Female Computer Science Students Who Pair Program Persist

Linda L. Werner, Computer Science, University of California, Santa Cruz
Brian Hanks, Computer Science Information Systems, Fort Lewis College, Durango, Colorado
Charlie McDowell, Computer Science, University of California, Santa Cruz

Abstract

Pair programming has been found to be very beneficial in educational settings. Students who pair in their introductory programming course are more confident, have greater course completion and pass rates, and are more likely to persist in computer-related majors. Although pairing helps all students, we believe that it is particularly beneficial for women because it addresses several significant factors that limit women's participation in computer science. We provide reasons for our belief that pair programming helps women persist in these majors. We also repeat, with special emphasis on the impact on women, some details published elsewhere regarding our experiments on pair programming with college and university students. Additionally, we provide new data that support our original findings.

Categories and Subject Descriptors: K.3.2 [Computer and Information Science Education]: Computer Science Education

General Terms: Experimentation and Human Factors.

Additional Keywords and Phrases: Pair programming, collaboration, and gender.

1 Introduction

A 2000 UCLA survey of over 400,000 entering freshman at 717 colleges and universities across the US reported the largest computer skills confidence gender gap in the 35-year history of the survey. The gender gap in computer use was almost non-existent (79.5% men and 77.8% women report frequent computer use); however, only 23.2% of the women vs. 46.4% of the men rated their computer skills as “above average” or within the “top 10 percent”. Also 9.3% of the men vs. 1.8% of the women reported intent to pursue computer programming careers [Sax 2000]. This gender gap has been extensively written about and, unfortunately, the pipeline has been shrinking [Camp 1997; Camp 2001]. In

2004, 65% of the SAT I test takers had completed computer literacy related course work or experience. The majority (55%) of these students were women, yet when narrowed to course work or experience in computer programming, the percentage of women drops to 40%. In addition, of the 5% of the 2004 SAT I test takers who intended to major in computer or information science once in college, only 14% of these were women [College Entrance Examination Board 2004]. As reported by the Computing Research Association (CRA), little change has occurred during the years from 1993/1994 to 2002/2003 when less than 20% of the computer engineering/computer science BS degrees were awarded to women in each of these years. During that same period when gender data are available from the National Science Foundation (NSF), the percentages of science and engineering BS degrees granted to women has steadily increased with the percentage of women at 50% in the year 2000/2001 [CRA 2005].

Even among the most mathematically talented, women favor medicine and law as professions over careers in information technology because they perceive these professions as more socially meaningful and more interactive [Lightbody et al. 1997]. This is consistent with the AAUW [AAUW 2000] report that girls are not avoiding high tech careers because they are failing. Rather, the AAUW report identifies the following reasons why fewer women are majoring in computer science (CS):

(1) the widely held perception that a career in computing is not well-rounded or conducive to family life;
(2) the belief that work in the information technology field is conducted in a competitive rather than collaborative environment;
(3) the perception of CS as a solitary occupation that is not well integrated into social discourse or social institutions; and
(4) concerns about safety and security reported by women and their friends and families about working alone at night and on weekends in computer laboratories.

We propose that using pair programming in college and university CS courses could address three of these reasons for why fewer women are majoring in CS. We also have suggestions for removing the last of these four reasons.

We present promising results from three studies regarding the use of pair programming in beginning programming courses. These findings show that students who pair programmed were more confident in their programming solutions and enjoyed completing the assignments more than students who programmed alone. Paired students were more likely to complete the course and consequently to pass it. Results have previously been published regarding the primary study of over 500 primarily residential students in introductory programming courses at the University of California, Santa Cruz (UCSC) [McDowell et al. 2003]. We report here previously unpublished findings regarding a repeat of this study with over 200 students at two additional institutions of higher education: San Jose State University (SJSU) and Cabrillo College, both commuter campuses. We found that the new findings mirror those at UCSC regarding confidence and pass rates. Because of smaller populations, we do not have statistically significant results in most of the areas for the subset of women at these two additional institutions. We refer to this second experimental group as the secondary study. We also report on an additional group of UCSC students from the 2003-2004 academic year which we refer to in this paper as the tertiary study.

Paired students performed as well on individually taken final exams as solo students. For the primary study, we also looked at whether students continued to be registered as CS-related majors one year later. We found that paired students were more likely to persist. Significantly more paired women programmers than solo women programmers went on to declare a CS-related major [McDowell et al. 2003]. Because of this we claim that pair programming holds promise for closing the gender gap in CS.

2 What is Pair Programming and how can it be used in education?

Essentially all non-trivial software projects are created collaboratively. Almost all professional programmers have, on occasion, worked with another colleague together on one computer to debug a program that didn't work as expected. This informal process involving two collaborating individuals using a single computer has been formalized as pair programming, and has become more widely known because it is a key practice of the extreme programming (XP) development methodology [Beck 2000].

In XP, all software is designed, developed, and tested using pair programming. While pairing, one of the programmers, referred to as the 'driver', controls the keyboard and mouse, and is responsible for entering program code. The second programmer, known as the 'navigator', sits next to the driver and watches for errors, discusses alternative design approaches, and offers suggestions. The programmers regularly trade roles while pairing. One goal of pair programming is to have all code created collaboratively by the pair and to have the pair collectively ‘own’ the code. Code that is written by only one member of the pair is reviewed by both partners together before it is officially accepted as part of the program.

Traditional undergraduate introductory programming courses generally require that students work individually on their programming assignments. In these courses, working with another student on a homework programming assignment constitutes cheating and is not tolerated. The only resources available to help students overcome any problems that they are having are the course instructor, the textbook, and the teaching assistant. They are not allowed to work with their peers, who are struggling with the same material. A female student interviewed by Berenson et al [Berenson et al. 2005] observed that "you have to do all this stuff on your own and there's nobody to talk to and to ask a question to." This pedagogical approach teaches students that software development is an individual activity, potentially giving students the mistaken impression that software engineering is an isolating and lonely career.

Often, collaborative methods are used in upper division computer science courses such as compiler design or software engineering in which group projects are encouraged or mandated. Sometimes a software engineering instructor offers assistance to the student groups regarding techniques for collaboration but these topics are rarely discussed in other CS courses. One example is the research on using agile processes including pair programming in software engineering courses at NCSU [Berenson et al. 2005].

By deferring collaborative exercises to the upper division courses, we believe that many CS departments are losing female students who were interested in computer science but became discouraged by its focus on individual, socially isolating work. As reported by Berenson et al. [Berenson et al, 2005], a female student "said she had been taking computer science courses for three years and did not know anyone in her classes." This changed when she began using pair programming.

We recommend requiring students to pair program in all programming courses, especially introductory courses. We introduce our students to pair programming by having them read "All I Really Need to Know about Pair Programming I Learned In Kindergarten" [Williams and Kessler 2000a]. Additionally, we have published pair programming implementation guidelines that we derived during our primary study [Bevan et al. 2002]. One of us (Hanks) also uses the 'pair-draw' exercise to help students appreciate the benefits of pairing [Kerievsky, 2004].

3 How Does Pair Programming Lead to Persistence for Women in CS?

In the 2000-2001 academic year, 555 students (141 women, 413 men, and 1 whose gender was not reported) participated in a study on pair programming at UCSC. We studied four sections of our introductory programming course which were taught by three different instructors. In three of the sections, students pair programmed; in the fourth they worked by themselves. The instructor of the solo section also taught one of the paired sections, and is one of the co-authors of this paper (McDowell). The statistics summarized here were collected as part of that study and reported in [McDowell et al. 2003]. There was no significant difference between the pairing students and the non-pairing students with regard to high school GPA, transfer GPA, or SAT math scores.

We wanted to answer several questions with our study; one of these was, "Are women who pair program in their introductory programming course more likely to complete and pass the course?" Our definition of course completion means that the student took the final exam. To pass the course, a student had to receive a grade of "C" or better.

A comparison of paired and solo women (101 vs. 39) showed that those who paired were more likely than those who worked alone to complete the course (88.1% vs. 79.5%, p=.19). This 8 percentage point difference in completion rate is practically significant although it is not statistically significant. For men, a 10 percentage point difference in completion rates between the paired and solo students was significant (91.7% vs. 81.5%, p<.05). Although the increase in completion rates was similar for women and men, the lack of statistical significance for the women can be explained by the much smaller population of women in our study (140 women compared with 411 men). Among those who completed the course (by taking the final exam), the difference in pass rates between paired and solo programming students was not statistically significant (79.6% vs. 78.2%).

Statistics from our secondary study validate these findings. A comparison of paired and solo women (13 vs. 20) showed that those who paired were more likely than those who worked alone to complete the course (92.3% vs. 75.0%, p=.21). This 17 percentage point difference is practically significant although it is not statistically significant. For men, a 15 percentage point difference between the paired and solo students was significant (85.1% vs. 69.9%, p<.05). Among those who completed the course (by taking the final exam), the difference in pass rates between paired and solo programming students was not statistically significant (79.1% vs. 87.9%,p=.15), however using our terminology, it can be said to be practically significant that more of the solo completers passed the class. If we look at all of the students, significantly more of the paired students than solo students passed the class (66.0% vs. 52.3%, p<.05).

Further evidence that female students who pair program perform better in an introductory programming course is provided by data that was collected in three additional sections of our introductory programming course as part of the tertiary study conducted by one of the co-authors of this paper (Hanks). All students in these courses paired. Of the 24 female students who participated in the study, 23 (95.8%) took the final exam; 21 of these passed the course (91.3%). Similar rates were seen for men. Of the 91 men enrolled in the three sections of the course, 85 (93.4%) attempted the final exam; 78 of these passed the course (91.8%) These rates are comparable to or better than those reported in our primary study.

Our second question concerned retention in CS-related majors. We wanted to know if pair programming in the introductory classes led to increased numbers of women persisting in CS. We followed students in our primary study for one full academic year after the introductory programming course. We only followed students who had passed the course with a "C" or better. Our sample size was further decreased by those who were no longer at UCSC. Additionally, the numbers reported here only include students who stated on the first day of the introductory class that their major (or intended major) was in CS or a CS-related field. Even though our introductory programming course is primarily intended for CS or CS-related majors, the class includes students majoring in a wide variety of fields. For this part of our analysis, our sample size is 237 (51 women, 186 men).

A significantly higher percentage of the students who paired in the introductory course attempted the subsequent programming course required for CS-related majors (76.7% vs. 62.2%, $\chi^2(1) = 6.17$, $p < .05$). A separate analysis by gender revealed an 18.2% difference for paired vs. solo women (73.8% vs. 55.6%). The increase in attempt rates by women who paired over solo women was not statistically significant ($\chi^2(1) = 1.19, p=.27$) even though the same approximate difference (18.6%) in attempt rates was seen for paired men vs. solo men and was statistically significant (88.0% vs. 69.4% $\chi^2(1) = 7.60, p<.01$). Again, the lack of statistical significance for the data on women is probably attributable to the relatively small number of women in this part of the analysis.

Among the students in our study who attempted the second course (which did not use pair programming), we found no significant difference in pass rates between paired and solo students. Therefore, more students who paired passed the introductory course, more of these students attempted the second course, and this larger pool of students passed the second course at similar rates to those who worked alone in the introductory course.

As a second measure of retention, we wanted to know if the paired women students were more likely to declare a CS-related major one year after completing the introductory programming course. We found that 59.5% of the female potential CS-majors who paired had declared a CS-related major one year later, compared with only 22.2% of the women who worked alone. This result is both practically and statistically significant ($\chi^2(1) = 4.14$, $p < .05$). Men who paired were also more likely to have declared a CS-related major one year later than those who worked individually (74% vs. 47.2%, $\chi^2(1) = 9.70$, $p < .005$). The same pattern of effects was seen with our students who successfully completed the introductory programming class and were still enrolled at UCSC one year later **regardless** of what major (or no major) they declared on the first day of the introductory course.

The potential impact of this increased retention on the gender gap can be seen by looking at a hypothetical example. Assume that there are 100 potential computer science majors (50 women, 50 men) enrolled in an introductory programming course. If these students worked alone, one year later there would be 35 declared majors who were 31% female (22.2% of 50 women and 47.2% of 50 men). If these students paired, then one year later there would be 67 declared majors who were 45% female (59.5% of 50 women and 74% of 50 men).

Another area of concern was the potential impact of pair programming on student confidence. We believe that students who are confident in their computing abilities will be more likely to pursue studies in those areas. As part of our study, we asked students to complete a short questionnaire when they turned in each of their programming assignments. To assess student confidence levels, we asked them to respond to the question, "On a scale from 0 (not at all confident) to 100 (very confident), how confident are you in your solution to this assignment?"

Overall, students who paired reported significantly higher confidence in their program solutions than students who worked independently (89.4 vs. 71.2, $p<.001$). This is consistent with the findings from interviews of female students done by Berenson et al [Berenson et al. 2005]. Although all men as a group were significantly more confident than all women (87.0 vs. 81.1, $p<.001$), there was 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 increased confidence for both women (86.8 vs. 63.0, $p<.001$) and men (90.3 vs. 74.6, $p<.001$). We also found that the gender of a student's partner was unrelated to pairers' confidence levels. Women's confidence increased by 24 points when they paired compared with a 15 point increase for men. It appears that pairing had a greater effect on confidence levels for women and, therefore, may have a visible positive impact on the gender gap. Unpaired men reported 1.18 times greater confidence than unpaired women, while paired men reported 1.04 times greater confidence than paired women. Pairing seems to close the confidence gap between women and men.

Similarly, for our secondary study, paired women reported greater confidence levels than unpaired women (83.2 vs. 72.6, p=.31) but this increased reported confidence is not statistically significant probably because of the small sample size (n=22). The average reported confidence level for all paired students in our secondary study was 86.6 vs. the average reported confidence level for all unpaired students of 76.0. This difference is significant with p<.005.

We asked participants in our tertiary study at UCSC (in which all students paired), to answer the same question pertaining to confidence. We found that these pairing students exhibited similar levels of confidence as the pairing students in our original study. In the more recent study, the average confidence level for all students was 88.7, for men it was 88.8, and for women it was 88.3. The results from our secondary and tertiary studies add weight to our earlier finding that students who pair are more confident in their work and that the gender gap in confidence is diminished with pair programming.

4 Why does Pair Programming lead to Persistence for Women in CS?

Women's belief about the solitary nature of computer science is confirmed when they enroll in an introductory programming course that requires programming assignments to be done individually. Instead, when pair programming is used, women view programming as a collaborative exercise. Williams and Kessler suggest that "peer pressure" may be at work as a possible explanation for higher completion rates among paired vs. solo programming students [Williams and Kessler 2000a]. It may be the collaborative aspect of pair programming that is a major reason that the students remain in the class. The increased levels of confidence that can be attributed to pairing are probably also a factor in improved retention.

It is important for us that not only do the women stay in the class but they pass at similar rates to others. Given that the exams are individually taken, we have paired students mastering the course material at the same rates as the solo students. Additionally, if a 'pair-oriented culture' is encouraged by the use of short discussion periods during class time, then women might question their belief that work in the information technology field is conducted in a competitive rather than collaborative environment. They also might question their perception of CS as a solitary occupation that is not well integrated into social discourse or social institutions. Additionally, a serendipitous outcome of pair programming is that no one works alone late at night or on weekends in a computer laboratory. Partners work together. We hypothesize that these reasons cause pair programming to contribute to persistence of women in CS.

There is one remaining reason for why fewer women are majoring in computer science as identified by the AAUW report that is not addressed by pair programming. The report states that women perceive that a career in computing is not well-rounded or conducive to family life. An effort needs to be made by introductory programming textbook authors to create exercises and examples that "highlight the human, social, and cultural dimensions and applications of computers rather than the technical advances, the speed of the machines or the entrepreneurial culture surrounding them" [AAUW 2000, page 10]. There seems to be some hope in this area. The recent Java textbook by Cohoon and Davidson [Cohoon and Davidson 2004] includes programming exercises and examples drawn from fields such as medicine, personal finance, health and fitness, and data visualization. We are encouraged by this, and hope that other authors follow this lead.

5 Conclusions

Pair programming has been shown to be beneficial to all students. We argue that it is particularly beneficial for women because it addresses factors that potentially limit their participation in CS. The collaborative nature of pair programming teaches women students that software development is not the competitive, socially isolating activity that they imagined. It encourages women to pursue computer science as a major and as a potential career. Because of this, we strongly advocate the use of pair programming in all introductory programming courses. We are now using pair programming in all introductory programming courses we teach. Additionally, we use optional pair programming in all upper division programming courses we teach. The teachers who experimented with pair programming for the secondary study all strongly believe in it and encourage their students to use it. We suggest you try it too!

6 Acknowledgements

This work was funded by National Science Foundation grants EIA-0089989 and DUE-0341276. 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 also want to thank both Heather Bullock and Julian Fernald at UCSC for their participation as investigators on our NSF study. Thanks also go to Michael Burke and Vladimir Drobot at SJSU and Susan Nerton and Ed Parrish at Cabrillo College who graciously allowed us to experiment with their classes. A final thanks goes to the anonymous reviewers, one of whom suggested the example regarding the gender gap and a hypothetical class of 100 students.

7 References

American Association of University Women Education Foundation Commission on Technology, Gender, and Teacher Education, 2000, Tech-Savvy Educating Girls in the New Computer Age, http://www.aauw.org/2000/techsavvy.html.

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

Berenson, S.B., Slaten, K.M., Williams, L., and Ho, C-w. under review 2005. Voices of Women in a Software Engineering Course: Reflections on Collaboration. Journal on Educational Resources in Computing (JERIC).

Bevan, J., L. Werner, and C. McDowell 2002, Guidelines for the Use of Pair Programming in a Freshman Programming Class, presented at Conference on Software Engineering Education and Training, Kentucky.

Camp, T. 1997, The Incredible Shrinking Pipeline. Communications of the ACM, 40 (10) 1997, 103-110.

Camp, T. 2001, Women in Computer Science: Reversing the Trend CRA-W August, 2001, accessed on November 19, 2004 at http://www2.cs.cmu.edu/~women/resources/aroundTheWeb/hostedPapers/Syllabus-Camp.pdf.

Cohoon, J., and J. Davidson, 2000, Java 1.5 Program Design, McGraw-Hill, Boston.

College Entrance Examination Board, 2004, College-Bound Seniors A Profile of SAT Program Test Takers, accessed on January 23, 2005 at http://www.collegeboard.com/prod_downloads/about/news_info/cbsenior/yr2004/2004_CBSNR_total_group.pdf

CRA. 2005, CRA Taulbee Trends: Women Students & Faculty, last updated May 6, 2004. CRA Taulbee Survey. See http://www.cra.org/info/taulbee/women.html.

Kerievsky, Josh, 2004, Pair Draw, accessed on February 6, 2005 at http://industriallogic.com/games/pairdraw.html

McDowell, C., L. Werner, H. Bullock, & J. Fernald, 2003, The Impact of Pair Programming on Student Performance, Perception, and Persistence, Proceedings of the 25th International Conference on Software Engineering, Portland, OR, 602-607.

Lightbody, P., G. Siann, L. Tait, and D. Walsh 1997, A Fulfilling Career? Factors Which Influence Women's Choice of Profession, Educational Studies, 23, 1997, pp. 25-37.

Sax, L.J., Astin, A.W., Korn, W.S., and Mahoney, K.M. 2000, The American Freshman: National Norms for Fall 2000. See http://www.gseis.ucla.edu/heri/norms_pr_oo.html for a summary.

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

Williams, L.A. and R.R. Kessler. 2000b, All I Really Need to Know about Pair Programming I Learned In Kindergarten in Communications of the ACM, 43, 5, 2000. p. 108-114.