

# Pair Programming: Not for Everyone?

Jacob Somervell

221 Darden Hall, Department of Mathematical Sciences

University of Virginia's College at Wise

Wise, VA 24293

[jps5a@uvawise.edu](mailto:jps5a@uvawise.edu)

*Abstract - Developments in CS1 pedagogy seek to improve the capabilities of students in their understanding of how to write programs. This involves learning how to solve various types of problems through a systematic approach, a challenging task for most students. "Pair programming" as a pedagogical approach, is one in which students help one another learn how to program by working in pairs on programming assignments. We were interested in using this technique for teaching CS1 topics, to assess applicability in our undergraduate Computer Science curriculum. We implemented pair programming in an offering of CS1, parallel to another offering without paired programming, and compared the results of student performance in each class. In addition we asked the students in the paired programming class about their experiences with paired programming, to gain some measure of how the students felt about the approach. The results were unexpected in that the paired programming students did not show increases performance over the other students, using traditional metrics; and, their subjective opinions indicate that the students did not enjoy or like doing paired programming. This was surprising because pair programming has seen tremendous success in recent years. We offer explanations for our results and caution to others pursuing this particular pedagogical technique.*

**Keywords** – computer science instruction, CS1 pedagogy, paired programming

## 1. Introduction

Learning how to program is hard. Learning to program involves more than just learning the syntax and semantics of a new language; it also entails learning critical thinking, problem dissection, and algorithm design and implementation. Imagine how tough it is to learn a new programming language, especially when you have no prior knowledge of programming or problem solving. Now add to that challenge the fact that you must learn all of this by yourself -- no working with others, no sharing code, no discussions; or it might be considered cheating.

The "work alone" mentality has long been rooted in computer science education. Students have to write their own program solutions, otherwise how could the instructor know if the student was learning the material?

The single learner, struggling to grasp foreign concepts handed down through lectures in a classroom, has been the traditional computer science educational approach. Of course, students can be, and are, successful with this method; but some undergraduate programs seek other pedagogical techniques to boost student success. One of the focuses of innovative pedagogical techniques in computer science education is to move away from the "work alone" tradition and include more collaborative activities [3].

Paired programming is a programming technique in which two programmers work together to solve a particular problem [1]. Usually one person is the driver, or coder, and the other is the navigator, or debugger. The idea is that two minds can more effectively handle the challenge of writing a program to solve some problem. While one person types the code, the other can point out deficiencies or problems, or suggest alternative methods.

Recognizing some of the drawbacks of a traditional approach, we were interested in exploring paired programming as an alternative method for teaching CS1 topics. We wanted to see how our students performed with this new technique when compared to our existing traditional method. We used both techniques in parallel offerings of CS1 to assess how well the paired programming technique compared to the traditional "work alone" approach.

We discuss some related work to provide some motivation for the paired programming technique in the next section. The remainder of the document describes the experimental setup, analysis of the results, and discussion of our findings.

## 2. Related Work

Research into paired programming has occurred in both academia and industry. Cockburn and Williams provide compelling evidence of the utility of paired programming in industrial settings [3]. In their work they provide real-world examples of managers and programmers, touting the utility and benefit of paired programming.

Other researchers have investigated using paired programming in software engineering courses [3][8]. These courses are usually junior or senior level courses and the students have experience with programming from the beginning. A good example comes from [1]. In their

work they describe impressive advantages of paired programming over a traditional “work alone” strategy. Students in their software engineering classes had better performance in terms of project grades and overall course grades. In addition, these students indicated they enjoyed the opportunities to work with other students [1].

Other examples of prior research into paired programming come from [2][4][5][7] and from North Carolina State University [8][9][6]. In this collective body of work, the paired programming approach in CS1 offerings was used with about 1200 students. Their results suggest that this technique helps students learn how to program and provides valuable exposure to working with others – a de facto standard in software engineering. Note that none of these works indicate any negative effects on student outcomes.

It seems clear that paired programming should be advantageous for teaching introductory programming topics (CS1). It should boost student performance, encourage participation, and provide valuable “team” experience [2][4][8]. With these advantages in mind, we sought to determine if paired programming would work for our students. Thus, we introduced paired programming in parallel to a traditional offering of CS1 and compared student performance of both, in hopes of seeing how paired programming could help our students succeed with learning to program.

### 3. Experimental Description

We used existing course offerings to investigate the utility and potential success of using paired programming to teach CS1 topics. One course (we will call it PAIRED) consisted of 16 students, whom we randomly paired up at the beginning of the semester. The pairs remained static throughout the course of the semester. The other offering (we will call it CONTROL) had 20 students, with no pairing. Throughout the course of the semester (15 weeks), both offerings had the same content, lectures, instructor, quizzes, tests, and programming assignments. Both courses were even taught in the same room! The only difference between the two courses, other than pairing, was the time of offering. CONTROL was offered on Monday, Wednesday, and Friday at 9:00 AM. PAIRED was offered on Monday, Wednesday, and Friday at 12:00 PM. Note that PAIRED allowed the student pairs to work on the programming assignments together.

Both courses were first courses in programming in C++. Topics included problem solving, algorithm design, and implementation in C++. Programming assignments started out with simple “hello world” type programs and progressively became more challenging as the students learned new techniques; with the most challenging programs coming towards the end of the semester (dealing with records, arrays, pointers, and linked lists).

Grading scales were identical for each class, regardless of use of paired programming, and students in a pair received the same grade on the programming assignments, regardless of the amount of effort put forth by each student. Thus, the paired students had an equal stake in the correctness on the programs submitted by that particular pair. The hope was that both students would try to ensure that they got the highest grade possible for a given programming assignment and would thus work together to complete the assignments.

Data collection consisted of the student grades in each offering, broken down by programming assignment and by test scores, and subjective opinions from the PAIRED students. Since the instructor was the same for both courses, the grading scheme and scales were identical. The instructor randomly graded all programs for a given assignment, so that he/she did not know to which class a particular solution belonged. Students submitted their programming solutions through email.

At the end of the semester, after the final exam, the PAIRED students completed a short survey, probing their opinions on how paired programming worked for them. The survey included five questions, each focused on a different aspect of the course and how paired programming could have impacted performance in those areas. Figure 1 has the list of the questions.

1. I enjoyed working with a partner on the programs.
2. Working with a partner helped me learn how to program.
3. Working with a partner helped me learn the C++ programming language.
4. Working with a partner helped me perform better on the tests and final.
5. I value the experience of working with a partner.

FIGURE 1  
QUESTIONS USED IN THE SUBJECTIVE SURVEY.

Each of the five questions was answered by indicating agreement on a 5-point Likert-type scale, ranging from strongly disagree to strongly agree. A numeric scale was used to facilitate the analysis of the questions (1 = strongly disagree, 5 = strongly agree). In addition to the questions, the students were provided with space to include their comments, for each question. These comments provide extra insight into how the students felt

about paired programming. More on the results from these questionnaires can be found in section 4 and 5.

In line with this setup, and based on strong evidence from prior research, we formulated four hypotheses:

1. PAIRED students would show greater overall performance.
2. PAIRED students would have higher test scores.
3. PAIRED students would have higher program scores.
4. PAIRED students would indicate that working with partners would be beneficial.

We feel these are reasonable expected outcomes, considering the positive results obtained by other researchers.

## 4. Results

Comparing the overall performance of each class, CONTROL performed better than PAIRED. This includes tests, quizzes, programs, and the final exam. Figure 2 shows the overall grade distribution for each section. The average grade for the CONTROL group was C (74) and the average grade for the PAIRED group was a D (68); with no statistical significance between the groups for overall score. Hypothesis 1 is not supported.

### 4.1 Tests

To more accurately assess student comprehension, tests are administered periodically throughout the semester. These tests are designed to gauge student mastery of material covered in lecture and programming assignments. These tests are individual performance tests; i.e. the PAIRED students had to take the tests on their own, just like the CONTROL group; they did not work with their partner on the tests. Comparing the test performance between the groups indicates that the CONTROL group had overall higher scores (average of 67, compared to average of 61), but the difference was not statistically significant. Figure 3 shows the test score comparison. Hypothesis 2 is not supported.

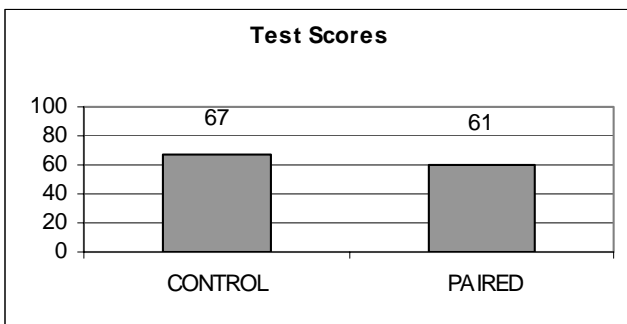


FIGURE 3  
TEST SCORES

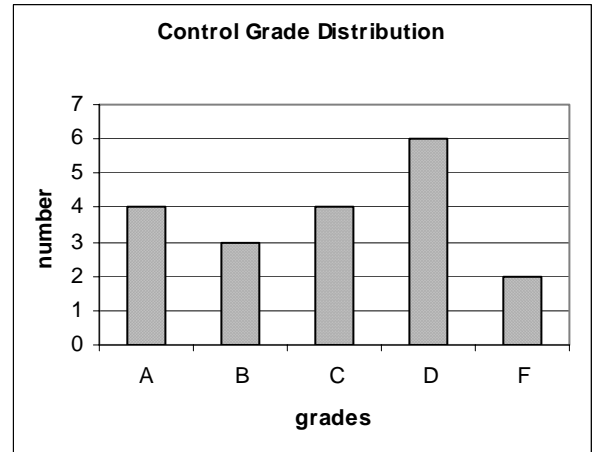
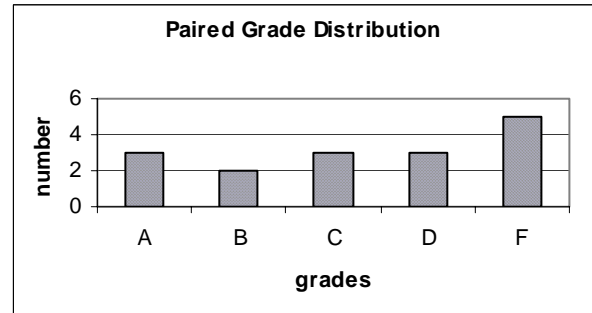


FIGURE 2  
GRADE DISTRIBUTIONS FOR BOTH GROUPS.

### 4.2 Programming

Another major part of the students' grade comes from their performance on the programming assignments. Both groups of students did much better with the programs than on the tests. Still, the CONTROL group performed better on average (87 versus 80), with no statistically significant differences. Figure 4 shows the program scores. Hypothesis 3 is not supported.

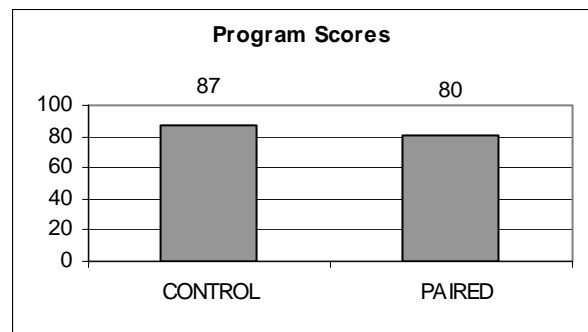


FIGURE 4  
PROGRAM SCORES FOR BOTH GROUPS.

### 4.3 Subjective Opinions

As previously mentioned, we also tried to assess the students' opinions on paired programming. The average

response for each question was tallied. The results of our survey are pictorially portrayed in Figure 5. Note that only the question on whether they enjoyed working with a partner received marks close to the “agree” rating. All of the other marks indicate neutral or “disagree” ratings. Hypothesis 4 is not supported.

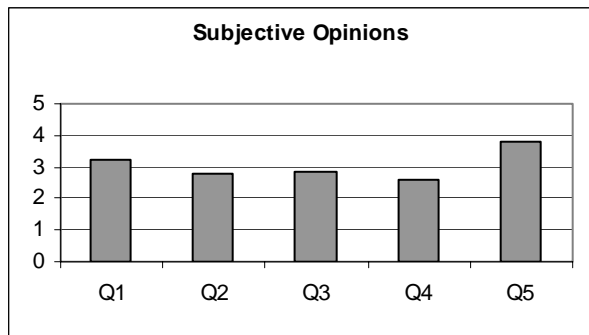


FIGURE 5  
SUBJECTIVE OPINIONS ON THE FIVE QUESTIONS IN FIGURE 1.

## 5. Discussion

None of our hypotheses were supported by our data. What could explain such unexpected results? Other research indicates that paired programming enhances the experience for students. This was not the case for this endeavor. On the contrary, the PAIRED group did not perform as well as the CONTROL group, in any measure we took [keep in mind that there were no statistically significant differences in performance]. Furthermore, the subjective opinions of the students indicate that the paired programming approach we implemented has serious flaws that need to be addressed before further adoption occurs.

Consider the overall performance of the two groups. We originally hypothesized that the PAIRED group would perform better. We felt that the opportunity to work with a partner would provide greater insight into the programming language, and this insight would be apparent in higher overall grades. This was not the case. There was no significant difference between the groups, but it was somewhat discouraging to see that the PAIRED students had lower performance in the course. Similar results were noted for the test scores.

The most surprising result came from analysis of the performance on the programming assignments. It seemed obvious that working with a partner on the programming assignments would invariably lead to better program scores – due to having extra input and feedback on program design, analysis, and coding. We did not see this in our study. The PAIRED group had a lower average program score than the CONTROL group [again with no statistical significance]. This performance indicator was contrary to our original expectations.

The only insight we have to explain these results comes from the comments provided by the students in the PAIRED group, as obtained from the survey. As mentioned in the previous section, we expected positive results from the student opinions of paired programming. Surely students would indicate some level of enjoyment or utility from working with others. As shown in Figure 5, the students indicated that they did not enjoy or benefit from the paired programming approach.

The additional comments provided by the students indicate some problems with working with a partner. For example, several students hinted that they “work better alone” or “learn things independently”. Others suggested that their partners did not contribute to the process:

“I did not like the partner I was working with, he wouldn’t help me.”

“I did all the work myself”

“[it is hard] when your partner doesn’t help you or know anything.”

One or two students indicated that working with partners is, in general, a positive thing; for example:

“A partner is a good thing...just depends on if you get a good partner.”

“I struggle with this stuff. A partner was helpful in many ways.”

While providing some idea as to why the performance indicators were so low, these comments do not completely explain the poor performance. Rather the mixed nature of the comments reflects the different feelings students have about sharing work responsibilities. Something else must be causing the mediocre results.

It seems that extraneous factors could have contributed to the weaker performance of the PAIRED group. Upon closer inspection of the enrollment for each course, it was found that there were three high school students enrolled in the PAIRED grouping. (Part of a local initiative to get students interested in mathematics and science.) This puts their age range from 16-18, while a typical freshman is 18-20. So maturity could have played a part in the students’ abilities to work in a team environment. Maturity level could have also contributed to the lower grades in that the high school students may not have been used to the rigorous requirements of a college level course. This argument is speculation at this point; further investigation through a longitudinal study would be required to fully support this argument.

We still need to explain why our results seem to be in such stark contrast with prior research. Closer inspection of the experimental setup highlights some differences that could explain our results. First, the descriptions of paired programming in [2] have the students paired in a non-

permanent fashion. That is, the pairs were dynamic and changed throughout the semester as the assignments changed. In our setup, we paired the students using random assignment and those pairs remained throughout the semester and through all programming assignments. It is not clear from the literature, nor through our study, what impacts this choice may have on performance. As such, we need to consider this factor as a possible explanation for our results.

Second, the amount of instruction on “how” to do paired programming was minimal. We were simply interested in finding out if working with a partner could improve student performance, without teaching them how to work with a partner. Students were left alone in defining roles and how each student would perform in those roles. This is in contrast to other descriptions of paired programming and could be a reason for our unexpected results. Perhaps not knowing how to work together led to less impressive results for the PAIRED groups.

Regardless, it seems that paired programming may not be as useful as other researchers have suggested. Care must be taken when implementing paired programming – choosing appropriate pairing strategies and teaching the students how to work in pairs seems to be a prerequisite for success. If a program does not have adequate resources for this type of instantiation, other teaching methods should be considered as alternatives to paired programming.

## 6. Conclusions

Our work has probed the use of paired programming in teaching CS1 topics. We fully expected this technique to improve student performance on several traditional measurements (programming scores, test scores, overall grades). However, we found that our implementation of the method did not significantly improve student performance. We also learned that the students did not receive the technique with much enthusiasm. These findings are surprising and lead us to more questions about pair programming and its role in CS1 education.

When one considers the existing body of knowledge on paired programming, one must ask why the results we have reported here are somewhat contradictory. As discussed previously, the setup used in this experiment could have contributed to the weaker results. We see this area as a “next step” in research: we need to focus on testing the implementation of paired programming to see what is most useful for CS1 education.

It should not be overlooked that the PAIRED students, regardless of their performance, have gained valuable experience in teamwork. This value of this experience cannot be adequately measured in a single course, but would require long-term assessment to see if this exposure to team programming contributes to success in a computer science degree program.

We fully expect that paired programming is a viable alternative to lecture-style teaching. We will continue to implement paired programming in CS1 courses, but will do so with different setups (have pairs change over time, define roles more explicitly, etc.) to more fully appreciate what students need and from what they benefit most.

## 7. References

- [1] Berenson, S., Slaten, K., Williams, L., Ho, C. (2004) “Voices of Women in a Software Engineering Course: Reflections on Collaboration.” *ACM Journal of Educational Resources in Computing*, Vol. 4, No. 1, March 2004.
- [2] Bevan, J., Werner, L., McDowell, C. (2002). “Guidelines for the Use of Pair Programming in a Freshman Programming Class”, *Fifteenth Conference on Software Engineering Education and Training (CSEE&T 2002)*.
- [3] Cockburn, A. & Williams, L. (2002). “The Costs and Benefits of Pair Programming” in *Extreme Programming Examined*, Boston: Addison-Wesley.
- [4] McDowell, C., Werner, L., Bullock, H., Fernald, J. (2003). “The impact of pair programming on student performance of computer science related majors.” In *Proceedings of the International Conference on Software Engineering (Portland, OR)*, 602-607.
- [5] McDowell, C., Werner, L., Bullock, H., Fernald, J. (2002). “The Effects of Pair Programming on Performance in an Introductory Programming Course”, *Proceedings of the Conference of the Special Interest Group of Computer Science Educators (SIGCSE 2002)*.
- [6] Nagappan, N. L., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. L., Balik, S. (2003) “Improving the CS1 experience with pair programming.” In *Proceedings of the 34TH SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New York, 359-362.
- [7] Werner, L. Hanks, B. McDowell, C. (2005) “Female computer science students who pair program persist.” *ACM Journal of Educational Resources in Computing*, Vol. 4, No. 1, March 2004.
- [8] Williams, L. A., & Kessler, R. R. (2001, March 2001). “Experimenting with Industry’s “Pair Programming” Model in the Computer Science Classroom.” *Journal of Computer Science Education*, pp. 1-20.
- [9] Williams, L., K. Yang, E. Wiebe, M. Ferzli, C. Miller (2002). “Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations.” *Paper presented at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2002)*. Seattle, WA, November 4-8.