# ASSESSMENT USING PEER EVALUATIONS, RANDOM PAIR ASSIGNMENT, AND COLLABORATIVE PROGRAMING IN CS1

Timothy Urness
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
515 271-2118
timothy.urness@drake.edu

**ABSTRACT**

In this paper we describe a technique for student assessment that uses peer evaluation and random pair assignment in collaborative programming assignments in CS1. A common concern of professors implementing collaborative (pair) programming is the potential for a student to not actively participate in the programming process. In this case, a student's skills and abilities may not be developed or evaluated. In addition to traditional grading of an assignment, a survey was given to each student following every assignment in order to assess the individual contribution and comprehension of the assignment requirements. The survey asked each student to briefly evaluate his or her teammate's cooperation as well as to project the teammate's ability to re-write the code. The results of this assessment method indicate that the assignment quality greatly increased and exam scores were comparable compared to previous course offerings when assignments were completed individually.

**INTRODUCTION**

Programming is a critical component of most CS1 courses. Studies have shown that requiring students to work in teams of two (pair programming) has many benefits. The benefits include: code quality improvement, increased number of students successfully passing the course, increased student enjoyment, lower levels of frustration, and reduced instances of cheating [1, 2, 4, 8, 9]. Collaborative learning, such as pair programming, is a more realistic model of how software is developed in industry as opposed to the solitary programming conducted in many CS1 courses.

In order to assure that every student experiences the advantages of pair programming, we have developed a system designed to evaluate and encourage participation. First, each student is held responsible for their contribution to the pair through the use of a peer evaluation that is delivered after each programming assignment is completed. In order to assess the individual contribution and comprehension of the assignment requirements, a survey was given to each student following every assignment. The survey asked each student to briefly evaluate his or her teammate's cooperation as well as to project the teammate's ability to re-write the submitted code. Additionally, the programming pairs are randomly assigned throughout the semester. This ensures that one student cannot "freeload" behind a friend or the same ambitious student throughout the term.

For the purposes of this paper, we define the term "pair programming" to mean two students working on a program that will be submitted jointly. Traditionally, pair

programming consists of two programmers working side-by-side at *one* computer [11]. It has been shown that distributive pair programming can achieve the same benefits as collocated pair programming [3]. For our purposes, we did not require teams to program at the same time or to model the *driver-navigator* relationship that is sometimes associated with pair programming. We also did not stipulate that students need to meet face-to-face to collaborate. Instead, a "team lead" was designated and was responsible for managing the work and final submission of the assignment.

## RELATED WORK

Several studies have demonstrated that pair programming is an effective pedagogical technique, especially in introductory computer science courses [5, 7, 10]. A number of different benefits that these studies have shown include: code quality improvement, increased number of students successfully passing the course, increased student enjoyment, lower levels of frustration, and reduced instances of cheating [1, 2, 4, 8, 9]. Also, pair programming helps develop communication skills and teamwork that will be required in industry [6, 11].

Additionally, the benefits extend to the teaching staff as pair programming often decreases the number of questions from students and reduces the number of assignments to grade by approximately half [9]. It has also been shown that distributed pair programming can have the same benefits of teams working at the same computer [3].

## MOTIVATION

There are several motivations for adopting pair programming in CS1. First, we wanted to encourage students to interact. A traditional stereotype of computer scientists states that their work is predominately isolated and solitary. In order to change this perception of computer science (and possibly attract new majors to the field), we required that students interact and work with other students in the class. Secondly, pair programming would cut grading obligations roughly in half. Another motivation was to increase the amount of code that students *read* during a semester. It is our impression that a well-rounded computer scientist should be able to understand written code in addition to writing and implementing algorithms. Collaborating in groups forces students to read others code in addition to writing it themselves. In our opinion, code reading and code comprehension is a skill that is often underdeveloped in CS1 courses and working in pairs would force students to evaluate how their own code is read by others. Lastly, we felt that introducing students to their peers and requiring collaboration could reduce the level of frustration that a student experiences when he or she is beginning to learn to program.

During the implementation of our random pair programming, we found that the quality of the assignments was greatly improved over previous semesters when the programs were developed individually. The results are discussed later in the paper.

## CONCERNS

A common concern of professors implementing pair programming is the potential for a student to not participate in the programming process. There is a significant possibility that a student may not fully understand the code the team has submitted as a student's teammate may complete a vast majority of the assignment. In this undesirable

situation, the students' skills and abilities may never be developed or evaluated. Furthermore, individual skills or abilities could be sacrificed if the time developing programs was shared. Would students fully understand the code their team is submitting? Would pair programming lead to decreased individual exam scores?

We developed a method described in the following section to help ensure that students were accountable for the skills that the programming assignments were intended to reinforce.

## METHODOLOGY

In an effort to achieve the advantages of pair programming as well as address the concerns, we implemented the following system:

First, we randomly selected the teams. A randomly generated algorithm was developed that paired two classmates together. The first student on the list was responsible for submitting the homework assignment. We referred to this person as the "team lead." The random pairing algorithm was developed so that each student would get the same number of opportunities to be the "team lead" throughout the course, while being paired with random partners for each assignment.

We did not require the teams to meet outside of class or program at the same time. While interaction was encouraged, we did not stipulate that students needed to meet face-to-face to collaborate. Instead, the "team lead" was responsible for managing the work and final submission of the assignment. This allowed students to communicate in a variety of different methods (e.g. email, phone, Facebook, etc.) and removed the restriction of requiring students to be in the same room in order to work on an assignment. It also removed the restriction that pairs had to find time to work on the assignment simultaneously. While some students chose to work together at the same computer at the same time, most students shared access to code by emailing their teammate.

Additionally, after each assignment was due, we required each student to fill out a survey asking two questions designed to evaluate his or her teammate. The questions asked students to rate their teammates on a Likert scale from 1 to 5 (see figure 1). The numerical results of the survey were included in each student's final assignment grade. In an effort to keep the students anonymous and uninhibited to honestly evaluate their teammate, the results of the peer evaluations were not disclosed to the students until after the final exam. This anonymity prevented a student from knowing the points earned or lost via a poor evaluation from a particular former teammate. It was our intention that this would allow students to be accountable and honest when filling out the survey.

**Is your partner easy to work with?**
Rate on a scale from 1 to 5 where
1 = "No. I am glad this assignment is over and I won't have to work with this person again."
5 = "Absolutely. I would like to have this partner for another assignment."

**Do you feel like your partner understands the code your team submitted?**
Rate on a scale from 1 to 5, where
1 = "understands nothing; it would be difficult for my partner to individually rewrite the code."
5 = "understands a vast majority of it; it would be very easy for my partner to individually rewrite the code."

Figure 1: Survey questions asked after every programming assignment.

**RESULTS**

We compared the results of average exam scores and average programming assignment scores from two different semesters of CS1. The first semester used individual programming and the second semester used the pair programming and survey approach described in the previous section. Figure 2 shows that the average assignment quality over the course of seven programming assignments greatly increased when pair programming was implemented. These results have also been reflected in previous studies [7]. The exam scores did decline slightly when pair programming was used (90% vs. 87.7%); however, the exam scores were comparable to the previous sessions of CS1 (see figure 3).

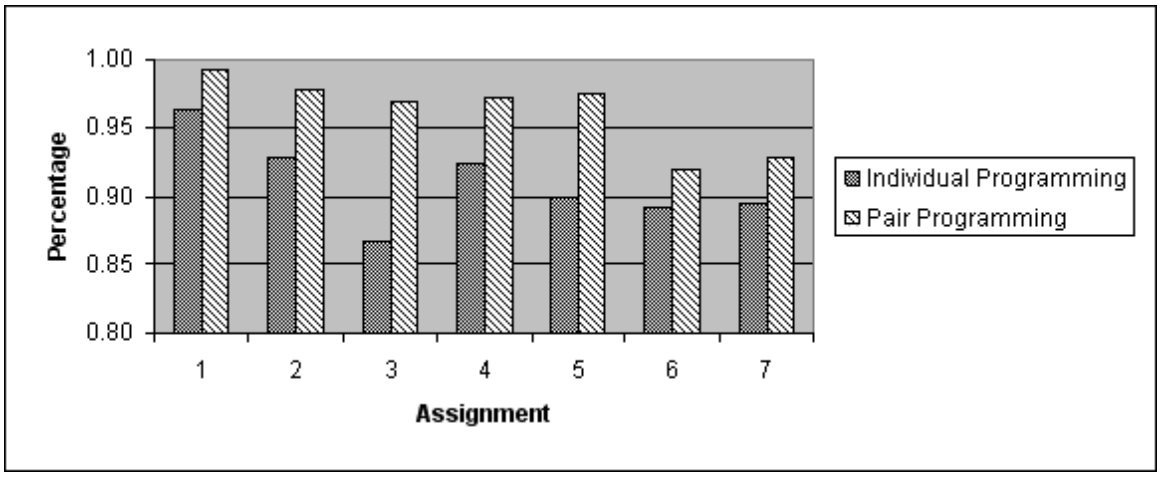


Figure 2: Average assignment scores of two semesters of CS1

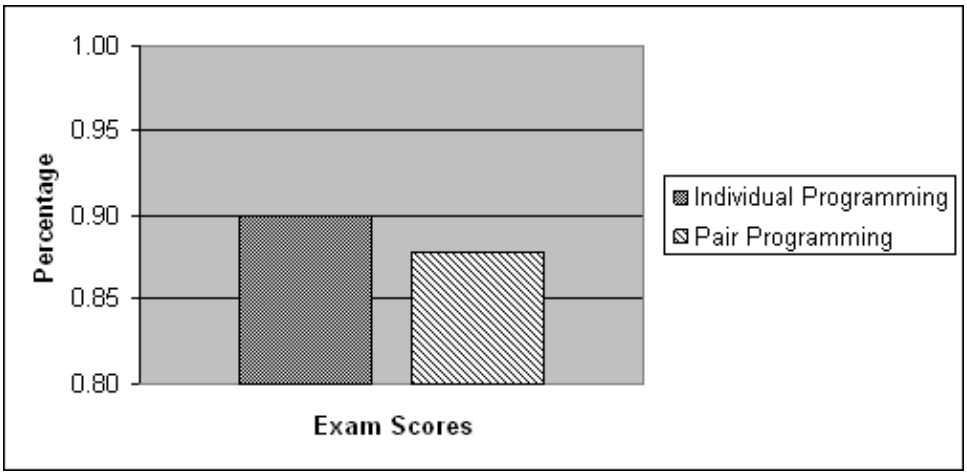

Figure 3: Average exam scores of two semesters of CS1

**ANALYSIS**

The initial motivations of this technique were largely selfish in nature: to make the class easier to teach and maintain by reducing the number of assignments to grade. We were pleasantly surprised that the students also thrived in this environment. The students demonstrated obvious benefits such as improved code quality and teamwork.

We have several hypotheses that explain the increased quality of the homework submissions: First, working in a team provides motivation by making a student accountable to another student in the class. We feel that a student in a team may work harder to ensure all of the points in the assignment are earned; a student may be satisfied with a lower grade if it only affects his or her grade. Secondly, the survey question that asks "Is your partner easy to work with" often prompts the students to contact each other shortly after the assignment is posted. In most cases, the pairs contact each other quickly and start working on the assignment soon enough to ask clarifying questions (if necessary) in order to complete the assignment and earn all of the points. The primary complaint that is registered amongst students is that several days pass before the partner responds to an email. This usually results in a partner giving a lower score for this question on the survey. Lastly, collaborating on code allows someone else to see possible mistakes or overlooked points.

An unanticipated positive side-effect of pair programming is that we saw fewer students during office hours who were frustrated. The assigned partner would often help frustrated or lost students, and we had very few instances of pair incompatibility. Additionally, we saw fewer questions from students "stuck" on the homework assignment the day before it was due when using pair programming.

**CONCLUSION**

We implemented a technique in which we conducted a brief survey after each assignment was completed. The survey asked each student to evaluate their partner on two questions: were they easy to work with and did they understand the code that was submitted. The results of these questions were used in the final grade of each student. Additionally we randomly assigned classmates in teams of two and switched the pairings after each assignment. This was designed to prevent students from "hiding" or

"freeloading" behind the same partner and inhibiting the development or evaluation of their abilities.

The results of pair programming in this implementation show a significant increase in quality of programming assignments when compared to a previous semester where individual programming was used. The individual exams scores were comparable between semesters.

In conclusion, we have found assessment using peer evaluation and random pair assignment in collaborative programming assignments in CS1 to be an effective pedagogical approach. The results of this method indicate that the assignment quality greatly increased and exam scores were comparable compared to a previous course offering when assignments were completed individually.

**REFERENCES**


[1] Braught, G., Eby, L. M., Wahls, T., The effects of pair-programming on individual programming skill, *ACM SIGCSE Bulletin*, 40, (1), 200-204, 2008.

[2] Hanks, B., McDowell, C., Draper, D., Krnjajic, M., Program quality with pair programming in CS1, *Proceedings of the 9th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 176-180, 2004.

[3] Hanks, B., Student performance in CS1 with distributed pair programming, *Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 316-320, 2005.

[4] Hanks, B., Student attitudes toward pair programming, *Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 113-117, 2006.

[5] Jacobson, N., Schaefer, S. K., Pair programming in CS1: overcoming objections to its adoption, *SIGCSE Bulletin,* 40, (2), 93-96, 2008.

[6] LeJeune, N., Critical components for successful collaborative learning in CS1, *Journal of Computing Sciences in Colleges*, 19, (1), 275-285, 2003.

[7] McDowell, C., Werner, L., Bullock, H., Fernald, J., The effects of pair-programming on performance in an introductory programming course, *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, 38-42, 2002.

[8] Mendes, E., Al-Fakhri, L., Luxton-Reilly, A., A replicated experiment of pair-programming in a 2nd-year software development and design computer science course, *Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 108-112, 2006.

[9] Mendes, E., Al-Fakhri, L. B., Luxton-Reilly, A., Investigating pair-programming in a 2nd-year software development and design computer science course, *Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 296-300, 2005.

[10] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S., Improving the CS1 experience with pair programming, *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 359-362, 2003.

[11] Williams, L., Lessons learned from seven years of pair programming at North Carolina State University, *SIGCSE Bulletin*, 39, (4), 79-83, 2007.