Improving the CS1 Experience with Pair Programming

Nachiappan Nagappan[1], Laurie Williams[1], Miriam Ferzli[2], Eric Wiebe[2], Kai Yang[1], Carol Miller[1], Suzanne Balik[1]

[1]Department of Computer Science
[2]Department of Math, Science and Technology Education
North Carolina State University, Raleigh, NC 27695
{nnagapp, lawilli3, mgferzli, wiebe, kyang, miller, spbalik}@unity.ncsu.edu

Abstract

Pair programming is a practice in which two programmers work collaboratively at one computer, on the same design, algorithm, or code. Prior research indicates that pair programmers produce higher quality code in essentially half the time taken by solo programmers. An experiment was run to assess the efficacy of pair programming in an introductory Computer Science course. Student pair programmers were more self-sufficient, generally perform better on projects and exams, and were more likely to complete the class with a grade of C or better than their solo counterparts. Results indicate that pair programming creates a laboratory environment conducive to more advanced, active learning than traditional labs; students and lab instructors report labs to be more productive and less frustrating.

Categories & Subject Descriptors

K.3 [**Computers & Education**]: Computer & information science Education- *Computer Science Education*.

General Terms: Management, Human Factors

Keywords: Pair programming, collaborative environment, Computer Science education.

1 Introduction

In industry, software developers generally spend 30% of their time working alone, 50% of their time working with one other person, and 20% of their time working with two or more people. [3] However, most often in an academic environment, programmers must learn to program alone, and collaboration is considered cheating. Unfortunately, this time spent working alone is inconsistent with a student's future professional life in which collaboration is both encouraged and required. In addition, studies show that cooperative and collaborative pedagogies are beneficial for students [6, 7].

In pair programming one person, called the *driver*, is responsible for typing at the computer or documenting a design. The other partner, called the *navigator*, observes the work of the driver, looking for defects in the work of the driver and is an ever-ready brainstorming partner. Research results [2, 8, 11] indicate that pair programmers produce higher quality code in about half the time when compared with solo programmers. These research results are based on experiments held at the University of Utah in a senior-level Software Engineering course. The focus of that research was the affordability of the practice of pair programming and the ability of the practice to yield higher quality code. However, the researchers observed educational benefits for the student pair programmers. These benefits included superior results on graded assignments, increased satisfaction/reduced frustration from the students, increased confidence from the students on their project results, and reduced workload of the teaching staff.

These observations inspired further research directed at the use of pair programming in educating Computer Science students. Educators at the University of California-Santa Cruz [1, 5] and North Carolina State University [9, 10] have reported on the use of pair programming in introductory undergraduate programming courses. Experiments specifically designed to assess the efficacy of pair programming in an introductory Computer Science classroom found that pair programming improved retention rates and performance on programming assignments.

This paper details the results of our experiment carried out at North Carolina State University. We provide results from a larger sample size than previously reported. The remainder of this paper is organized as follows: Section 2 provides a description of the experiment; Section 3 discusses qualitative findings on pair programming in the CS1 laboratory; Section 4 shares the results of our quantitative findings; Section 5 highlights a few challenges we faced during this experiment and Section 6, summarizes our findings and discusses our future work.

2 Experiment

In the 2001-2 academic year, an experiment was conducted in the CS1 course at North Carolina State University. The course was taught with two 50-minute–lectures and one three-hour lab each week. Students attended labs in groups of 24 with others in their own lecture section. The lab period was run as a closed lab where students were given a weekly assignment to complete during the allotted time. Lab assignments are "completion" assignments whereby students fill in the body of methods in a skeleton of the program prepared by the instructor. Student grades are based on two midterm exams, one final exam, lab assignments, and programming projects that are completed outside of the closed lab. The programming projects are generative, that is, the students start the project from scratch without any structure imposed by the instructor. The course is a service course and is therefore taken by many students throughout the university. Most students are from the College of Engineering and are either freshmen or sophomores.

SIGCSE'03, February 19-23, 2003, Reno, Nevada, USA.

However, students of all undergraduate and graduate levels may take the course.

The Fall 2001 experiment was run in two sections of the course; the same instructor taught both sections. Additionally, the midterm exams and the final exam were identical in both sections. One section had traditional, solo programming labs. In the other section, students were required to complete their lab assignments utilizing the pair programming practice. When students enrolled for the class, they had no knowledge of the experiment or if their section would have paired or solo labs. In the pair programming labs, students were randomly assigned partners based on a web-based computer program; pair assignments were not based on student preferences. Students worked with the same partner for two to three weeks. If a student's partner did not show up for a particular lab, after 10 minutes, the student was assigned to another partner. If there were an odd number of students, three students worked together; no one worked alone. Closed labs are excellent for controlled use of pair programming [1]. The instructor or teaching assistant can ensure that people are, indeed, working in pairs at one computer. He or she can also monitor that the roles of driver and navigator are rotated periodically.

Our course also includes programming projects that require work outside of the closed lab. We gave the students in both sections the option of working alone or in pairs for these projects. Only students who attained a score of 70% or better on the exams could opt to pair. (We felt those who did not attain a score of 70% or above should not work with a pair on the project lest they rely too heavily on their partner to produce the project.) Most students, who were eligible to pair, chose to pair program on projects. However, the instructors now feel that the 70% eligibility might be unfair to the students, and this practice has been discontinued as of Fall 2002.

Using this Fall 2001 research design, we also completed a study on a larger scale in the Spring 2002 semester. In the fall, 112 students were in the solo section and 87 were in the paired section, whereas in the spring 156 students worked solo and 346 students worked in pairs. Our study was specifically aimed at the effects of pair programming on beginning students. Therefore, we analyzed the results of the freshman and sophomores only. We also only analyzed students who took the course for a grade, concluding that students who audited the class or took it for credit only were not as motivated to excel as other students. This reduced our sample size to N=69 in the solo section and N=44 in the paired section for the Fall semester, and N=102 for the solo section and N=280 in the paired section for the Spring semester.

In our experiment (spanning both Fall and Spring semesters), we examined the following five hypotheses:

H1. A higher percentage of students who have participated in pair programming in CS1 will succeed in completing the class with a grade of C or better when compared with students who have worked solo in CS1.

H2. Students' participation in pair-programming in CS1 will lead to better performance (higher scores) on the examinations when compared with students who have worked solo in CS1. (Examinations are completed solo by all students)

H3. Students' participation in pair-programming in CS1 will lead to better performance on course projects (higher project scores) in that class when compared with students who have worked solo in CS1.

H4. Students' participation in pair-programming will lead to a reduced workload in terms of grading, questions answered, and teaching effort for the course staff when compared with the teaching staff for students who worked solo in CS1.

H5. Students in paired labs will have a positive attitude towards collaborative programming settings when compared with students who have worked solo in CS1.

3 Qualitative Results

Each semester, we observed and codified many paired and solo lab sections. In addition, two focus groups were held, one with a randomly selected group of students and the other with a randomly selected group of lab instructors (LIs). (See focus group technical report [4].) Analysis of qualitative data from lab observations and focus groups strongly support pair programming in the CS1 laboratory. The next sections detail student and lab instructor perspectives on pair programming.

3.1 Students

Solo lab sessions were quiet and appeared to be very frustrating for the students. Frequently, a student needed to wait 10-30 minutes to ask a question, often a fairly simple one. During this waiting period or "down time", students were often very unproductive (i.e. "stuck"). Alternately, paired labs were vocal and interactive. Students in paired labs engaged in extensive discussion throughout the entire lab session, and students seemed to help each other resolve questions. Most often, each pair could piece together the knowledge they needed to figure out questions and remain productive. Because most pairs were self-sufficient, lab instructors had time to get around to more students than in the unpaired sections. Paired students who needed help, found it easy to get help from the LI, and had little "down time." [9]

During the focus group discussion, students stressed the advantages of pairing. Primarily, students brought up the benefits of having their questions answered immediately by their partner rather than having to wait for an LI. Having someone there while working on problems also seemed to help them pick up on minor errors and to focus on understanding conceptual knowledge.

Since communication skills and collaboration are important components of paired learning, students recognized that the paired labs made them work on these skills. Students realized that the paired format mimics real world settings where people are often randomly matched to work and collaborate on programming projects.

3.2 Lab Instructors

In solo lab sections, the LIs were often overwhelmed with questions. LIs often spent a minimum of five minutes and a maximum of 20 minutes with each student. LIs remained busy answering basic questions for the duration of the lab sessions. In paired labs, instructors spent more time discussing advanced issues with students, rather than answering basic questions.[9] For example, students in paired labs would ask the LIs how to improve their algorithm, or how to apply it to another scenario. Questions from students in solo labs were mostly about fixing syntax errors or getting compilation errors clarified

In the focus groups, the LIs all agreed that implementing the paired protocol gave them flexibility and time to give students equal opportunities for questions, discussions, and other support. As a result of having more time for meaningful exchanges with students, LIs found their jobs more satisfying and rewarding when teaching in paired labs. An added benefit is that LIs of paired labs graded

half the number of projects and labs as compared to the LIs of solo labs.

LIs noted that students in paired labs displayed more active participation in their learning than students in the unpaired labs. Paired student questions displayed higher order thinking such as application, synthesis, and evaluation. LIs observed that paired students' efforts and willingness to learn seemed to surpass their "traditional" counterparts.

(H4) We hypothesized that students' participation in pair programming will lead to a reduced workload for course staff. Our qualitative findings support this claim.

3.3 Common Concern

In both focus groups, the students and LIs noted the importance of having "compatible" partners. Two suggestions for constructing compatible pairings were to have them be based on personality type and/or on skill level. We address our research plans in this area in Section 5.

4 Quantitative Findings

In the prior section, we shared our qualitative findings that pairing creates a laboratory environment conducive to more advanced, active learning; both students and lab instructors reported this lab time to be more productive and less frustrating. In this section, we discuss quantitative results from data comparing paired to solo students.

4.1 Success Rate/Retention

First, we examined the percentage of students who succeeded in the class by completing the course with a grade of C or better. Historically, beginning Computer Science classes have poor success rates. Despite the good intentions and diligent work of computer science educators, students find introductory computer science courses very daunting—so daunting that typically one-quarter of the students drop out of the classes and many others perform poorly (by receiving a grade of D or F).

Using the above criteria, we combined results for the Fall 2001 and Spring 2002 semesters as shown in Table 1. Our results indicate that pairing helped the non-CS majors but did not cause any significant improvement among the CS majors. A Chi-Square test was run on the success rates and it showed the solo and paired sections to be statistically independent ($\chi^2(1)=0.0043$, $p < 0.98$). These results are consistent with a similar study at the University of California UC-Santa Cruz that reported 92% of their paired class and 76% of their solo class completed the course [5].

Table 1: Success Rate

Semester	Paired (%)	Solo (%)
Non-CS Majors	**66.4 (N=274)**	**55.9 (N=145)**
CS Majors	**83.0 (N=50)**	**84.0 (N=26)**

(H1) We hypothesized that pair programming would increase the success rate of the students who used the practice (measured by taking students with a grade of C or higher). Our results validated this claim for non-CS majors.

4.2 Performance on Examinations

In the fall semester, students in the paired section performed better on the two-midterm examinations and the final examination, as shown in Table 2. We removed 0 scores from our analysis, making these results based on scores of students who attempted to take the exam.

Table 2: Examination Scores Fall 2001

Exam	Paired Mean	Paired Std Dev	Solo Mean	Solo Std Dev
Midterm 1	**78.7**	11.8	73.4	13.8
Midterm 2	**65.8**	24.2	49.5	27.2
Final	**74.1**	16.5	67.2	18.4

As stated earlier, students chose their class section without knowledge of the experiment or pair programming. We had hoped that their random enrollment in the class would yield equivalent sample groups based on their SAT-Math scores. However, the students in the paired group had a mean SAT-Math score of 662.1 while the solo group had a mean score of 625.4. When using SAT-Math as a covariate, an ANCOVA test does not show any significant difference between sections with regards to any of the exams. Based on these results, we cannot conclude that pair programming in the laboratory helped students perform better on exams. Correspondingly, in the Spring semester we obtained exam results that did not yield any statistically significant improvement in test results by pair programmers. Educators can be concerned that pairs will learn less because they had the ability to lean on their partner. We have certainly not found this to be the case.

(H2) We hypothesized that Students' participation in pair-programming in CS1 will lead to better performance measured by higher scores on the examinations. Our results have not validated this claim to a statistically significant level.

4.3 Performance on Programming Projects

In the fall semester, students in the paired section performed better on the first two of three programming projects, as shown in Table 3.

Table 3: Programming Projects-Fall 2001

Exam	Paired Mean	Paired Std Dev	Solo Mean	Solo Std Dev
Project 1	**94.6**	5.3	78.2	26.5
Project 2	**86.3**	19.7	68.7	33.7
Project 3	73.7	27.1	74.4	29.0

To validate the statistical significance of these results, we ran an ANCOVA test on the data (again examining possible correlation between project scores and the student's SAT-Math scores). The ANCOVA demonstrated a statistically significant improvement in performance of the pairs on Project 1 ($F(1,94)=8.12$, $p<0.0054$) and Project 2 ($F(1,78)=4.52$, $p<0.0367$). However, this analysis did not demonstrate improved performance on Project 3. Perhaps, this is because by Project 3 the lower performing students had dropped in the solo section but were still working in the paired section. In the Spring 2002 semester, we saw no statistically significant difference in project scores by either group, though the paired students often performed marginally better.

(*H3) We hypothesized that students who pair programmed would have higher project scores compared with the solo programmers. From our results, paired and solo programmers have comparable scores in the projects, though in some cases paired programmers have marginally higher scores than the solo students.*

4.4 Results Commentary

We wish to discuss two factors that may influence these results on both the examinations and the projects. First, the implementation of

pairing in the lab portion of the course may have enough of a positive influence to keep students from dropping out of the course, or it could have boosted their grades enough to allow them to pass the course. As a result, the poorer performing students may have negatively influenced the calculation results of the paired section. These poorer performing students dropped the class or did not take exams in the solo section, removing themselves from the calculation pool. Researchers at UC-Santa Cruz have also made this same speculation, [5] because their paired section also did not achieve statistically significant higher test scores than the unpaired section. Additionally, only approximately 40% of the exam content required program code to be written in the answers. The rest of the exams were short answer and multiple choices. Quite feasibly, pair programming might not help improve students' answers to short-answer and multiple-choice questions.

4.5 Attitude

Students in paired labs will have a positive attitude toward working in collaborative software development environments. A survey was conducted among the students who worked in pairs throughout the spring semester. Eighty percent of the students in the paired section indicated that they were neutral (19.8%) or positive (59.9%) about pairing in the future.

(H5) We hypothesized that students in paired labs will have a positive attitude towards working in collaborative software development environments. Our survey results supported these claims.

5 Challenges

As with all learning methodologies there were certain challenges we encountered during this experiment over the fall and spring semesters.

- In a small percentage of cases, the random pairing led to incompatible partners, which led to conflicts during working. We hope to address this in our future work by matching people according to personality profile and/or skill type.
- The LIs have to monitor that one partner does not dominate the pair or that one partner is burdened with the entire workload. Student peer evaluations often to not reflect such difficulties. However, to certain degree, students to not want to "turn in" their partner. As a result, the LIs must also be observant of the chemistry and working of the pair in the closed labs

6 Conclusions and Future Work

Our study provides strong results of the following findings:

- Pair programming helps in the retention of more students in the introductory computer science stream.
- Students in paired labs have a more positive attitude toward working in collaborative environments; this should ultimately help the student in his/her professional life.
- Pair programming in an academic environment reduces the burden on the LI because the pairs helped each other, enabling the LI to perform more efficiently.
- From the results we have obtained regarding the tests and the projects, we can conclude significantly that pair programming among students is in no way a deterrent to student performance.

We plan to continue the experiment in the 2002-3 academic year with some modifications. Personality profiles like the Myer-Briggs personality tests will be used to determine a student's personality. We will experiment with successful matching patterns. This will help to provide us with more insight as to how personality profile matters in pair programming. We will also gather results for minority and female students to obtain meaningful results for these important groups.

7 Acknowledgements

The National Science Foundation Grant DUE CCLI 0088178 provided funding for the research in this pair programming experiment.

References

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

[2] Cockburn, A. and Williams, L., "The Costs and Benefits of Pair Programming," in *Extreme Programming Examined*, G. Succi and M. Marchesi, Eds. Boston, MA: Addison Wesley, 2001, pp. 223-248.

[3] DeMarco, T. and Lister, T., *Peopleware*. New York: Dorset House Publishers, 1977.

[4] Ferzli, M., Wiebe, E., and Williams, L., "Paired Programming Project: Focus Groups with Teaching Assistants and Students," North Carolina State University, Raleigh, NC CSC TR-2002-16, 2002.

[5] McDowell, C., Werner, L., Bullock, H., and Fernald, J., "The Effect of Pair Programming on Performance in an Introductory Programming Course," presented at ACM Special Interest Group of Computer Science Educators, Kentucky, 2002.

[6] Slavin, R., *Using Student Team Learning*. Boston: The Center for Social Organization of Schools, The Johns Hopkins University, 1980.

[7] Slavin, R., *Cooperative Learning: Theory, Research and Practice*. New Jersey: Prentice Hall, 1990.

[8] Williams, L., Kessler, R., Cunningham, W., and Jeffries, R., "Strengthening the Case for Pair-Programming," in *IEEE Software*, vol. 17, 2000, pp. 19-25.

[9] Williams, L., Wiebe, E., Yang, K., Ferzli, M., and Miller, C., "In Support of Pair Programming in the Introductory Computer Science Course," *Computer Science Education*, vol. September, 2002.

[10] Williams, L., Yang, K., Wiebe, E., Ferzli, M., and Miller, C., "Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations," presented at OOPSLA Educator's Symposium, Seattle, WA, 2002.

[11] Williams, L. A., "The Collaborative Software Process PhD Dissertation," in *Department of Computer Science*. Salt Lake City, UT: University of Utah, 2000.