**Pair Programming based on Computer Programming Self-Efficacy in CS1**

Jun Rao

[jr2339@nau.edu](mailto:jr2339@nau.edu)

Northern Arizona University

July 2018

Advisor

James Palmer, Ph.D.

**Abstract**

This paper describes the new pair programming method in the earliest days of the first programming course in computer science. Based on the score of the computer programming self-efficacy, instructors placed students in level-matched for a portion of their programming exercises. Students (novice programmer) who have the lower score of the computer programming self-efficacy showed significantly better exercise completion rates when paired than working individually. Students response to the pair programming which based on computer programming self-efficacy is positive. These data indicate that successful pair programming can be constructed based on the computer programming self-efficacy in the first class in the computer science course. Thus Pair Programming based on the score of the computer programming self-efficacy can be an active classroom intervention in computer science course.

Keywords: Pair Programming, self-efficacy, computer science, novice programmer

**INTRODUCTION**

Pair Programming is widely used in the in the first days or weeks of the semester to help students to start a programming course efficiently in computer science course I. Pair programming is a formal software development protocol which consists of two programmers working side-by-side at the same keyboard, continuously collaborating on the same design, algorithm, code or test Although pair programming originated in the industry and had become a vital practice of the Extreme Programming (XP) development methodology, we focus here on its use in educational settings In 2006, McDowell, Werner, Bullock, and Frenal found that student in classes that used Pair Programming can improve the learning outcomes

Some benefits of the Pair Programming in class can be implemented merely through random pairing. For example, in 2007, Williams statemented the interests of the Pair Programming for both teachers and students in a variety of Computer Science course at a large university For students, Pair Programming supports the building of stronger social relationships (through the need to work together), increase retention, and reduces the waiting time for teacher feedback. For teachers, Pair Programming reduces marking time (by halving the number of submitted assignments), reduces student demand in practical sessions, and improves general work ethic by, they hypothesis, engendering a sense of mutual responsibility between partners.

Williams also has developed that the careful selection of pair reduces the probability of dysfunctional pairing. They paired students in various combinations and found that the most successful pairings were those based on similar mid-term exam score. However, we are interested in using the Pair Programming in the earliest days to reduce the alarmingly high failure rate in first computer programming course; it will be too late to pair after mid-term exam

Most recent research (Krissi, 2003) used the pair programming method in the most initial weeks of the Computer Science I based on the instructor subjective observation of student performance in the first weeks of the semester and pointed out that when the two members of a pair have similar levels of programming ability, the educational benefits and satisfaction are the maximum However, the measurement of the student’s programming skill which is the instructor subjective observation remains problematic.

These studies have used metrics such as mid-term exam scores or the instructor subjective observation of student performance to estimate the programming skill of students, but to our knowledge, no other researchers have directly assessed programming performance with computer programming Self-Efficacy. The computer Programming Self-efficacy is an individual's belief in the student innate ability to achieve the programming goals, which can be used in the first days of the semester to estimate the programming ability in the computer science course. Thus, in our research, the observation assessment of the student’s programming ability is based on the computer programming Self-Efficacy Scale.

In 1997, Bandura statemented that self-efficacy is one psychological concept which evaluates an individual’s mental state and refers to individual beliefs in different situations Moreover, Bandura also explored that self-efficacy influences students’activity choice, including how much effort or time they will invest in solving particular tasks and situations

In 1998, Ramalingam and Wiedenbeck developed some questions to examine novice student’s self-efficacy in learning the C++ programing language In 2015, Base on Ramalingam and Wiedenbeck’s scale, Govender, Desmond Wesley, and Sujit Kumar modified the questions to examine student’s self-efficacy for the Java Language However, most of these previous researches focused on specific programming language rather than general computer programming self-efficacy.

In 2018, Tsai developed a more general self-efficacy scale, based on Berland and Lee’s computational thinking frameworkThis new Computer Programming Self-Efficacy Scale (CPSES) is excepted to be beneficial for all students above middle school levels, which includes five subscales: Logical Thinking, Algorithm, Debug, Control, and Cooperation. The research also confirmed the positive correlation between computer programming experience and computer programming self-efficacy

Thus, following the current best practice for the implementation of Pair Programming in the classroom, this study aimed to develop a pair programming method based on more objective measurement based on Krissi’s pair programming method and Tsai’s computer programming Self-Efficacy scale. This approach allowed the instructor to directly compare the achievement of the programming abilities of students who used pair programming based on computer programming Self-Efficacy in the lab with those of students who programmed randomly in the lab.

The primary contribution of our study is a direct assessment of the effects of the new pair programming method based on the student’s computer programming Self-Efficacy which can be used on the first days of the semester.

**Literature Review**

Randomly paired, careful selection of pairs (paired with some constraints) are reviewed separately here due to the tremendous differences in the two. In this study, they will be compared against one another to cross-analyze the differences and similarities in how the students are performed in CS1 classes depending upon their type of pairing.

**Randomly Paired**

Pair Programming is a formal software development protocol in which two programmers work synchronously on a single piece of code (Williams and Kessler, 1998). Studies have shown that randomly paired programming method can contribute to an improvement in learning outcomes. Randomly Paired Programming is widely used to help students to start a programming course efficiently in computer science course I. Research (McDowell, Werner, Bullock, and Fernald, 2004) found that students in pair programming classes were more likely to complete their course and to continue in a computer science major than were students in comparable classes that used only solo programming.

Some benefits of pair programming in a class can be implemented merely through random pairing. For example, in 2007, Williams statemented the interests of the randomly paired programming for both teachers and students in a variety of Computer Science course at a large university For students, randomly paired programming method supports the building of stronger social relationships (through the need to work together), increase retention, and reduces the waiting time for teacher feedback. For teachers, randomly paired programming method reduces marking time (by halving the number of submitted assignments), reduces student demand in practical sessions, and improves general work ethic by, they hypothesis, engendering a sense of mutual responsibility between partners.

Although there are some benefits of pair programming can be obtained merely by using the randomly paired method (e.g., McDowell et al., 2004 used only random pairing). However, there is convincing evidence that careful selection of pairs reduces the likelihood of dysfunctional pairings. In particular, when two members of a pair have similar levels of programming level, both educational benefit and student satisfaction appear to be maximized.

**Careful selection of pairs**

In the long-term research described by Williams (2007), teachers tried various indicators to determine the pairing method, including standardized general test scores, grade point average, the outcome of personality tests, learning style scores, and work ethic (based on self-report). The research team paired students in various combinations of these measures, using both similarity and the dissimilarity of scores. The most successful pairings are those students who have the similar mid-term exam score, the most direct measure of a student’s programming skill at the time of the pairing.

Some researches (e.g., Radermacher and Walia, 2011 and 2012; Braught, Wahls and Eby, 2008) have accepted that pairing based on skill level as the appropriate default, citing the accumulating evidence in its favor.

There is a growing consensus that the pairing method based on skill level produce the most successful result in the pair programming experience. However, the measurement of skill remains problematic in the pairing process. Because we are interested in using the pair programming method very early --- ideally, in the first few days in the class---we need a way to measure the skill ability before the exam or major project scores are available.

While much literature exists about the pair programming method based on the skill level (or programming ability), unfortunately, the measurement of the skill level (or programming ability) is still a problem in these research. Reviewing the literature leads back to the question: how could teacher measure the programming ability objectively? And which pairing programming could be used in the first days of the class?

**Methodology**

To answer the given research questions, ‘computer programming self-efficacy’ would be most appropriate to find an answer. The literature on related topics suggests that pair programming based on computer programming self-efficacy are most appropriate (Tsai, Meng-Jung, Ching-Yeh Wang, and Po-fen Hsu. "Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education." Journal of Educational Computing Research, 2018). This would aim to aid the researcher to solve the measurement of the skill level (or programming ability). Specifically, objectively measuring, to be able to get accurately skill level (or programming ability) for each student.

This measurement which is Computer Programming Self-Efficacy Scale (CPSES), for all students above middle school levels. Based on Berland and Lee’s computational thinking framework, and the instrument (CPSES) included the five subscales: Logical Thinking, Algorithm, Debug, Control, and Cooperation.

In the first day of each class section, the instructor administered class practica (computer programming self-efficacy test to be completed under quiz conditions) that all students completed individually. This test allowed the instructor to collect the computer programming self-efficacy score for all students and analysis it at the end of the semester. Table 1 presents all questions on the computer programming self-efficacy test. During the semester, all the students were randomly paired, and they did not know the computer programming self-efficacy score of their partner for each assignment in the lab. All assignments in the lab are group assignments. This approach also allowed the instructor to directly compare the effects of the student who paired with different programming self-efficacy score partner. Meanwhile, we did not allow the students to transfer between sections after the start of the course for the whole semester.

Our essential instrument for measuring student learning was the final grade include successful completion of the course (final grade higher than or equal to 70%), class participation (feedback by the partner), and assignment scores. All assignments were submitted and graded for correctness using the NAU BBLearn system. Students were allowed to provide their work as often as they wished within the one week for each assignment. BBLearn system was also used for the other classes in university, so students were familiar with the submission process and with interpreting BBL feedback. The instructor automatically scored all the assignments by using collaboratively developed grading criteria and frequent consultation on questionable cases to ensure consistency. For all sections of the research, students are not allowed to share the assignment to the other sections. And the instructor will not grade their homework until next week. As this strategy, there was no evidence of information flow between sections or other cheating. At the end of the semester, a separate analysis was performed for each student of this study.

1. I can understand the basic logical structure of a program.

2. I can understand a conditional expression such as ‘‘if ... else ...’’

3. I can predict the final result of a program with logical conditions.

4. I can predict the result of a program when given its input values.

5. I know programming work can be divided into sub-tasks for people.

6. I can work with others while writing a program.

7. I can make use of divisions to enhance programming efficiency.

8. I can figure out program procedures without a sample.

9. I don’t need others’ help to construct a program.

10. I can make use of programming to solve a problem.

11. I can open and save a program in a program editor.

12. I can edit and revise a program in a program editor

13. I can. I can run and test a program in a program editor.

14. I can find the origin of an error while testing a program.

15. I can fix an error while testing a program.

16. I can learn more about programming via the debugging process.

Table 1: Computer Self-efficacy Test

**References**

1.Williams, Laurie A., and Robert R. Kessler. "All I need to know about pair programming I learned in kindergarten." Communications of the ACM 43.5 (2000): 108-114.

2. BECK, K. and Andres, C. 2004. Extreme Programming Explained: Embrace Change 2nd, Ed. Addison-Wesley Professional.

3. McDowell, Charlie, et al. "Pair programming improves student retention, confidence, and program quality." Communications of the ACM 49.8 (2006): 90-95.

4. Williams, Laurie. "Lessons learned from seven years of pair programming at North Carolina State University." ACM SIGCSE Bulletin 39.4 (2007): 79-83.

5. Bennedsen, Jens, and Michael E. Caspersen. "Failure rates in introductory programming." ACM SIGcSE Bulletin 39.2 (2007): 32-36.

6. Wood, Krissi, et al. "It's never too early: pair programming in CS1." Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136. Australian Computer Society, Inc., 2013.

7. Bandura, Albert. Self-efficacy: The exercise of control. Macmillan, 1997.

8. Bandura, Albert. "Self-efficacy: toward a unifying theory of behavioral change." Psychological Review 84.2 (1977): 191.

9. Ramalingam, Vennila, and Susan Wiedenbeck. "Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy." Journal of Educational Computing Research 19.4 (1998): 367-381.

10. Govender, Desmond Wesley, and Sujit Kumar Basak. "An investigation of factors related to self-efficacy for java programming among computer science education students." Journal of Governance and Regulation (2015): 612.

11. Berland, Matthew, and Victor R. Lee. "Collaborative strategic board games as a site for distributed computational thinking." International Journal of Game-Based Learning (IJGBL) 1.2 (2011): 65-81.

12.Tsai, Meng-Jung, Ching-Yeh Wang, and Po-fen Hsu. "Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education." Journal of Educational Computing Research (2018).