Cognitive load comparison of traditional and distributed pair programming on visual programming language

Chia-Yin Tsai
Department of Information and Learning Technology
National University of Tainan
Tainan, Taiwan

Chih-Kai Chang
Department of Information and Learning Technology
National University of Tainan
Tainan, Taiwan
chihkai@mail.nutn.edu.tw

Ya-Fei Yang
Department of Information and Learning Technology
National University of Tainan
Tainan, Taiwan

***Abstract*—Research results from computer science education show that pair programming is an effective teaching strategy on learning performance for computer science education in K-12. However, pair programming is not popularly used in K-12. One important issue is because sometimes pair programming by K-12 students is not as effective as expected. This study examines whether pair programming is effective from cognitive load perspective in distributed situation. We compares the cognitive loads for the visual programming language, called StarLogo TNG, under three circumstances that is alone, pair programming, and distributed pair programming. The experimental results show that learners prefer traditional pair programming to alone. Moreover, the pair programming can reduce germane cognitive load significantly. However, there is no significant difference between the single and distributed pair programming. Meanwhile, there is no significant difference on learning performance between traditional and distributed pair programming. Those results indicate the learning support and knowledge sharing tools for distributed pair programming could be improved.**

Keywords—computer science education; distributed pair programming; visual programming language; cognitive load; StarLogo TNG.

I. INTRODUCTION

Many educators realize that teaching computational thinking by programming design can bring learners additional benefits in information society. Various learning strategies for efficiently learning programming are also scrambling to emerge. For instance, learners can improve programming efficiency, enhance confidence, and improve programming correctness through collaborative learning. Proper collaborative learning activity can provide effective feedback according learners' competence or learning styles. Moreover, collaborative learning can sharpen programming skills and apply computational thinking in other areas. Pair programming comes from agile software development methods, are often used on program design courses. Pair programming works through two learners co-authored a program, including a program editor (driver) to write the program and the other as mentors (observer) is responsible for assisting.

Some literature indicated that pair programming can reduce the errors caused by misconceptions, improving the quality of programs, increase program editors' confidence, and increase program editors' efficiency [3][6][9]. In the literature of computer science education, there are positive reports of using pair programming as the teaching strategy [1][6][7]. For instance, learners with less experience have high satisfaction to work together with learners with more experience in pair programming. Moreover, pair programming is usually faster than separate programming on the same task. In other words, pair programming is useful for attaining high satisfaction and correctness when novice-expert pairing.

In pair programming, learners can reduce their programming misconceptions and deepen their understanding of how programs work through conversation. The strategy uses the concept of collaborative learning to make learners achieve higher effectiveness and avoid common mistakes with respect to study alone. However, pair programming does not always bring benefits. Figure 1 illustrates advantages and disadvantages of pair programming modes. When pairs have complementary strengths and play their strengths to achieve mutual benefits, they are in "Flow" state. When novice-expert pairing, novice, who will be in the "Learning" state, can develop his/her ability through the collaboration. Meanwhile, expert, who will be in the "Coaching" state, can guide novice to work together to increase efficiency through his/her knowledge [2][9].

In addition to aforementioned pair programming, visual programming language (VPL) is another viable way to increase learners' willingness for learning programming, improve

978-1-4673-8038-6/15 $31.00 © 2015 IEEE
DOI 10.1109/EITT.2015.37

Authorized licensed use limited to: California State University - Chico. Downloaded on September 09,2021 at 05:45:11 UTC from IEEE Xplore. Restrictions apply.

learners' confidence, and reduce learning frustration while programming. Rather than learning an esoteric text-based programming language, visual programming language enables learners to program effectively and efficiently without spending lots of time specifying the program textually. Visual programming language can facilitate learning by manipulating program elements graphically on a flowchart. The abstract concepts can be transformed into visual representations that help students develop debugging skills, observe variables, and trace logic, which leads to correcting their programs and solving problems [5].

Teach students how to use information technology to solve problems and self-learning is an important issue. Therefore, all the advanced countries in recent years, strongly advocated reform of primary and secondary computer science education through integrating computational thinking into the curriculum framework of computer science education in primary and secondary schools. To promote computational thinking, visual programming languages are used in new curriculum, such as the well-known Scratch, Alice, AgentSheets, and so on. Although studies have shown pair programming to be an effective teaching strategy in computer science education literature, pair programming is not much really used in secondary education. One major reason is that programming assignment is usually done after class, but pair programming activity is difficult to implement in distant situation.

Cognitive load can measure the total amount of a learner's working memory (i.e. mental effort). Hence, cognitive load is a very useful indicator for instructional design improvement. Heavy cognitive load generally have negative effects on learning performance although proper cognitive load is not the same for everyone. For instance, there are individual differences in cognitive processing capacities between novices and experts. Cognitive load has become a fundamental concept for instructional design to prevent the chances of learners' errors and mistakes in the last decade. Computer programming skills can be classified as recurrent and non-recurrent. Visual programming language can reduce the cognitive loads while learning recurrent programming skills.

StarLogo is an extension of the Logo programming language, which designed for students to model the behavior of decentralized systems. Specifically, StarLogo can create a model without an underlying equation, but can nonetheless be represented formally as agent-based simulation by programming. StarLogo TNG (The Next Generation) was released in 2008. It provides a 3D world and a visual programming language to increase ease of use and learnability. In other words, StarLogo TNG uses "blocks" to put together like puzzle pieces. StarLogo TNG allows users to use multi-agents for simulation. Multi-agent systems can be used for simulating transportation, logistics, GIS, science inquiry as well as in many other fields. The left part of the following figure shows the visual programming environment in StarLogo TNG, and the right part of the following figure demonstrates the 3D simulation world in StarLogo TNG.

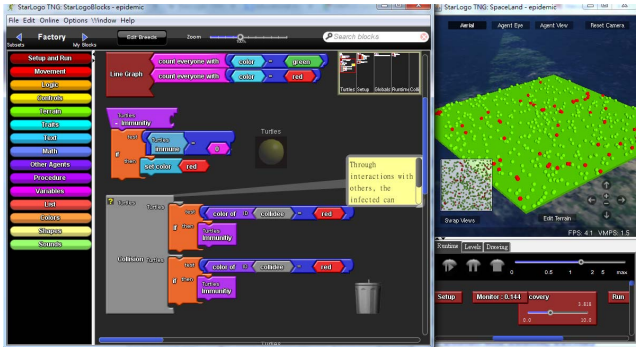

Fig. 1. StarLogo TNG for multi-agent simulation.

There is no other study comparing cognitive load of individual, face-to-face pair, and distributed pair programming on visual programming language except for Han, Lee, & Lee (2010). Similar with Han, Lee, & Lee's study, the agent-based simulation language StarLogo TNG was selected for our study. For this reason, this study required students to use StarLogo TNG software to complete the assignments, and programming in the following three situations: First, gather cognitive load data as the base period data when learners complete the program assignment individually. Second, learners were arranged in a computer classroom in the form of a traditional face-to-face pair programming and cognitive load data collected. Finally, students completed their assignment in distributed pair programming at home and we collected their cognitive load data again for analysis.

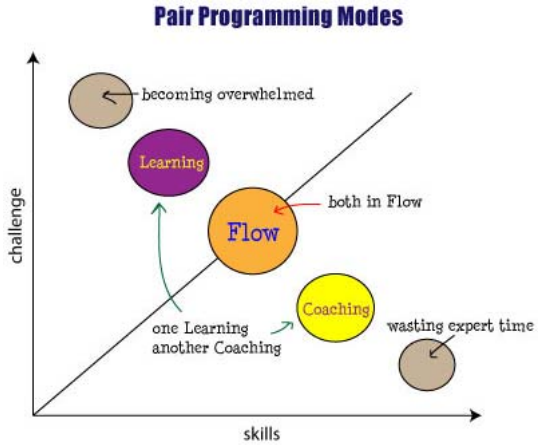

Fig. 2. Advantages and disadvantages of pair programming modes.

II. Experiment Design

The study will use the following two experimental research tools: StarLogo TNG and cognitive load questionnaire. First, students use the StarLogo TNG program to complete the assignments, for instance biology simulation [8], given by individual, face-to-face pair programming, and distributed pair programming. Then, students were requested to fill the cognitive load questionnaire. The final data analysis explored the differences for the three forms of programming activity. The study used the following two experimental research tools: StarLogo TNG and cognitive load questionnaire. The students

Authorized licensed use limited to: California State University - Chico. Downloaded on September 09,2021 at 05:45:11 UTC from IEEE Xplore. Restrictions apply.

used StarLogo TNG in three forms of pair programming. After learning activities, students were requested to fill the cognitive load questionnaire.

In the intrinsic, extraneous, germane three kinds of cognitive load, the extraneous and germane cognitive loads are related to instructional design. Pair programming document has been proved to be appropriate teaching activities. In other words, pair programming can enhance learners' learning performance although it will cause germane cognitive load. After students used StarLogo TNG for three forms of pair programming, participants filled out a brief questionnaire to assess the cognitive load they experienced while programming by StarLogo TNG. The cognitive load questionnaire consisted of six Likert 9-point scales questions (i.e. 1 represents strongly disagree, 2 represents disagree, 3 represents moderately disagree, 4 represents mildly disagree, 5 represents undecided, 6 represents mildly agree, 7 represents moderately agree, 8 represents agree, and 9 represents strongly agree). Likert 9-point scales have high granularity, which bring advantages of more precise data, higher reliability and validity, and fewer neutral and "uncertain" responses. The items in cognitive load questionnaire are shown as the following table.

TABLE I. Cognitive load questionnaire for three activities.

Dimension	Questionnaire item
Self-reported invested mental effort	How much mental effort did it cost you to solve this programming assignment?
	How much attention does it take up to solve this programming assignment?
Self-reported stress level	How frustrated were you while working on this programming assignment?
	How stressed were you while working on this programming assignment?
Self-reported difficulty of materials	How difficult was this programming assignment to solve?
	How many mental images do you have for solving this programming assignment?

Before conducting the three phases experiment, we introduced and gave tutorials about StarLogo TNG to make sure that students had learned basic skills of StarLogo TNG. There are 46 subjects, who are freshmen and major information technology, in the experiment. Students went through three different activities in a semester. In the first phase, students should think and solve programming assignment alone after the class. The first phase lasted for five weeks and every week had one assignment. In the second phase, the face-to-face pair programming activity was conducted every week. Students can freely seek for his/her partner, allow them discussing, and use one computer to design their programs. The second phase lasted for five weeks, too.

The profile of the pair programming sessions showed an overall pattern with most time spent on comprehension (understanding existing code and/or the nature of the problem), followed by writing new code and then testing and least time discussing the IDE and commenting code [10]. In the third phase, distributed pair programming activity was conducted. Students can freely seek for his/her partner, but the role, i.e. driver or observer, should be determined in advance. The third phase only lasted for four weeks. At the end of the semester, students were requested to complete this cognitive load questionnaire. The experiment flow is shown in the following Figure.

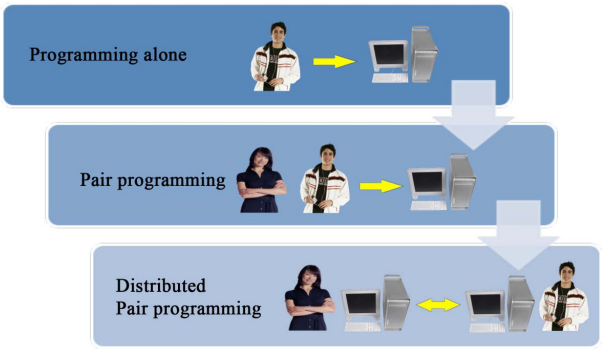

Fig. 3. Experiment design and flow.

III. Experiment Results

In individual programming activity, the average of cognitive load is 7.837 and the standard deviation is 1.272. However, in the face-to-face pair programming activity, i.e. traditional pair programming, the average of cognitive load is 6.473 and the standard deviation is 1.801. The average difference is 1.364 between individual and traditional pair programming. The t-value of paired t-test is 5.429 (degree of freedom 43) with p value less than 0.001. In other words, the statistical results show that students' cognitive load of individual programming is significant higher (7.837>6.473) than programming in the face-to-face collaborative activity. The result, shown in Table 2, is consistent with reports in the literature.

TABLE II. Cognitive load comparison of individual and traditional pair programming.

Group	N	Average	SD	T value
Individual	44	7.837	1.272	5.429***
Face-to-face PP	44	6.473	1.801	

There is significant difference between face-to-face and distributed pair programming, too. As aforementioned, the average of cognitive load is 6.473 and the standard deviation is 1.801 in traditional pair programming. Nevertheless, the cognitive load of distributed pair programming is much higher. In distributed pair programming, the average of cognitive load is 7.371 and the standard deviation is 1.506. The t-value of paired t-test is -4.319 (degree of freedom 43) with p value less than 0.001. The results of Table 3 show that students prefer traditional pair programming to distributed pair programming although distributed pair program can give students the power to complete assignment at their own pace and when it is convenient for them.

Authorized licensed use limited to: California State University - Chico. Downloaded on September 09,2021 at 05:45:11 UTC from IEEE Xplore. Restrictions apply.

TABLE III. Cognitive load comparison of traditional and distributed pair programming.

Group	N	Average	SD	T value
Face-to-face PP	44	6.473	1.801	-4.319***
Distributed PP	44	7.371	1.506	

The most interesting part is the comparison between individual and distributed pair programming although there is no significant difference between them. As aforementioned in Table 3 and Table 4, the average of cognitive load for individual and distributed pair programming is 7.837 and 7.371 respectively. Since the averages are so close, the t-test shows no significant difference with t-value 1.688. From our informal interview with students, they prefer distributed pair programming to solving problem individually. Although online conferencing brings some cognitive load about additional technology, students indicated that they prefer to actively participate real-time discussions rather than solve programming assignment alone.

TABLE IV. Cognitive load comparison of individual and distributed pair programming.

Group	N	Average	SD	T value
Individual	44	7.837	1.272	1.688
Distributed PP	44	7.371	1.506	

In summary, most students prefer traditional pair programming, followed by the distributed pair programming, and then individual learning activity according the results from our three phases experiment ($6.473 < 7.371 < 7.837$). While students prefer collaborative mode, distributed pair programming will raise additional cognitive loads. Thus, there is room for improvement on the tool for distributed pair programming. Individual programming assignment should be avoided because it has the highest cognitive load.

Although the studies report highly positively on pair programming, they also raise a number of further questions:

- How can the tool for distributed pair programming be improved?
- Do the driver and navigator have similar contributions as traditional face-to-face pair programming after improvement of distributed pair programming tool?
- Does VPL pair programming have similar learning outcome as pair programming on traditional text-based programming language in nature, and if not, can this be encouraged?

There is still much to learn about the nature of distributed VPL pair programming. Particularly, a computer science instructor need guidelines of distributed pair programming when teaching VPL in the classroom in order to bring the many benefits it has been shown to have.

IV. Conclusion

Teach students how to use information technology self-learning and problem solving is an important issue. Visual programming languages, such as StarLogo TNG, are widely used to promote computational thinking in primary and secondary schools. Pair programming was proved to be an effective teaching strategy in literature, but the effects on cognitive load in distributed situation is unclear. Therefore, we designed a three phase experiment to compare the cognitive loads of learning StarLogo TNG among individual, face-to-face pair programming, and distributed pair programming. After cross analysis of experiment data, we may claim that learners prefer pair programming to individual learning. Furthermore, the difference of cognitive loads between distributed pair programing and individual learning is insignificant. Distributed pair programming, by way of explanation, still is not as nature as traditional pair programming. We suggest that instructors should provide more scaffolding while conducting distributed pair programming.

Acknowledgment

This study is funded by the Ministry of Science and Technology (MOST) of Taiwan under contract numbers 104-2628-S-024-001-MY3, 103-2511-S-024-003, and 100-2628-S-024-001-MY3.

References

[1] Bishop-Clark, C., Courte, J., & Howard, E. V. (2006). Programming in pairs with Alice to improve confidence, enjoyment, and achievement. Journal of educational computing research, 34(2), 213-228.

[2] Bryant, S., Romero, P., & du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. International Journal of Human-Computer Studies, 66(7), 519-529.

[3] Gorla, N., & Lam, Y. W. (2004). Who should work with whom?: building effective software project teams. Communications of the ACM, 47(6), 79-82.

[4] Han, K.-W., Lee, E., & Lee, Y. (2010). The impact of a peer-learning agent based on pair programming in a programming course. Education, IEEE Transactions on, 53(2), 318-327.

[5] He, Y. Y., Chang, C. K., & Liu, B. J. (2010). Teaching computer programming for freshmen: A study on using Scratch as remedial teaching. International Journal on Digital Learning Technology, 2(1), 11-32.

[6] McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. Communications of the ACM, 49(8), 90-95.

[7] McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. Paper presented at the ACM SIGCSE Bulletin.

[8] Smith, V. A., & Duncan, I. (2011). Biology students building computer simulations using StarLogo TNG. Bioscience Education, (18).

[9] Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. Communications of the ACM, 43(5), 108-114.

[10] Bryant, S., Romero, P., & du Boulay, B. (2006). The collaborative nature of pair programming. In Extreme programming and agile processes in software engineering (pp. 53-64). Springer Berlin Heidelberg.

Authorized licensed use limited to: California State University - Chico. Downloaded on September 09,2021 at 05:45:11 UTC from IEEE Xplore. Restrictions apply.