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 pair programming experience and self-efficacy to the final learning results in introductory programming courses. We developed a 2x2 fractional design to explore their roles. Data was collected by distributing questionnaires to students have learnt or are taking 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 programming problems indicating their learning results. Results show that students with high self-efficacy levels tended to earn a higher score in the Java programming quiz. However, pair programming 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 and advancement of modern society. According to data from White house [1] in 2016, there were more than 600,000 high-paying tech jobs across the United States that were unfilled, and by 2018, 51 percent of all STEM jobs are projected to be in computer science-related fields. Computer science and data science are not only important for the tech sector, but for so many industries, including transportation, health-care, education, and financial services. Although both global governments and companies are spending attention, energy and money 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 rate could be as high as 30 percent [5]. However, for some, learning programming is both enjoyable and motivating while other find it a miserable struggle they fail to complete.

There have been many investigations of the factors that may influence beginners' success in programming course [3, 17]. These factors include previous computing experience [6], computer self-efficacy [7], mathematics or science background [17], teaching method like pair-programing [9, 4] and student personal attributes like learning style and mental model [11, 17].

This goal of this project is to investigate the effect of self-efficacy and pair programming experiences to the success of elementary 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 [18]. Scales of measuring self-efficacy were developed to assess people's self-efficacy level in different disciplines [14]. In a two-year research conducted in North Carolina State University, a pair programming experiment was executed to evaluate students' efficacy in an elementary computer science course [15]. 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 [16, 9, 10]. Therefore, as a proven efficient pedagogy methodology, pair programming is widely utilized in introductory computer courses in the U.S. Yet there are still issues caused

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

CHI'14, April 26–May 1, 2014, Toronto, Canada.

pair programming to fail. For instance, in a Microsoft paper, cost-efficiency and personality conflicts were reported as two top questions disturbing pair programming [2].

Based on Schumacher et al, self-efficacy was recognized as a significant internal factor that influencing self-assessment skills which impacts learning skill development [13]. 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 manipulated by a between - participants two way design. We collect our data by distributing questionnaires, each of which has 3 parts to measure Self Efficacy Level, Pair Programming Experience, learning results of CS elementary programming courses respectively. The participants are students who have learnt or are taking CS367 at UW-Madison. We use 2 x 2 factorial design to study the influence of individual self-efficacy and pair-programming experience on the learning results of CS elementary 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:

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

In equation (1), Y represents the dependent variable, the learning results of CS elementary programming course. In our project, this random variable is measured by scores from a Java knowledge test. α is the first independent variable, self-efficacy level; β is the other independent variable, pair programming experience; $(\alpha \times \beta)$ represent the interaction term of self-efficacy level and pair programming experience. And details of subscripts is as follows:

- i = 1, 2; represent the 2 levels of Self Efficacy;
- j = 1, 2; represent the 2 levels of Pair programming Experience;
- l = 1, 2,..., 9; represent the 9 individuals in each small block
- $\varepsilon_{ijl} \sim N(0, \sigma_{\varepsilon_{ijl}})$

And the hypothesis for our designs as as following:

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

MEASUREMENT

To measure self-efficacy levels and pair programming experience, we set 10 questions for self-efficacy level, and 11 questions for pair programming experience. For all these questions, we set level 1 to 7, where 1 means strongly disagree, and 7 means strongly agree. Questions for self-efficacy are designed from International Personality Item Pool(IPIP)[2]. And questions for pair programming experience is compiled from [12].

For measuring the learning results of CS elementary programming course, we prepared 11 multiple questions from Java Review website[3], a wonderful, free and open website for Java learners created by Prof. Barbara Ericson at Georgia Tech. The Java knowledge separates the questions by different topics. 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.

We distributed the questionnaire which gathers above 32 questions by using Google form[4] and finally, we got 36 valid questionnaires. Then, we sum up each studentsâĂŹ the points respectively for each part. For the self-efficacy level and pair programing experience, we transformed the raw points in the questions which had negative attitude to 7 - raw point.

Then, based on the unchanging feature of self-efficacy, we first transform the sum of self-efficacy points into two levels, low, and high. To do so, we sort the sum points of self-efficacy for the 36 participants by ascending order, and then treat 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 sort the 2 groups of 18 participants in increasing order separately by the sum of pair programming experience scores, and treat the top 9 of the participants in each group have bad pair programming experience, while the rest 9 of each group have good pair programming experience as shown in Fig. 2.

RESULTS

Now, we get a 2x2 factorial design dataset, where the first treatment is self-efficacy with 2 levels: low and high, while the second treatment is pair programming experience with 2 levels: bad, good. Within each small block, there are 9 participants. What we can do next is to make 2 barplots for the 2 treatments, and get some general information from the barplots.

According to the Fig. 3, we can see that the each subgroup has similar standard error. Besides, people who have relatively bad pair programming experience get higher score no matter

[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

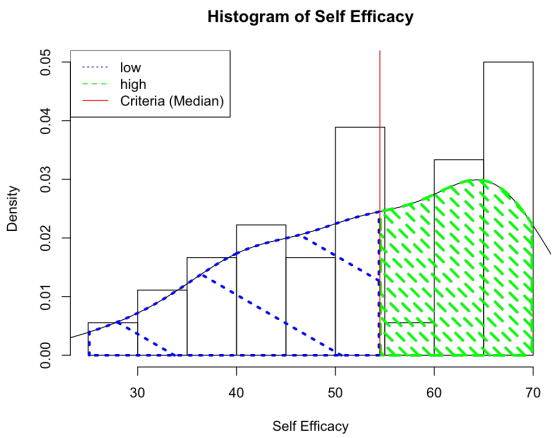

Figure 1. Dirstribution of self-efficacy points and changing self-efficacy points into factor levels low and high

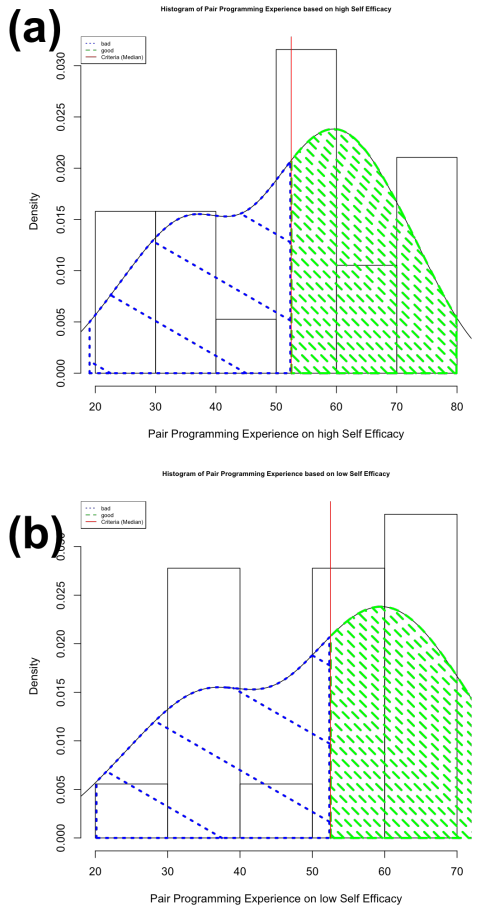

Figure 2. (a)Dirstribution of pair programming experience points in high self-efficacy group and changing pair programming experience points into factor levels bad and good.(b)Dirstribution of pair programming experience points in low self-efficacy group and changing pair programming experience points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

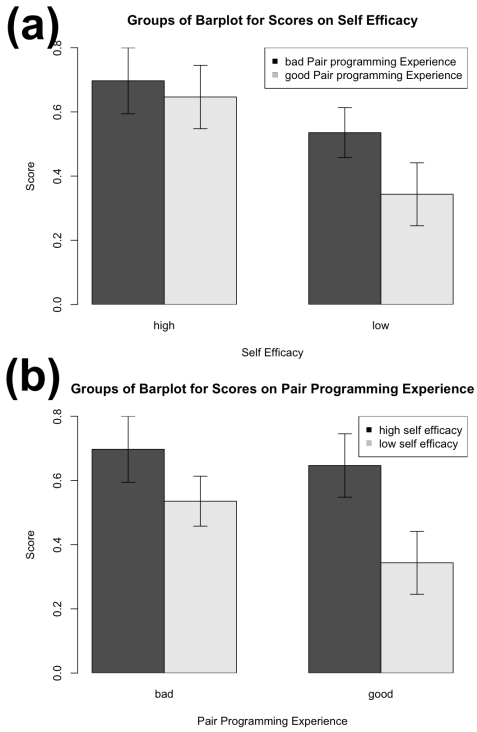

Figure 3. (a)Barplots for scores on self efficacy groups. (b)Barplots for scores on pair programing experience groups.

their self efficacy is high or low, while high self efficacy results in high score.

After we get some sense about the main effects of the model, our next step is to make a general plot to see whether there is any potential interaction between the 2 treatments, Self Efficacy, and Pair Programming Experience, and the visualizations are shown as in Fig. 4

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

Now, after some general view of the main effects and interaction effects, we calculate the ANOVA table to get a more precise test. The result is shown in Table. 1 .

Based on the ANOVA table above, we can see that we only have weak evidence to say that the self-efficacy level does have positive influence on Java programming scores, which means that we have weak evidence to say that the higher the level of self-efficacy, the better the learning result of CS elementary programming courses. As for the experience of pair programming or the interaction effect of pair programming experience and self-efficacy level, we do not have any evidence to say

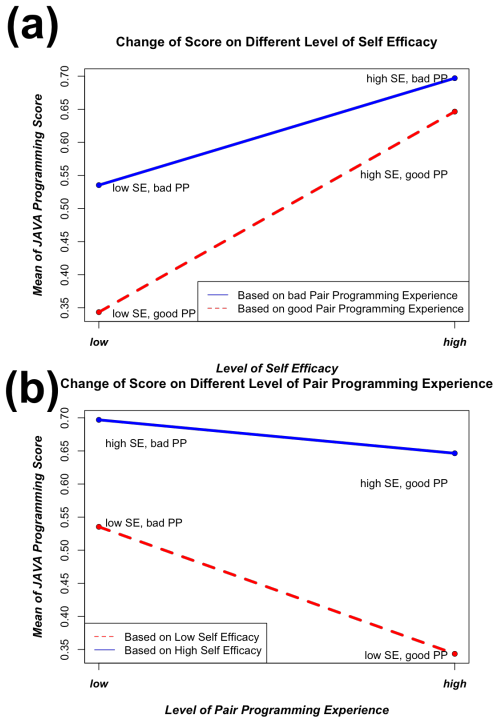

Figure 4. (a)Influence of different self-efficacy levels. (b) Influence of different levels of pair programing.

Table 1. ANOVA Table

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
SE	1	0.4858	0.4858	6.003	0.0199*
PPE	1	0.1322	0.1322	1.634	0.2103
SE & PPE	1	0.045	0.045	0.556	0.4613
Residuals	32	2.5895	0.0809		
Signif. codes 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1					

SE : self-efficacy; PPE : pair programming experience

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

they have influence on the learning result of CS elementary programming courses.

The result here partly fit our hypothesis.

DISCUSSION

Sample deviation in this project is not large enough probably because we selected snowball sampling for research convenience. As a result, majority of the participants were our friends and classmates. Another limitation in this research could be that some prerequisites of pair programming are not met. For instance, students work in pairs should have a similar level of knowledge and they must obey pair programming rules such as switching roles frequently. In addition, Some participants who have a more advanced background usually do not rely on pair programming to gain programming experience. For them, individual learning process is efficient and effective enough. Finally, pair programming may lead to good code but not good quiz performance, the latter one which is more dependent on people's exam ability difference.

CONCLUSION

Our study focuses on testing the main effects and interaction effects of self-efficacy level and pair programming experience to the learning result of CS elementary programming courses. We conducted a 2 x 2 factorial design for the design, and we find out that self-efficacy is 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 have no significant influence on the final learning results.

ACKNOWLEDGMENTS

We thank all the volunteers, and all publications support, classmates and staff, who wrote and provided helpful comments on previous versions of this document. We especially want to thank Prof. Mutlu for instructing our whole design of this study, Prof. Legault for helping us preparing the questionnaire and Prof. Deppeler for helping us recruiting the volutenners in CS 367. We would like to thank Prof. Barbara Ericson at Georgia Tech for the Java questions we used from hera 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.

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

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. 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.
13. 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.
14. 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.
15. Laurie Williams and Richard L Upchurch. 2001. In support of student pair-programming. In *ACM SIGCSE Bulletin*, Vol. 33. ACM, 327–331.
16. 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.
17. 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.
18. Barry J Zimmerman. 2000. Self-efficacy: An essential motive to learn. *Contemporary educational psychology* 25, 1 (2000), 82–91.