3)	0.459
Constant	69.7 (62.5, 77.0)	<.001
Model 5. Exams (Female Students)		
Same: Gender	-4.8 (-19.0, 9.5)	0.495
Same: Deadlines	12.7 (1.2, 24.3)	0.032
Others leading		
Similar but not the same[a]	.7 (-15.3, 16.7)	0.926
Same[b]	5.1 (-6.1, 16.2)	0.355
Constant	71.6 (62.1, 81.2)	<.001

[a]Pair responses were both positive or both negative, but not identical

[b]Pair responses were identical

4.1 Model 1. Assignments

Among all pairs, backward selection regression models indicated that similarity with respect to deadlines, communication style, and programming confidence marginally impacted assignment scores ($p > .05$ & $< .10$). Similarity in programming confidence was the only factor that had a net positive effect on assignment score, indicating that students may have overall higher scores when paired according to similar programming confidence, but different preferences regarding deadlines and communication style (e.g., to avoid both students preferring to submit close to the deadline or both students preferring little communication and interaction).

4.2 Model 2. Assignments (at least 1 female student)

In the 26 pairs with at least one female student, those with similar communication style preference scored on average 14.1% lower than those whose styles were different (p=.006) on assignments. Pairs with students who were similarly confident in their programming skills scored on average 11.0% higher on assignments than those whose confidence were different (p=.036).

4.3 Model 3. Exams

Exam scores are an important indicator of the effect of pair programming, as they are individual in nature and thus not directly related to the pairs themselves. When modeling exam scores, models indicated that being in a pair with similar, but not the same, preference toward others leading was significantly associated with exam scores, and those students scored on average 9.7% higher on exams compared to others (p=.014).

4.4 Model 4. Exams (Male Students)

For male students, being in a pair of same gender and having similar (but not the same) preference toward others leading were significantly associated with better exam scores. Male students who were in pairs of the same gender scored on average 8.3% higher on exams (p=.017), and those in pairs where students had similar (but not the same) preference with respect to others leading scored on average 11.0% higher on exams compared to others (p=.016).

4.5 Model 5. Exams (Female Students)

For female student performance on exams, those in pairs with the same preference toward deadlines scored on average 12.7% higher than others (p=.032). A T-test to compare exam scores for male and female students revealed that there was no significant difference in the mean exam scores.

4.6 Pair Programming Survey Results

Most students (62%) strongly agreed or somewhat agreed that pair programming was *helpful*, while 24% strongly disagreed (Figure 1). Most students (75%) strongly agreed or agreed that pair programming was *enjoyable*, while 14% strongly disagreed. Just under half the students (47%) strongly agreed or somewhat agreed that pair programming helped to prepare them for exams, while almost three-quarters (74%) strongly agreed or somewhat agreed that pair programming helped to prepare them for their future career. Most students (69%) always worked remotely, compared to 30% who worked partially or completely in-person. Finally, 74% of students strongly agreed or agreed that they would prefer to self-select their programming partner, although there were a few who wanted to be assigned a partner.

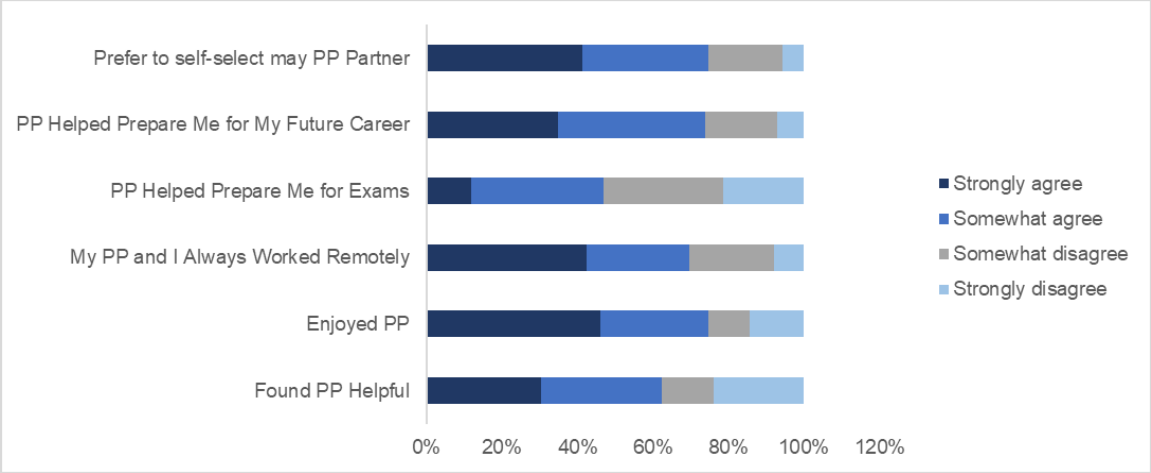

Fig. 1. Pair Programming Survey Results

Pairs were similar with respect to whether they 1) found pair programming helpful, 2) enjoyed pair programming, 3) believed pair programming prepared them for exams, 4) believed pair programming prepared them for their future careers, and 5) would prefer to self-select their pair programming partner. However, there were significant differences in how they rated their partner's programming abilities compared to their own (p=.012). Rating of partner abilities was explored for concordance and strong concordance in which student was perceived to have stronger programming abilities. Ratings had concordance if one student (strongly or somewhat) agreed with the statement, "My pair programming partner's programming abilities were greater than mine" and their partner (strongly or somewhat) disagreed with the statement. I.e., they both agreed on which partner had stronger programming abilities. Ratings had *strong* concordance if one student strongly agreed/disagreed with the statement while their partner strongly disagreed/agreed with it, or if one student somewhat agreed/disagreed while their partner somewhat disagreed/agreed with the statement. I.e., they both perceived one student to be more proficient. 44 (65%) of the pairs had concordance, while 26 (38%) had strong concordance, signifying that students can often identify the partner with more experience. Pairs with concordance and strong concordance scored higher on assignments. However, these differences were not statistically significant.

4.7 Group Issues

There were 7 pairs of the 65 total (11%) that had issues tagged by the instructor over the course of the semester. These issues included lack of communication between students, lack of participating or completing the work as agreed, and scheduling conflicts. A T-test to compare the mean assignment score for these groups compared to the other groups revealed a significant difference. Groups that had issues scored on average 14.2% (p=.007) lower on assignments than groups that did not report any issues. Group issues also impacted students' performance on

exams. Students from pairs reporting group issues scored on average 10.8% (p=.010) lower on their exams compared to students from pairs who did not report any group issues.

5. Discussion

In pairs with at least one female student, those with similar communication style preference scored lower than those whose styles were different on assignments. Pairs with students who were similarly confident in their programming skills scored higher on assignments than those whose confidence were different. Thus, pairing female students based on similar programming confidence (not necessarily programming experience) appears to be an important factor for best performance on assignments. This supports prior findings that students may prefer, and receive benefit from, working with a similarly skilled partner [7, 9].

Modeling exam scores indicated that being in a pair with similar, but not the same, preference toward others leading was significantly associated with exam scores, and those students scored higher on exams compared to others. This suggests that it is better to have a pair where one student strongly desires to lead or support, and where the second student is more flexible. Being in a pair with the same gender marginally impacted ($p > .05$ & $< .10$) exam scores in the positive direction, however, further investigation indicated that this positive effect was driven by the larger number of male student pairs compared to female student pairs. For male students, being in a pair of same gender and having similar (but not the same) preference toward others leading were significantly associated with better exam scores. Male students who were in pairs of 1) the same gender or 2) with similar (but not the same) preference with respect to leading scored higher on exams. Thus, for male student performance on exams, our results suggest that a pair of two male students, with one flexible on who leads, will perform best. For female student performance on exams, those in pairs with the same preference toward deadlines scored higher than others. This suggests that the most important factor identified for female student success in pair programming was for a female student to be paired with another student (male or female) who had the same preference toward submission deadlines.

Most students strongly agreed or somewhat agreed that pair programming was helpful and enjoyable. However, it is noteworthy that students enjoyed pair programming more than they found it helpful. This may point to the social impact of pair programming for students and may have implications for student retention. Students believe that pair programming better prepared them for their future career than it prepared them for exams. Students probably realize that pair programming, as opposed to working individually, more closely resembles the way they will develop software in their future careers. It is encouraging that students notice and appreciate the big picture use of pair programming. Most students always worked remotely, but it is important to make an in-person option available, rather than constraining how students must work together.

Finally, the majority of students preferred to self-select their programming partner, although there were a few who wanted to be assigned a partner.

Group issues were noted in a few pairs. Pairs that experienced group issues scored lower on assignments and exams. While conflict is not unexpected, it is important to note that this conflict not only impacted their group work (assignments), but also their individual work (exams). While assigning pairs in a more data-driven manner can help to mitigate issues, no method will be fail-safe. In fact, there is significant benefit to an instructor taking time to coach students in conflict on how to communicate and resolve issues such as scheduling, flexibility, setting and meeting goals, and accountability. Doing so allows students to raise their EQ (emotional intelligence) and to prepare to handle the inevitable workplace or social conflict. In the experience of the course instructor, it is not always advisable to immediately switch pairs or split a pair when a conflict is first identified, especially if re-assignment may affect other pairs. Instead, effort should be made to allow students to develop these soft skills before more drastic measures (such as allowing a student to work individually or in a group of three) are taken. Another option is to rotate pairs throughout the semester. Aside from the administrative cost this incurs, it may reduce 1) the opportunity for pairs to resolve conflict and 2) the teamwork capabilities of the students. For example, it causes overhead for a student to adjust to a new partner, schedule, communication/leadership style, and prior ability for several projects in a single semester. Still another option is to allow students to self-select their teams (which was the majority preference in our post-course survey responses). This is an approach for conflict mitigation we will consider in future studies.

In the post-course survey, we asked students about their perception of the “ideal" pair programming partner in a free-response question. Responses were coded based on several common themes: communication, even balance of workload, ability of pair programming partner, and time management. Most students mentioned good communication, discussion, or a partner who responds, supporting prior work noting the importance of communication [1, 29]. This was far greater than the percentage that mentioned partner ability, time management, or balancing the workload. Several frequently used words and phrases were patience, flexibility, and motivation. The results of this analysis suggest the obvious: that communication and social interaction are very important to students. Thus, we highlight the importance of students learning and practicing good communication soft skills in the computer science major, rather than simply focusing on technical skills (i.e., programming languages and paradigms).

5.2 Plagiarism

We remain optimistic that violations of academic integrity are reduced through the use of pair programming. When students meet synchronously to pair program, it is difficult, if not impossible, to copy code and present it as one’s own work. There is probably enough social

stigma to prevent one partner from convincing another to plagiarize. Secondly, one student may encourage a less-motivated student to meet earlier and not leave work to the last minute. In cases where plagiarism still occurs, it is likely due to a breakdown of pair communication or planning when one or both students procrastinate on the project. This breakdown of teamwork can be an early indicator that cheating may occur and should be closely monitored. When handling academic integrity cases, plagiarism is easier to identify with pairs of students since conversation and sharing of code are typically well-documented.

6. Conclusion and Future Work

Pair programming is a social and active learning technique that clearly impacts student collaboration and course outcomes. It can confer an advantage over individual coding given that students pair well so that team breakdown does not occur. We analyzed several factors for correlation to student success on both assignments and exams and for male and female students separately: preference toward deadlines, communication, leadership style, and programming confidence. In a free-response section on a post-course survey, students themselves indicated that proper communication is the most important factor they perceive in a high-quality pair programming partner. Our quantitative analysis identified that, for best results on assignment scores, students with similar programming confidence, but different preferences toward deadlines and communication style should be paired. Students paired with similar preference toward others leading performed best on individual assignments (i.e., exams). For male students same-gender pairs with similar preference toward others leading scored higher on exams. Finally, female students pair best when they have similar programming confidence, and when both students have the same preference toward deadlines. Ultimately, we found that proper pairing of students is an important first step toward ensuring student enjoyment of programming and success in the course.

For future work, we intend to analyze the results of the survey free-response questions with a natural language processing algorithm to identify common trends (i.e., aspects with correlated sentiment) [4, 6]. This will allow us to better understand what would increase students' confidence in their programming abilities, their career aspirations, and their like or dislike of pair programming. This will also allow us to compare the sentiment in the responses on the "ideal" vs. the "non-ideal" pair programming partner. We would also like to allow students to select their pair programming partner to determine whether students self-select certain traits organically, and compare those groups with students paired according to our findings in this paper to further validate the effectiveness of our findings. Finally, we are interested in exploring additional factors such as partner responsiveness and reliability, and whether these factors change or improve over time with intervention to further increase the positive impact of pair programming.

References

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

[2] Rohan Ahuja, Daniyal Khan, Danilo Symonette, Shimei Pan, Simon Stacey, and Don Engel. “Towards the Automatic Assessment of Student Teamwork”. In *Companion of the 2020 ACM International Conference on Supporting Group Work*. pp. 143–146.

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

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

[5] Zachariah J Beasley and Ayesha R Johnson. 2022. “The Impact of Remote Pair Programming in an Upper-Level CS Course”. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education* Vol. 1, pp. 235-240.

[6] Zachariah J Beasley and Les A Piegl. “Helps: A domain-specific lexicon for cad peer review”. *In Proceedings. CAD Conf. Exhib*. 2020. pp. 21–25.

[7] Nicholas A Bowman, Lindsay Jarratt, KC Culver, and Alberto Maria Segre. “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*. 2019. pp. 170–175.

[8] John T Cacioppo, Stephanie Cacioppo, Gian C Gonzaga, Elizabeth L Ogburn, and Tyler J VanderWeele. “Marital satisfaction and break-ups differ across on-line and off-line meeting venues”. In *Proceedings of the National Academy of Sciences 110, 25* (2013), pp. 10135–10140.

[9] Mehmet Celepkolu and Kristy Elizabeth Boyer. “Thematic analysis of students’ reflections on pair programming in cs1”. *In Proceedings of the 49th ACM technical symposium on computer science education*. 2018. pp. 771–776.

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

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

[12] Bernardo José da Silva Estácio and Rafael Prikladnicki. 2015. “Distributed pair programming: A systematic literature review”. *Information and Software Technology* 63 (2015), pp. 1–10.

[13] Nicole Ellison, Rebecca Heino, and Jennifer Gibbs. “Managing impressions online: Self-presentation processes in the online dating environment”. *Journal of computer-mediated communication* 11, 2 (2006), pp. 415–441.
[14] Guo Freeman, Nathan McNeese, Jeffrey Bardzell, and Shaowen Bardzell. “’Pro-Amateur’-Driven Technological Innovation: Participation and Challenges in Indie Game Development.” *Proceedings of the ACM on Human-Computer Interaction 4*, GROUP (2020), pp. 1–22.
[15] Jo E Hannay, Tore Dybå, Erik Arisholm, and Dag IK Sjøberg. “The effectiveness of pair programming: A meta-analysis”. *Information and software technology* 51, 7 (2009), pp. 1110–1122.
[16] Henry Kautz, Bart Selman, and Mehul Shah. “Referral Web: combining social networks and collaborative filtering”. *Communications*. ACM 40, 3 (1997), pp. 63–65.
[17] Ching-Yung Lin, Kate Ehrlich, Vicky Griffiths-Fisher, and Christopher Desforges. “Smallblue: People mining for expertise search”. *IEEE MultiMedia* 15, 1 (2008), pp. 78–84.
[18] Julia M Mayer, Starr Roxanne Hiltz, and Quentin Jones. “Making social matching context-aware: Design concepts and open challenges”. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015. pp. 545–554.
[19] David W McDonald and Mark S Ackerman. “Expertise recommender: a flexible recommendation system and architecture”. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. 2000. pp. 231–240.
[20] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. “The effects of pair-programming on performance in an introductory programming course”. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*. 2002. pp. 38–42.
[21] Leo Porter, Cynthia Bailey Lee, and Beth Simon. “Halving fail rates using peer instruction: a study of four computer science courses”. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013. pp. 177–182.
[22] Peter Robe and Sandeep Kaur Kuttal. “Designing PairBuddy–A Conversational Agent for Pair Programming. *ACM Transactions on Computer-Human Interaction (TOCHI)* 29.4 (2022): pp. 1-44.
[23] Marc J Rubin. “The effectiveness of live-coding to teach introductory programming.” In *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013. pp. 651–656.
[24] Loren Terveen and David W McDonald. “Social matching: A framework and research agenda”. *ACM transactions on computer-human interaction (TOCHI)* 12, 3 (2005), pp. 401–434.
[25] Karthikeyan Umapathy and Albert D Ritzhaupt. “A meta-analysis of pair-programming in computer programming courses: Implications for educational practice”. *ACM Transactions on Computing Education (TOCE)* 17, 4 (2017), pp. 1–13.
[26] Jennifer Varney. “Proactive (intrusive) advising”. *Academic Advising Today* 35, 3 (2012), pp. 1–3.

[27] Laurie Williams and Richard L Upchurch. “In support of student pair-programming”. *ACM SIGCSE Bulletin* 33, 1 (2001), 327–331.
[28] Laurie A Williams and Robert R Kessler. “All I really need to know about pair programming I learned in kindergarten”. *ACM Communication* 43, 5 (2000), pp. 108–114.
[29] Kimberly Michelle Ying, Lydia G Pezzullo, Mohona Ahmed, Kassandra Crompton, Jeremiah Blanchard, and Kristy Elizabeth Boyer. “In their own words: Gender differences in student perceptions of pair programming”. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 2019. pp. 1053–1059.
[30] Douglas Zytko and Leanne DeVreugd. “Designing a Social Matching System to Connect Academic Researchers with Local Community Collaborators”. *Proceedings of the ACM on Human-Computer Interaction 3*, GROUP (2019), pp. 1–15.
[31] Doug Zytko, Sukeshini Grandhi, and Quentin Jones. “The Coaches Said... What? Analysis of Online Dating Strategies Recommended by Dating Coaches”. In *Proceedings of the 19th international conference on supporting group work*. 2016. pp. 385–397.