**Ph.D. Research Proposal**

Doctoral Program in Informatics and Computing

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

**1.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.

**2.Methodology**

In Fall 2018, researchers at Northern Arizona University (NAU) collected data on the effects of different pair-programming methods on the lab sections at CS122L (Matlab), CS126L (Python), and CS136L (Java). These three courses serve a diverse population. CS122 is the first programming course in the computer-related major or engineering major, CS126 is the first programming course in computer science major, and CS136 is the second programming course in computer science major. Students also take CS126 and CS136 as a way to fulfill two of NAU’s general education requirements which is laboratory science and quantitative reasoning. CS122 meets for fifty minutes of lecture two times a week, and a two-hour lab period once a week. CS126 and CS136 meet for one hour and fifty minutes of class two times a week, and a one and half -hour lab period once a week.

In our study, we offered eight sections of the CS122 lab, four sections of the CS126 lab and two sections of CS 136 lab. In the first day of each lab section, the instructor administered laboratory 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.

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

After each week, students completed a brief questionnaire covering their attitudes toward the assignment process for a previous week. The surveys were submitted anonymously and were administered by a non-teaching member of the research team. The feedback questions asked are shown in Table 2.

1. How do you rate your learning this week?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

2. How do you your rate your experience with your partner in this assignment?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

3. How do you rate your partner's contribution to this assignment?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

4. How do you rate your partner's contribution to your learning in this assignment?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

5. Do you feel you work better in a programming pair than on your own?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Solo Pair

6.Any other comments?

Table 2: Feedback questionnaire in each week

Our essential instrument for measuring student learning was the final grade include successful completion of the lab course (final grade higher than or equal to 70%), lab participation (feedback by the partner), and lab 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 NAU, 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.

**3 General Discussion**

3.1 Threats to Validity

Two differences may threaten the validity of our results.

The first potential threat element in our research is the instructor because the different instructor has a different teaching style, which means the instructor element in the study is too subjective. To control for instructor effects, we used a collaboratively developed set of lecture notes and examples, and all assignments and exams were the same cross sections in the same course. At the same time, we used a detailed rubric for grading both written and laboratory practica, with frequent consultation on cases not directly covered by the rubric. And all instructor meet each other to discuss all the situation in the lab every week. In addition, because each instructor in the lab taught one matched section based on self-efficacy and one randomly paired section, our statistical analysis was able to detect differences by the instructor, as well as interactions of instructor with student ability (measured by computer self-efficacy scores), section type (paired based on self-efficacy vs. randomly paired) and course level.

The second potential threat is that the lab practica were administered to the control and paired sections on different days, which could allow information flow between the sections (i.e., some students may have more time to finish the assignment). For all sections of the research, students are don’t allowed to share the assignment to the other sections. Students can submit the job within one week multiple times, and the instructor will not grade their homework until next week.

**4.Analysis**

In this section, we consider the effects of paired-type (randomly paired with the similar computer programming self-efficacy score vs. randomly paired with the different computer programming self-efficacy score),course level (CS122L, CS126L, and CS136L),the mean score of lab participation (feedback by the partner), the mean score of assignments and successful completion of the course with a grade of C (70%) or higher.

**4.1 Preliminaries**

During the research, a total of 240 students enrolled in CS122, a total of 80 students enrolled CS126, and a total of 80 students enrolled CS136. All of these students consented to participate in the study. Students who did not complete the course or did not provide consent were excluded from the analysis in the research.

Before looking at the effects of paired-type and course level, we examined the computer self-efficacy scores of the 200 students constituting our sample. We used an ANOVA (Analysis of Variance) to test for differences in computer self-efficacy score between students in different sections. Based on the ANOVA, we can conclude that there is insufficient evidence for a difference in computer self-efficacy scores across the course level or the sections.

Based on this analysis we proceeded under the assumption that the initial populations were similar when divided by section. Further tests were performed to determine if the paired-type were reasonable predictors of performance in our course. A binary response ANCOVA (i.e., logistic regression analysis of covariance) using the Starters indicated that successful completion of each section (final grade >= 70%) was positively correlated with the paired-type (randomly paired with the similar computer programming self-efficacy score). At the same time, a continuous response ANCOVA using the Completers provided evidence of a positive correlation between randomly paired with the similar computer programming self-efficacy score and the mean score of laboratory practice, the mean score of laboratory assignments, and final grades. Given the association with the paired-type (randomly paired with the similar computer programming self-efficacy score) of performance on the majority of course components, our later analysis of these metrics included computer self-efficacy score as a covariate.

**4.2 Performance on Lab Practice**

Since the mean score of lab practica can be regarded as a continuous variable, we performed a continuous-response ANCOVA (Analysis of Covariance) to investigate the effect of the paired-type and the mean score of lab practica. The first step of this analysis determined if there were any interactions between the paired-type and the mean score of lab practica. This analysis revealed evidence (α = 0.05) of communication between paired-type and lab practica scores. The interaction between paired-type and the mean of lab practica scores indicates that the linear relationship between SAT scores and mean lab practica score in all sections. Figure1 illustrates this interaction by fitting lines to the data from all classes in the research.

**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).