The effect of self-efficacy and pair programming experience in learning results of introductory programming courses

Yifan Mei
Department of Statistics,
UW-Madison
Madison, USA
ymei8@wisc.edu

Heng Ping
iSchool,
UW-Madison
Madison, USA
hping2@wisc.edu

Mingren Shen
Biophysics,
UW-Madison
Madison, USA
mshen32@wisc.edu

ABSTRACT

The purpose of this study was to explore the interactive effect of self-efficacy and pair programming experience to the final learning results in introductory programming courses. We developed a 2x2 fractional design to explore their roles and relationships. Data was collected by distributing questionnaires to students have learnt or are learning CS367 at UW-Madison. They were asked to evaluate their self-efficacy levels and pair programming experience. After that, they needed to complete a quiz of 11 Java knowledge quiz indicating their learning results. We present results from 36 participants which show that students with high self-efficacy levels tended to earn a higher score in the Java knowledge quiz. However, pair programming experience shows no significant effects on learning results.Our finding suggests that high self-efficacy levels have a positive impact on students' performance in introductory programming courses.

ACM Classification Keywords
H.5.m. [Information Interfaces and Presentation (e.g. HCI)]: Miscellaneous; K.3.2. [Computer and Education]: Computer and Information Science Education

Author Keywords
Human-Computer Interaction; Self-efficacy; Pair Programming; Collaborative Learning

INTRODUCTION

Computer education is important for the development modern society. According to data from White house [1], there were more than 600,000 high-paying computer science-related jobs across the United States that were unfilled in 2016. Computer science and data science are not only important for the technology companies, but also for so many industries, including transportation, health-care, education, and financial services.

[1]https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Although both global governments and companies like Facebook, Google and Microsoft have been spending large amount of energy ,money and human resources for computer science education throughout the past few years, the dropout and failure rates in introductory programming courses are still very high. Some studies even suggest that the dropout and failure rates could be as high as 30 percent [5]. However, for some, learning programming is both enjoyable and motivating while others find it a miserable struggle they fail to complete.

There have been many studies of the factors that may influence beginners' success in introductory programming course [3, 18]. These factors include previous computing experience [6], computer self-efficacy [7], mathematics or science background [18], teaching method like pair-programing [9, 4] and student personal attributes like learning style and mental models [11, 18].

The goal of this project is to investigate the effect of self-efficacy and pair programming experiences to the study results of introductory level programmers, explore the relationship between these two very important concepts and study their interactions on students' learning performance.

RELATED WORK

According to Bandura, self-efficacy is defined as

> People's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives [1].

Researchers have recognized self-efficacy as an essential enhancement of learning motivations [19]. Scales of measuring self-efficacy are developed to assess people's self-efficacy levels [15]. In a two-year research conducted in North Carolina State University, a pair programming experiment was executed to evaluate students' self-efficacy in an introductory computer science course [16]. The result shows that positive self-efficacy perception would encourage students to perform more actively in programming practice [8].

Previous research has shown that working in pairs would enable students to improve their programming skills as well as final scores than individually programming learning [17, 9, 10]. Therefore, as a proven efficient pedagogy methodology, pair programming is widely utilized in introductory computer courses in the Uinted States. Yet there are still issues that might cause pair programming to fail. For example, in a

Microsoft paper, cost-efficiency and personality conflicts are reported as two top questions disturbing pair programming [2].

Self-efficacy is recognized as a significant internal factor that influencing new skill learning [14] and pair programing is believed a good learning method for introductory courses [9, 4]. However, the interaction between self-efficacy and pair programming has not been assessed, and this is the question we interested most in our project.

METHODOLOGY

Our design is a two way between - participants design. We collect our data by distributing questionnaires, each of which has 3 parts to measure self-efficacy levels, pair programming experience and learning results of CS introductory programming courses respectively. The participants are students who have learned or are learning CS367 at UW-Madison. We use 2 x 2 factorial design to study the influence of self-efficacy and pair-programming experience on the learning results of CS introductory programming courses. The two independent variables, self-efficacy level and pair programming experience are both categorical variables with 2 levels, low (bad) or high (good).

The model for the fractional design is shown as following:

$$Y_{ijl} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijl} \tag{1}$$

In equation (1), Y represents the dependent variable, the learning results of CS introductory programming course. In our project, this dependent variable is measured by scores from a Java knowledge quiz. α is the first independent variable, self-efficacy levels; β is the second independent variable, pair programming experience; $(\alpha \times \beta)$ represents the interaction term of self-efficacy levels and pair programming experience. The meaning of subscripts in equation (1) is given below:

- i = 1, 2; representing the 2 levels of self-efficacy, low and high;
- j = 1, 2; representing the 2 levels of pair programming experience, bad and good;
- l = 1, 2,..., 9; representing the 9 individual participants in each 2 x 2 block
- $\varepsilon_{ijl} \sim N(0, \sigma_{\varepsilon_{ijl}})$; representing the error term of the model

And the hypotheses for our designs are as following:

1. H1: The higher the self-efficacy levels, the better the learning results of CS introductory programming courses.
2. H2: The better the pair programming experience, the better the learning results of CS introductory programming courses.
3. H3: High self-efficacy but good pair programming still leads to good learning results.
4. H4: Low self-efficacy but bad pair programming leads to bad learning results.

MEASUREMENT

The measurement in our study included a self-efficacy self-evaluation, a pair programming experience survey and a Java knowledge quiz.

Self-efficacy levels was measured using the items and key points from International Personality Item Pool(IPIP)[2] and questions from previous studies [12, 18]. The scale of self-efficacy contains 10 questions helping participants judge their self-efficacy in a wide range of programing tasks and situations, e.g. "I felt confident in my ability during the programming tasks.", "I could come up with good solutions when programming." and "I do not see the consequence of my program",etc. Responses are on a 7-points Likert scale, where 1 means strongly disagree, and 7 means strongly agree.

Participants' pair programming experience was evaluated by using questions used by Salleh et al. [13]. The purpose of their study was to investigate the effects of personality traits on pair programming and we modifying their questions to suit our purpose to test the participants' pair programming satisfaction scale based on their personal experience. Sample questions are like "My motivation level increased when working with my partner(Working with a partner makes it easy to understand what I am doing and why I am doing it)." ,"My level of confidence in solving the exercises increased when working with my partner." and "Do you switch roles working in pair programming(change âĂŸdriverâĂŹ and âĂŸnavigatorâĂŹ fairly regularly)?", etc. We also used 7-points Likert scale survey items ranging form 1 (bad) to 7 (good).

For measuring the learning results of CS introductory programming course, we prepared 11 multiple questions from Java Review website[3]. Java Review website is a wonderful, free and open website providing questions for Java learners to test their Java knowledge. It is created by Prof. Barbara Ericson at Georgia Tech. The Java knowledge quiz contains questions from different topics like "Java Basics","Classes and Objects" and "Strings", etc. Within each topic, the questions are separated by 3 different levels: basic, intermediate, and hard. In our design, around 70% of the questions are chosen from the basic level, around 20% of the questions are chosen from the intermediate level, and the rest question is hard level.

Procedure

This study was conducted during the course of CS 770 in UW-Madison. And we distributed the questionnaire that contains 3 parts of the above 32 questions by Google form[4]. All participants are required to finish all the 3 parts of the survey which normally took 20 to 30 minutes.

We collected 36 valid questionnaires. Then, we summed up each students' the points for each part respectively. For the self-efficacy level and pair programming experience, we transformed the raw points in the questions to get the self-efficacy

[2]http://ipip.ori.org/newindexofscalelabels.htm

[3]http://interactivepython.org/runestone/static/JavaReview/index.html

[4]https://docs.google.com/forms/d/e/1FAIpQLSf4tpQ5G2PpgMqj0he0EpliJaHNlK5zVPaoBMCx2fbIy6Q9xA/viewform?usp=sf_link

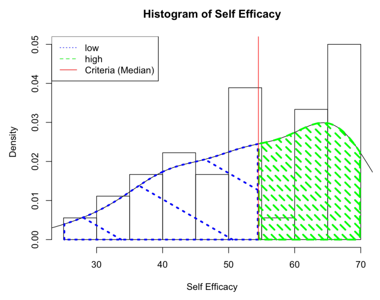

Figure 1. Distribution of self-efficacy points and changing self-efficacy points into two factor levels low and high

or pair programming experience scale points. For negative workload items, we converted the value using the max value(7 in our scales) - raw point.

Then, based on the summarized self-efficacy scale points, we first transformed the sum of self-efficacy points into two levels, low and high. To do so, we sorted the sum points of self-efficacy for the 36 participants by ascending order and treated the top 18 participants as low level self-efficacy participants, while the rest 18 participants as high-level self-efficacy participants as shown in Fig.1.

Then, we sorted the 2 groups of low and high self-efficacy(18 participants each) in increasing order separately by the sum of pair programming experience scores, and treated the top 9 of the participants in each group had bad pair programming experience, while the rest 9 of each group had good pair programming experience as shown in Fig.2.

RESULTS

After data collection, we got a 2x2 factorial design data set, where the first independent variable is self-efficacy with 2 levels: low and high, while the second independent variable is pair programming experience with 2 levels: bad, good. Within each 2 x 2 block, there were 9 participants. What we could do next is to make 2 bar-plots for the 2 treatments, and get some general information from the bar-plots.

According to the Fig.3, we can see that the each subgroup has a similar standard error. Besides, people who have relatively bad pair programming experience get higher scores no matter their self-efficacy was high or low, while high self-efficacy resulted in high scores.

After we got some sense about the main effects of the model, our next step was to make a general plot to see whether there were any potential interactions between the 2 independent variables, self efficacy levels and pair programming experience. The results are shown in Fig. 4.

According to the interaction detecting plots in Fig.4, there was no obvious interaction existed in any of the graph. Besides, the interaction plots showed the same result as bar plot, where no matter self-efficacy was high or low, people with worse pair programming experience have higher scores, and no matter

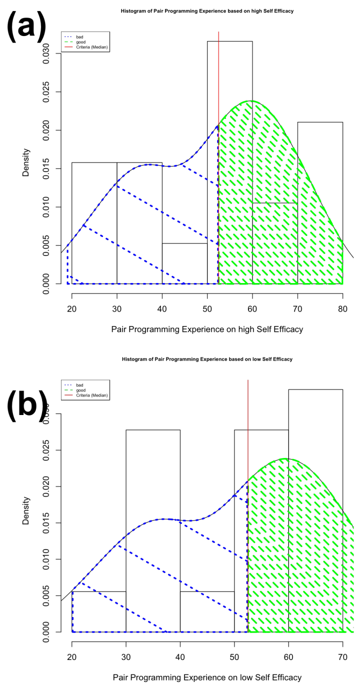

Figure 2. (a)Distribution of pair programming experience points in high self-efficacy group and changing pair programming experience points into two factor levels bad and good.(b)Distribution of pair programming experience points in low self-efficacy group and changing pair programming experience points into factor levels bad and good

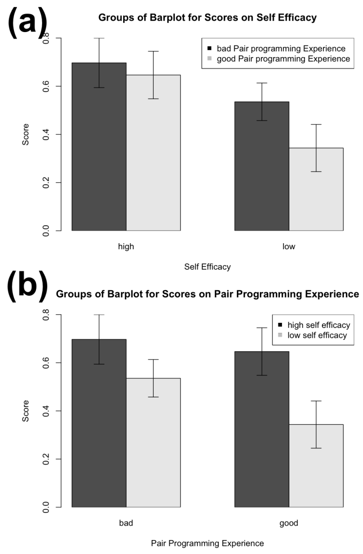

Figure 3. (a)Barplots for scores on self-efficacy groups. (b)Barplots for scores on pair programing experience groups.

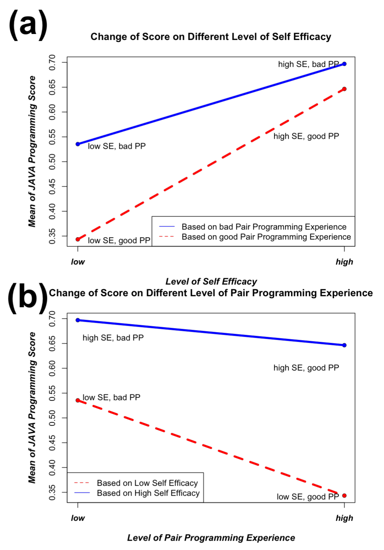

Figure 4. (a)Influence of different self-efficacy levels. (b) Influence of different levels of pair programing.

pair programming experience was good or bad, higher self-efficacy leaded to better scores.

Now, after some general views of the main effects and interaction effects, we calculated the ANOVA table to get a more precise test about the interactions and relationships between the 2 independent variables. The result was shown in Table.1 .

Table 1. ***Two Way Between-Participants ANOVA Table of self-efficacy and pair programming experience***

Factors	*df*	*SS*	*MS*	*F*	*p*
SE	1	0.49	0.49	6.00	0.020*
PPE	1	0.13	0.13	1.63	0.21
SE & PPE	1	0.045	0.045	0.56	0.46
Residuals	32	2.59	0.081		

Signif. codes 0 *** 0.001 ** 0.010 * 0.050 . 0.10 1

SE : self-efficacy; PPE : pair programming experience

SE & PPE : interactions between self-efficacy and pair programming experience.

Based on the ANOVA table(Table.1), we could find that we only had weak evidences to say that the self-efficacy levels have positive influences on Java knowledge quiz scores meaning that we had weak evidences to say that the higher the level of self-efficacy, the better the learning result of CS introductory programming courses. As for the experience of pair programming or the interactions between pair programming experience and self-efficacy levels, we did not have any evidence to say they had influences on the learning results of CS introductory programming courses.

The result here partly fitted our hypothesis. For the four hypotheses we proposed, only H1 was supported by our data which meant that the higher the self-efficacy levels, the better the learning results of CS introductory programming courses.

DISCUSSION

self-efficacy

According to previous research results [11], self-efficacy has been recognized as a motivation factor in college-level learning. High self-efficacy level leads to strong desire in competition with other students and inspires students to earn higher scores in the Java programming quiz which agrees with our project results.

So in general, we can propose that self-efficacy is more important for people to perform well in introductory programming courses.

pair programming experience

It seems that higher level of pair programming experience did not contribute to a high score in the Java programming quiz. Many factors might lead to this results. Firstly, to full utilize the power of pair programming, all learners should receive proper training, such as switching roles frequently and the steering keyboard and mouse alternatively. Such a complex human-human interactions like pair programming requires both parties put great effort to communicate, discuss and cooperate with each other.

Secondly, some participants with advanced programming background usually do not rely on pair programming or cooperate well with their pair programming partner. In another word, the individual learning process is efficient and effective enough for them. This could be attributed to sampling bias when distributing questionnaires.

Finally, pair programming is aimed at improving studentsâĂŹ coding ability instead of their examination scores [4]. However, as we used a multiple choice Java knowledge quiz to assess the learning results, there could be a possibility that some participants are not good at doing multiple choice quiz but good at coding.

CONCLUSION

Our study focused on testing the main effects and interaction effects of self-efficacy level and pair programming experience to the learning result of CS introductory programming courses. We conducted a 2 x 2 factorial design for the design, and we found out that self-efficacy was the only significant factor that influencing the final learning results, whereas pair programming experience and the interaction effects of self-efficacy and pair programming experience had no significant influence on the final learning results.

ACKNOWLEDGMENTS

We thank all the volunteers, all classmates and professors who supported and provided helpful suggestions on our project.

We especially want to thank Prof. Mutlu at UW-Madison for instructing our whole project and helpful suggestions and discussions, Prof. Legault at UW-Madison for helping us prepare the questionnaire used in our study and Prof. Deppeler at UW-Madison for helping us recruit the volunteers in CS 367.We also would like to thank Prof. Barbara Ericson at Georgia

Tech for the Java questions we used from her wonderful Java Review Website[5].

REFERENCES

1. Albert Bandura. 1997. *Self-efficacy: The exercise of control*. Macmillan.
2. Andrew Begel and Nachiappan Nagappan. 2008. Pair programming: what's in it for me?. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 120–128.
3. Susan Bergin and Ronan Reilly. 2005. Programming: factors that influence success. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 411–415.
4. Carolina Alves de Lima Salge and Nicholas Berente. 2016. Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research?. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE, 5398–5406.
5. Mark Guzdial and Elliot Soloway. 2002. Teaching the Nintendo generation to program. *Commun. ACM* 45, 4 (2002), 17–21.
6. Dianne Hagan and Selby Markham. 2000. Does it help to have some programming experience before beginning a computing degree program?. In *ACM SIGCSE Bulletin*, Vol. 32. ACM, 25–28.
7. Rex Karsten and Roberta M Roth. 1998. Computer self-efficacy: A practical indicator of student computer competency in introductory IS courses. *Informing Science* 1, 3 (1998), 61–68.
8. Päivi Kinnunen and Beth Simon. 2011. CS majors' self-efficacy perceptions in CS1: results in light of social cognitive theory. In *Proceedings of the seventh international workshop on Computing education research*. ACM, 19–26.
9. Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin* 34, 1 (2002), 38–42.
10. Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2003. The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th international conference on Software engineering*. IEEE Computer Society, 602–607.
11. Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and mental models in learning to program. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 171–175.
12. Vennila Ramalingam and Susan Wiedenbeck. 1998. 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.
13. Norsaremah Salleh, Emilia Mendes, and John Grundy. 2014. Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. *Empirical Software Engineering* 19, 3 (2014), 714–752.
14. Daniel J Schumacher, Robert Englander, and Carol Carraccio. 2013. Developing the master learner: applying learning theory to the learner, the teacher, and the learning environment. *Academic Medicine* 88, 11 (2013), 1635–1645.
15. Mark Sherer, James E Maddux, Blaise Mercandante, Steven Prentice-Dunn, Beth Jacobs, and Ronald W Rogers. 1982. The self-efficacy scale: Construction and validation. *Psychological reports* 51, 2 (1982), 663–671.
16. Laurie Williams and Richard L Upchurch. 2001. In support of student pair-programming. In *ACM SIGCSE Bulletin*, Vol. 33. ACM, 327–331.
17. Laurie A Williams and Robert R Kessler. 2000. All I really need to know about pair programming I learned in kindergarten. *Commun. ACM* 43, 5 (2000), 108–114.
18. Brenda Cantwell Wilson and Sharon Shrock. 2001. Contributing to success in an introductory computer science course: a study of twelve factors. In *ACM SIGCSE Bulletin*, Vol. 33. ACM, 184–188.
19. Barry J Zimmerman. 2000. Self-efficacy: An essential motive to learn. *Contemporary educational psychology* 25, 1 (2000), 82–91.

[5]http://interactivepython.org/runestone/static/JavaReview/index.html