The Impact of Pair Programming on Student Performance, Perception and Persistence

Charlie McDowell and Linda Werner
Computer Science Department
University of California, Santa Cruz
{charlie,linda}@cs.ucsc.edu,

Heather E. Bullock and Julian Fernald
Psychology Department
University of California, Santa Cruz
{hbullock,jfernald}@cats.ucsc.edu

Abstract

This study examined the effectiveness of pair programming in four lecture sections of a large introductory programming course. We were particularly interested in assessing how the use of pair programming affects student performance and decisions to pursue computer science related majors. We found that students who used pair programming produced better programs, were more confident in their solutions, and enjoyed completing the assignments more than students who programmed alone. Moreover, pairing students were significantly more likely than non-pairing students to complete the course, and consequently to pass it. Among those who completed the course, pairers performed as well on the final exam as non-pairers, were significantly more likely to be registered as computer science related majors one year later, and to have taken subsequent programming courses. Our findings suggest that not only does pairing not compromise students' learning, but that it may enhance the quality of their programs and encourage them to pursue computer science degrees.

1. Introduction

In recent years, the growth of extreme programming (XP) has brought considerable attention to collaborative programming. Developed over a fifteen year period by Kent Beck and his colleagues, Ron Jeffries and Ward Cunningham[1], XP is a computer software development approach that credits much of its success to the use of pair programming by all programmers, regardless of experience[2]. The pair programming dimension of XP requires that teams of two programmers work simultaneously on the same design, algorithm, code, or test. Sitting shoulder to shoulder at one computer, one member of the pair is the "designated driver," actively creating code and controlling the keyboard and mouse. The "non-driver" constantly reviews the keyed data in order to identify tactical and strategic deficiencies, including erroneous syntax and logic, misspelling, and implementations that don't map to the design. After a designated period of time, the partners reverse roles. Code produced by only one partner is discarded, or reviewed collaboratively before it is integrated.

Computer science faculty are increasingly experimenting, either formally or informally, with pair programming in the classroom. All of the published studies to date indicate that pair programming has a positive impact on some aspect of student performance or enjoyment, while none have demonstrated that student learning is compromised[2-4]. Nevertheless, many instructors continue to require students to complete programming assignments independently. Presumably, continued reliance on solo programming in academic settings is rooted in instructor concern that one of the partners in a pair will not learn as much as they would if they completed the assignment alone. In the worst case, one member of the pair might do essentially all of the work. Although this would not be "pair programming," it is often difficult, if not impossible, to monitor how students actually spend their programming time and how closely they are following the pairing protocol.

Previously, we reported on preliminary data from two of the four introductory programming sections discussed in this paper[3]. We found that a greater percentage of students who were required to use pair programming made it to the final exam, and that among those who completed the course, pairing and non-pairing students performed equally well on the final exam. In this paper we report on course performance data from all four sections. We also present data about students' confidence in their programming solutions, and their enjoyment of, and satisfaction with the process, as well as subsequent course taking patterns and major declarations.

2. Method

The purpose of this study was to investigate the effects of pair programming on student performance and subsequent pursuit of computer science related degrees among both female and male college students. Participants were 555 students (413 men and 141 women[1]) who attempted an introductory computer programming course,

[1] Gender information was unavailable for one student.

intended primarily for computer science, computer engineering, electrical engineering, and information systems management majors, at the University of California-Santa Cruz during the 2000-01 academic year.

Four sections of the course were offered during the year: one in the Fall quarter, two in Winter, and one during the Spring. One of the principle investigators of this study, Charlie McDowell, taught the Fall and Spring sections of the course. The two Winter sections were taught by UCSC faculty members not associated with this project.

Students enrolled in the Fall and Winter sections were required to complete all assignments using pair programming. Students in the Spring section, on the other hand, were required to complete programming assignments independently. On the first day of class students in the pairing sections were given a brief 15 to 20 minute description of pair programming and instructed to read Williams and Kessler's paper "All I Really Need to Know About Pair Programming I Learned in Kindergarten [5]." As an incentive to read the paper they were told that the first quiz might include a question about it.

Students in the pairing sections submitted a list of three names of potential partners, and partners were assigned based on these preferences. In nearly all instances, students were assigned a partner from their list. Those that stated no preference were randomly assigned a partner. Whenever possible students remained with the same partner throughout the quarter, however, due to schedule changes and drops, a small number of partner reassignments were necessary. As a result of hardships such as heavy work schedules or living far from campus seventeen students across the three pairing sections were permitted to program alone for various reasons. Data from these students was combined with the data from the students in the non-pairing section.

Although each student was assigned to one 90-minute lab time per week, most programming assignments were completed outside of scheduled lab time. The labs functioned primarily as teaching assistant office hours. There were no specific in-lab assignments and attendance was not mandatory.

Students in two of the three pairing classes and in the non-pairing class were required to submit five graded programming assignments. In the remaining pairing class students completed four graded assignments. In all four sections students were also encouraged to complete four additional practice programming assignments that did not contribute directly to their grade. Programming assignments were scored for functionality and readability. Along with each assignment students submitted a log indicating the amount of time they spent on the assignment (pairing students were asked to differentiate between time spent driving, reviewing, and alone), their level of confidence in their solution, how much they enjoyed working on the assignment, and how satisfied they were with the process.

Regardless of whether they completed assignments in pairs, all students took exams independently. The final exam assessed students' knowledge of programming concepts and their ability to write new code. Additionally, we collected information about students SAT scores, the courses they took over the following year, and their major declarations a year after taking the class.

3. Results

An important assumption of this study was that all four sections of the course were similar in terms of students' academic preparation to succeed. In order to test this assumption, one-way ANOVAs by course section on SAT math and verbal scores, and high school GPA (required of most freshmen applicants for admission to UCSC) and transfer GPA (required of all transfer students) were conducted. There were no significant differences between sections on SAT math scores, or on high school or transfer GPA. There was a significant main effect on SAT verbal scores $F(3, 465) = 5.05$, $p<.005$. Tukey follow-up tests revealed that the average SAT-V scores in one of the sections that required pairing was significantly lower than the two other pairing sections ($M = 521$ vs. $M = 559$ and $M = 573$), but not significantly different than the non-pairing section ($M = 547$). Combining the three pairing sections indicated that there was not a significant difference in SAT-V scores between all pairers and non-pairers ($M = 553$ and $M = 547$ respectively). Because the difference was only between pairing sections it seemed acceptable.

3.1. Completion and Pass Rates

Perhaps the most straightforward indicators of student success are completing the course and passing it. For the purpose of this study, we defined *completion* as taking the final exam, and *passing* as earning a "C" or above.

Table 1: Completion and pass rates

	Completion Rates for all Students	Pass Rates for all Students	Completer's Pass Rates
Pairers[2] (N=404)	90.8%	72.3%	79.6%
Non-pairers (N=148)	80.4%	62.8%	78.2%

A comparison of students who used pair programming with those who didn't indicated that pairers were significantly more likely to complete the course (90.8%)

[2] The pairing status of 3 students was unavailable.

than were non-pairers (80.4%), $\chi^2(1) = 11.21$, $p<.001$. Not surprisingly then, pairers were also more likely to pass the course (72.3% vs. 62.8%), $\chi^2(1) = 4.57$, $p <.05$. Among just those who took the final exam, the difference in pass rates between pairing (79.6%) and non-pairing students (78.2%) was not statistically significant (see Table 1).

Williams and Kessler have proposed "pair pressure" as a possible explanation for higher completion rates among paired versus unpaired students[6]. According to Williams and Kessler, students who work in pairs may be more likely to complete programming courses because of the shared responsibility that results from collaborative partnerships. As a consequence, paired students may remain in the class for the sake of their partner. Although this is a plausible explanation, it is not supported by these data. The fact that in our study there was no difference in pass rates between pairers who completed the course and non-pairers who completed the course suggests that it was not simply the case that pairers were more likely to "stick it out," but rather that a larger proportion of paired students were able to master enough of the course material to pass.

Overall the difference between women's and men's completion rates (85.8% and 89.1% respectively) was not statistically significant $\chi^2(1) = 1.10$, $p>.05$. An examination of the effects of pairing on women and men separately indicated that men who paired were significantly more likely to complete the course (91.7%) than men who did not pair (81.5%). On the other hand, there was not a significant difference in completion rates between women who paired and those who did not (88.1% vs. 79.5%). The fact that the 10% difference in completion rates between paired and unpaired men was statistically significant, while the 8% difference in completion rates between paired and unpaired women was not is probably best explained by the substantially smaller number of women (N = 141) than men (N = 413) in the programming classes.

3.2. Course Performance

In addition to completion and pass rates, we were interested in the effects of pairing on course performance. We measured two related, but distinct indicators of course mastery. The first, the quality of the programs that students produce was operationalized as student's normalized average score on the graded programming assignments. The second is the extent to which students are able to apply the concepts covered during the course. We used final exam scores, which all students took independently regardless of whether they paired or not, as the measure of the extent to which they had learned the material. For the following analyses we included only those students who completed the course, defined as taking the final exam (N=486).

Among students who completed the class, a two-way ANOVA (pair type X gender) indicated that those who paired produced significantly better programs (86.6%) than those who worked alone (68.1%), $F(1, 482) =77.42$, $p<.001$. There was no significant gender difference in average programming scores (men's and women's scores were 81.9% and 82.5% respectively), nor was there an interaction between gender and pairing. In other words, pairing was associated with significantly higher scores for both women and men (see Table 2). A two-way (gender X partners' gender) ANOVA on pairer's programming scores indicated that neither pairers' gender nor their partner's gender was related to the quality of the programs they produced.

Table 2: Average programming scores

	Mean	Std. dev.
All pairers	86.6%	14.7
All non-pairers	68.1%	22.4
Paired women	86.9%	16.0
Unpaired women	70.1%	19.7
Paired men	86.5%	14.2
Unpaired men	67.3%	23.3

It may be that the reluctance of some computer science faculty to use pair programming in their classes is due to a concern that at least some students will "earn" grades that predominantly reflect their partner's work. It is possible, for example, that the pairing students in our study earned higher average programming scores simply because weaker students received scores that were primarily due to the work of the stronger student in the pair, thus artificially inflating the average programming scores of the pairers.

Elsewhere we have argued that the very process of working collaboratively enhances the quality of programs that pairs produce[3]. In that paper we compared the two sections of the introductory programming course that were taught by the same instructor, and for which assignments were intentionally designed to be equivalent. We found that the average score on programming assignments of all of the students in the pairing section was significantly higher than the average score of the top 50% (based on final exam scores) of the non-pairing section.

Nevertheless, it is important to understand the effects of pairing on individual student learning. Because all of the students in this study took the final exam independently, we considered final exam scores the best indication of the extent to which students had mastered the course material. A two-way (pairing X gender) ANOVA indicated that scores were not influenced by whether students paired, $F(1, 482) =.02$, $p>.05$. Pairers averaged 75.2% on the final compared with non-pairers' average of 74.4% (see Table 3). This finding strongly suggests that a student's ability to independently apply concepts to novel problems is not compromised by learning to program in pairs. Indeed, considering that a significantly greater percentage of the students who paired took the final, it seems that learning to program in pairs results in mastery for a greater percentage of students.

We did note a gender difference in final exam scores that approached, but did not reach significance, *F* (1, 482) =2.86, p=.09. Men and women scored 76.2% and 71.4% respectively. Analysis of covariance (ANCOVA), using SAT math scores as a covariate (*F*(1, 409)=.43, *p*=.51.) indicated that the marginal gender difference we observed may be the result of differential preparation.

A two-way (gender X partner's gender) ANOVA on pairers exam scores only, revealed a significant main effect of gender on final exam scores, F(1, 362) = 7.46, p<.01. Men who paired averaged 76.7%, while women who paired averaged 70.8% (see Table 3). Again, this difference was not significant when math SAT scores were held constant, F(1, 303) = 2.9, p>.05. The gender of the partner was not related to performance on the final exam.

Table 3: Final exam scores

	Mean	Std. dev.	N
All pairers	75.2%	18.9	367
All non-pairers	74.4%	18.5	119
All men	76.2%	18.4	366
All women	71.4%	19.7	120
Paired men	76.7%	18.6	277
Paired women	70.8%	19.2	89

3.3. Confidence, satisfaction, and enjoyment

Table 4: Questions asked about each program

Confidence	On a scale from 0 (not at all confident) to 100 (very confident), how confident are you in your solution to this assignment?
Satisfaction (pairers only)	How satisfied are you with the way that you and your partner worked together on this assignment? (1=very dissatisfied, 7=very satisfied)
Satisfaction (non-pairers only)	How satisfied are you with how you spent your time on this assignment? (1=very dissatisfied, 7=very satisfied)
Enjoyment	How much did you enjoy working on this programming assignment? (1=not at all, 7=very much)

Of course the most important goal for students in any class is mastery of the material. This is certainly the case for introductory programming courses, where future success is dependent on a strong foundational knowledge. However, subjective experiences in introductory programming courses may also contribute to decisions about whether to pursue computer science related degrees. For this reason it is important to understand how the experience of pairing influences students' confidence in, enjoyment of, and satisfaction with their work. The specific questions students responded to in their logs are presented in Table 4.

3.3.1. Confidence in program solution

Overall, students who paired reported significantly higher confidence in their program solutions (89.4%) than students who worked independently (71.2%), *F*(1,482)=99.38, *p*<.001, and men were significantly more confident (87.0%) than women (81.1%), *F*(1,482)=14.62, *p* <.001. There was also a significant interaction between pairing and gender with regard to reported confidence. Simple effects follow-up tests of the interaction indicated that pairing resulted in more confidence for both women (86.8% vs. 63.0%), F(1, 482) = 50.54, p<.001 and men (90.3% vs. 74.6%), F(1, 482) = 54.94, p<.001. However, the 24% increase in confidence that pairing afforded women was even greater than the 15% confidence boost experienced by men who had the benefit of pairing (see Table 5). Partners' gender was unrelated to pairers' confidence levels.

Table 5: Confidence in solutions

	Mean	Std. dev.	N
All pairers	89.38	13.73	380
All non-pairers	71.21	23.95	106
All men	86.99	16.42	356
All women	81.11	21.57	130
Paired men	90.30	13.08	281
Unpaired men	74.61	16.42	75
Paired women	86.79	15.19	99
Unpaired women	62.97	28.29	31

3.3.2. Satisfaction with programming

Similarly, pairing students reported greater satisfaction (*M*=5.95) than non-pairing students (*M*=4.58), *F*(1,475)=118.05, p<.001, and men reported greater satisfaction (*M*=5.72) than women (*M*=5.45), *F*(1,475)=4.56, *p*<.05 (see Table 6). No interaction between pairing and gender was found. Because the question about satisfaction was not identical for the pairing and non-pairing students these results should be interpreted cautiously. We realized too late that we had asked the pairing students a question that was not appropriate for the non-pairers.

Table 6: Satisfaction (1=very dissatisfied, 7=very satisfied)

	Mean	Std. dev.	N
Pairers	5.95	.98	374
Non-pairers	4.58	1.28	105
Men	5.72	1.09	351
Women	5.45	1.40	128
Paired w/ a man	5.91	1.00	277
Paired w/ a woman	6.05	0.87	96

A partner by partner's gender ANOVA of pairers satisfaction revealed a marginal effect of partner's gender, F(1,369)=3.84, *p*=.051. Students that partnered with a woman were more satisfied with how they worked with their partner (*M*=6.05) than students that partnered with a man (*M*=5.91).

3.3.3. Enjoyment

A two-way (pairing X gender) ANOVA indicated that pairing students enjoyed working on programming assignments (*M*=5.14) more than non-pairing students (*M*=4.69), *F*(1,482)=9.00, *p*<.005. Overall women (*M*=4.84) and men (*M*=5.11) enjoyed the assignments equally well, *F*(1,482)=2.69, *p*>.05. However, among just the students who worked in pairs, men reported significantly higher enjoyment (*M*=5.21) than the women (*M*=4.91), *F*(1,482)=6.42, *p*<.05 (see Table 7) regardless of whether they partnered with a woman or a man.

Table 7: Enjoyment (1=very unenjoyable, 7=very enjoyable)

	Mean	Std. dev.	N
Pairers	5.14	1.14	380
Non-pairers	4.69	1.02	106
All men	5.11	1.11	356
All women	4.84	1.15	130
Paired men	5.21	1.13	280
Paired women	4.91	1.12	99

3.4. Persistence in computer science

In addition to being interested in the effects of programming on students' performance and subjective experiences with pairing, we were interested in the effect pairing might have on student persistence in computer science related majors. Specifically we wanted to know if using pairing as a learning tool for beginning programmers would influence subsequent computer programming course taking behavior, both in terms of attempts and pass rates, and students' decisions to major in computer science related fields.

Of course pursuing a computer science degree is dependent on success in the introductory course. For that reason we limited our analyses of persistence to the students who had successfully passed the class with a "C" or above. When interpreting these data it is important to keep in mind that nearly 10% more of the students who had paired in the introductory course successfully passed it. Because we followed students for one full academic year, our sub-sample was further limited to those who were still enrolled at UCSC three quarters after taking the course (N=321). Finally, although the introductory programming class is intended primarily for computer science majors, some students not intending to major in any of the computer science related fields also take the course. Most of the analyses below focus on those students who indicated on the first day of class an intention to pursue one of the computer science related majors offered at UCSC.

Among the students who were intending to pursue a computer science related major at the start of the introductory programming class, successfully passed the class with a "C" or better, and were still enrolled at UCSC a full year later (N=238; 187 men and 51 women), a significantly higher percentage of the students who had paired had gone on to attempt the subsequent programming course (Introduction to Data Structures) within a year (76.7%), than had the non-pairing students (62.2%), $\chi^2(1)$ =6.17, *p* <.05. Separate analyses by gender of the effect of pairing on whether the subsequent course was attempted within a year revealed about an 18% difference between pairers and non-pairers for both women and men (73.8% of paired women vs. 55.6% of non-paired women, and 88.0% of paired men vs. 69.4% of unpaired men). The pairing effect was statistically significant for men, $\chi^2(1)$ = 7.60, *p*<.01, but not for women, $\chi^2(1)$ = 1.19, *p*>.05. The increase in the percentage of students associated with pairing appears to be quite similar for men and women. The fact that this difference was statistically significant for men but not women, again is most likely attributable to the relatively small number of women (51 compared to 186 men) in this study.

Among students that attempted the Data Structures course, there was no significant difference between pairers and non-pairers pass rates on their first attempt (see Table 8). That is, students who paired in the introductory programming course were more likely to attempt the subsequent programming class, and were just as likely to pass it as those who learned to program independently.

Among the students initially intending a computer science major, and who passed the introductory course and remained at UCSC for at least a year, the pairing students were also more likely to have declared a computer science related major 1 year after completing the introductory programming class, $\chi^2(1) = 13.19$, $p < .001$ This was the case for both women and men. Women who paired were more likely than women who worked independently to be in a computer science related major (59.5% vs. 22.2%), $\chi^2(1) = 4.14$, $p < .05$. Similarly pairing men were also more likely to have declared a computer science related major 1 year later than men who worked alone (74% vs. 47.2%), $\chi^2(1) = 9.70$, $p < .005$ (see Table 9).

Table 8: Attempted and passed data structures course

	Attempt Rates	Pass Rates (on 1st attempt) of Attempters
All pairers	76.7%	73.6%
All non-pairers	62.2%	72.4%
Paired women	73.8%	77.4%
Unpaired women	55.6%	66.0%
Paired men	88.0%	72.7%
Unpaired men	69.4%	75.0%

Interestingly, the same pattern of results was observed among all students who successfully completed the introductory programming course and were still enrolled at UCSC a year later, regardless of whether they had initially been planning to major in one of the computer science related majors. Pairers were significantly more likely to have declared a computer science related major than non-pairers, $\chi^2(1) = 12.18$, $p < .001$, and that was the case for both men, $\chi^2(1) = 6.23$, $p < .05$ and women, $\chi^2(1) = 7.13$, $p < .01$ (see Table 9 for percentages).

Table 9: Percentage of students declaring a computer science major 3 quarters after intro course

	Pairers	Non-pairers
Students intending CS majors	70.8%	42.2%
Women intending CS majors	59.5%	22.2%
Men intending CS majors	74.0%	47.2%
Students intending any major (including undeclared)	56.9%	33.8%
Women intending any major (including undeclared)	46.3%	11.1%
Men intending any major (including undeclared)	59.9%	41.1%

4. Conclusion

The results of this study provide some of the most compelling evidence to date of the effectiveness of pair programming as a pedagogical tool. It appears that pairing bolsters course completion and consequently course pass rates, and contributes to greater persistence in computer science related majors. Moreover, students who pair produce higher quality programs, are more confident in their work, and enjoy it more. We hope these findings will encourage instructors to use pair programming not only in their introductory courses, but also in their upper level courses.

The continued underrepresentation of women in computer science underscores the need for strategies that foster women's interest and promote their success[7]. Pair programming appears to be one such approach. That the benefits associated with pair programming extend to both men and women speaks to its broad-based appeal. As we continue to investigate the effects of this technique on attracting and retaining female students, parallel research investigating these phenomena in the workplace is also needed.

Acknowledgments

This work was funded by National Science Foundation grant EIA-0089989. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation. We wish to thank undergraduate student Tristan Thomte and graduate students Wendy R. Williams, Wendy M. Limbert, Jennifer Bevan, and Brian Hanks for help with the data.

5. References

[1] Beck, K., *Extreme Programming Explained: Embrace Change*. 2000, Reading, Mass: Addison-Wesley.

[2] Williams, L., et al., *Strengthening the Case for Pair Programming*. IEEE Software, 2000. **17**(4): p. 19-25.

[3] McDowell, C., et al. *The Effects of Pair-Programming on Performance in an Introductory Programming Course*. in *33rd SIGCSE Technical Symposium on Computer Science Education*. 2002. Northern Kentucky: ACM Press.

[4] Nosek, J.T., *The Case for Collaborative Programming*. Communications of the ACM, 1998. **41**(3): p. 105-108.

[5] Williams, L.A. and R.R. Kessler, *All I Really Need to Know About Pair Programming I Learned in Kindergarten.* Communications of the ACM, 2000. **43**(5): p. 108-114.

[6] Williams, L.A. and R.R. Kessler. *The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education*, in *Thirteenth Conference on Software Engineering Education and Training*. 2000. Austin Texas: IEEE Computer Soc.

[7] *Tech-Savvy Educating Girls in the New Computer Age Executive Summary*. 2000, American Association of University Women Education Foundation (available at http://www.aauw.org/2000/techsavvy.html).