Kashihara, A. et al. (Eds.) (2024). Proceedings of the 32nd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Exploring the Effect of Collaborative Programming Learning Environment on Student's Computer Programming Competencies and Cognitive Learning

Chia-Jung CHANG[a*] & Wen-Lung HUANG[b]
[a] *Department of Information Communication, Yuan Ze University, Taiwan*
[b] *Department of Communication, Fo Guang University, Taiwan*
*ccj@saturn.yzu.edu.tw

Abstract: The study aims to develop a synchronous collaborative programming learning environment to facilitate students' computer programming and to explore its impact on programming competencies, problem-solving behaviors, and cognitive learning. The results showed that the environment not only improved students' programming competencies, but also enhanced their high-level cognitive learning. Additionally, students in collaborative programming demonstrated a higher percentage of problem-solving behaviors in exploring and understanding, and planning and executing. Meanwhile, the study identified the differences between individual and collaborative activities on problem-solving behaviors.

Keywords: collaborative learning, computer programming, self-efficacy, problem-solving behaviors

1. Introduction

Researchers and educators have paid much attention for fostering students' computer programming competencies as the competencies are regarded as one of the core competencies of 21st century (Hsu & Hwang, 2017). However, computer programming is a complex learning process which involves both the implementations of coding and problem-solving. Students often encounter learning difficulties related to syntax, logical statements, and structures (Derus & Ali, 2012), which lead to high dropout rates and decreased motivation to learning computer programming. Hence, how to solve the difficulties students encountered is an imperative practical issue.

Many studies designed various teaching and learning instructions to promote students' computer programming competencies (Cheah, 2020). For example, Zhi et al. (2019) proposed worked example instructions to support students in understanding computer programming. In a study of Nainan and Balakrishnan (2019), they combined worked examples with explanation and debugging to help novices to aware and analyze programming problems. The results indicated that worked example didn't significantly improve students' programming skills, and increased their cognitive load (Zhi et al., 2019).

Collaborative learning is considered an effective approach for enhancing student's cognition and competencies through social interaction (Wu et al., 2019). Social interaction not only enhances the conceptual understanding, but also facilitates members monitoring and reflecting to develop better solutions (Dillenbourg & Traum, 2006). However, most programming learning environment are mainly individual-based, meaning that students cannot code a program together. Hence, some studies implemented collaborative learning in computer programming contexts. For instance, Boyer et al (2008) developed a distributed synchronous collaborative programming, indicating that such collaborative programming could benefit students' learning motivation and retention. In a study of Wu et al (2019), they analyzed

the discourse and epistemic network models of collaborative programming groups, showing that low-performing groups used guess-and-check approaches while high-performing groups applied systematic approach. Another similar study by Jonsson and Tholander (2022), they designed a co-coding with AI application which synchronized group members codes. They found that students exhibited inconsistent and erroneous behaviors and suggested that the activity should integrate reflection and exploration into collaborative programming contexts.

Many studies stated that programming learning is a developmental process which transformed mental representations of problems and solutions into executable codes (Loksa, Jernigan, Oleson, Mendez, & Burnett, 2016). Cheah (2020) highlighted the issue of the high failure in computer programming learning since students' lack of problem-solving abilities. Although these studies showed the positive effect of collaborative learning in computer programming contexts, its impact on students' cognition activities and learning behaviors remains unclear. Moreover, these studies lack the support of monitoring and reflection.

To support students in collaborative programming, the study developed a synchronous collaborative programming learning environment which provided students with a shared workspace for joint coding. The environment also provided cognitive and metacognitive scaffoldings to help students understand the concepts of programming and monitor their learning process. Specifically, research questions are presented as follows.

1. Does the synchronous collaborative programming learning environment facilitate students' computer programming competencies?
2. What problem-solving behaviors do students exhibit in the synchronous collaborative programming learning environment?
3. What impact does the synchronous collaborative programming learning environment have on students' cognitive learning?

2. Method

2.1 Participants

The study conducted a pilot study to explore the effect of the synchronous collaborative programming environment on students' programing behaviors and their perceptions of learning programming self-efficacy. The study involved 12 students from a private university in northern Taiwan. Since they had no experience in making games using computer programming, they were suitable as research samples.

2.2 Game making programming learning activity

The study developed a collaborative programming learning environment which provided students with a shared workspace of real-time code editor and compiler for code execution. Group members' codes also were immediately synchronized in a shared workspace. In order to support group discussion, the environment allowed group communication via an online text-based chatroom. Moreover, the environment tracked and displayed each member's cursor, and synchronously remarks on all members' screens to support group monitoring of their programming actions. Since students lacked the experience in programming games, the study integrated problem-solving processes into the environment, including exploring and understanding, representing and formulating, planning and executing, and monitoring and reflecting, to guide them through game programming.

Two game-making learning activities were designed within the environment: an individual activity and a collaborative activity. Each activity was divided into sub-tasks based on the four phases of problem-solving process. In the exploring and understanding phase (E&U), students experience and understand the goals of each sub-task by interacting with a manipulable game. In the representing and formulating phase (R&F), it provides basic concepts and syntax necessary for creating game functions. In the planning and executing phase (P&E), students used an online programming editor and compiler, and received procedural and structured scaffolding to support task completion. The monitoring and

reflecting phase (M&R), the metacognitive questions were designed to help students monitoring and reflecting learning process.

2.3 Procedure

Students first participated in the individual activity, followed by the collaboration activity. The individual and collaborative activities both lasted for three 50-minute sessions. Before the activities, researchers introduced the interface of the programming learning environment and game making learning task with the four problem-solving phases. After each activity, a computer programming self-efficacy scale (CPSES) was used to measure students' perception of programming learning. Moreover, students were required to answer a reflective question.

2.4 Data collection and analysis

The computer programming self-efficacy scale developed by Tsai, Wang, and Hsu (2018) was used to measure students' perception of programming learning in the individual and collaborative activities. The Cronbach's alpha of the scales was 0.97 and 0.94, indicating that the scale was sufficiently reliable. The responses to the scales were analyzed by Wilcoxon signed-rank test. Besides, students' learning behaviors in the environment were automatically recorded as logfiles. Their learning behaviors in the individual and collaborative activities were counted and compared using a chi-square test. Besides, students' responses to question were analyzed by two researchers based on revised Bloom's taxonomy (Krathwohl, 2002). The Cohen's kappa value of interrater reliability was 0.81, indicating that two researchers had a high-level agreement.

3. Result and Discussion

3.1 Learning computer programming self-efficacy

Table 1 presents the results of students' computer programming self-efficacy using Wilcoxon sign rank test. The results indicated that students perceived a significantly higher level of programming competencies on logical thinking ($z=-2.31$, $p<.05$), algorithm ($z=-1.92$, $p=.05$), and debug ($z=-2.06$, $p<.05$) in collaborative activity than did they in individual activity. There was no significant difference in control, showing that students perceived a similar level of control competency in the individual and collaborative activities. The results suggested that the collaborative activity improved their programming competencies in constructing logical statements, designing algorithm to solve problems, and finding and fixing error codes.

Table 1. *The result of students' computer programming self0efficacy in the two activities*

	Individual		Collaboration		Z-score
	Mean	SD	Mean	SD	
Logical thinking	3.67	1.13	4.48	1.41	-2.31*
Algorithm	2.92	1.29	3.53	1.13	-1.92*
Debug	3.83	1.21	4.44	1.52	-2.06*
Control	4.69	1.09	4.17	1.57	-0.53

*<.05

3.2 The problem-solving behaviors

Table 2 shows the results of the frequency and percentage of students' learning behaviors in the two activities. There was a significant difference between the two activities. Compared to individual activity, students in collaborative activity exhibited a higher percentage of behaviors

in ‘planning and executing’ and ‘exploring and understanding’. However, the percentage of behaviors in ‘representing and formulating’ and ‘monitoring and reflecting’ decreased.

Table 2. *The distribution of students’ problem-solving behaviors in the two activities*

Activity	E&U	R&F	P&E	M&R	Total	x^2
Individual	233 (4.33%)	1316 (24.44%)	3100 (57.57%)	736 (13.67%)	5385 (100%)	345.2.***
Collaboration	589 (6.84%)	1582 (18.36%)	5537 (64.26%)	909 (10.55%)	8617 (100%)	
Total	822	2898	8637	1645	14002	

*<.05, **<.01, ***<.001

3.3 Students’ responses to programming learning

Students’ response to learning programming based on Bloom cognitive domain were analyzed as shown in Figure 1. The results indicated that students in individual activity mainly exhibited a high level of cognitive learning in application (39.02%), remember (26.83%), and understand (19.51%), and low level of cognitive learning in analysis, evaluation and create. Regarding to collaborative activity, students exhibited high level of cognitive learning in application (39.13%), evaluation (23.91%), and remember (21.74%), and low level of cognitive learning in understand, analysis, and create.

Specifically, the main differences between the two activities were that students in collaborative activity demonstrated a higher level of cognitive learning in evaluation (23.91%) and create (8.70%) while students in individual activity demonstrated a higher percentage in understand (19.51%) and a lower percentage in evaluation (0%) and create (2.44%). The results imply that collaborative activity may be helpful for enhancing high level of cognitive learning in evaluation and create. Besides, the results suggested that students in the both activities may be difficulties in analysis learning.

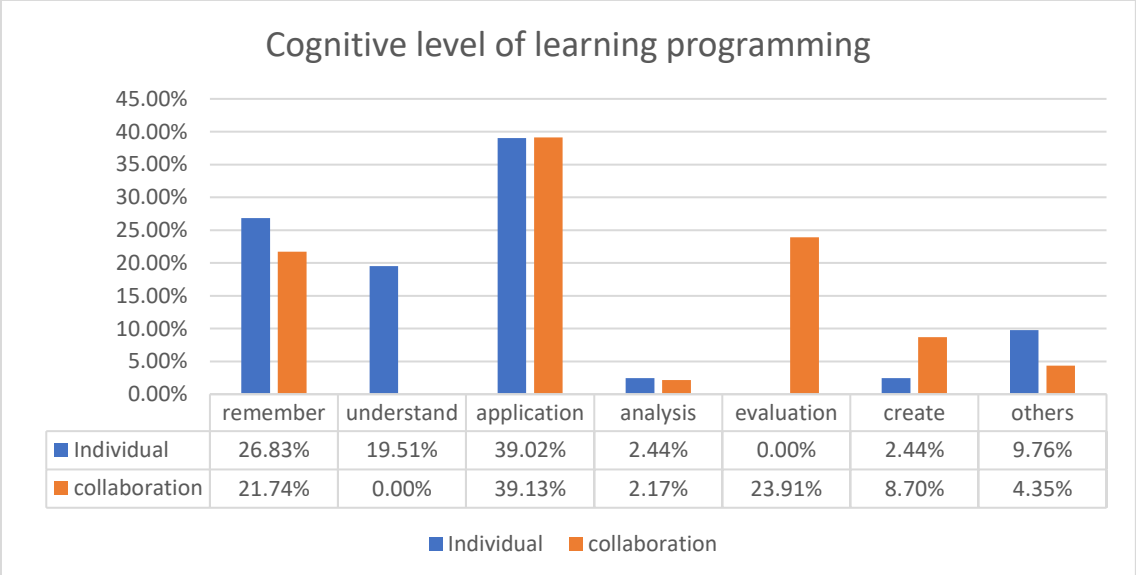

Figure 1. The distribution of cognitive level of learning programming in the individual and collaborative activities

4. Conclusion and Future

The findings indicated that the environment not only improved students’ computer programming competencies but also enhanced high-level cognitive activities. However, due to the limitation of small sample size and experimental design, the results cannot be over-

generalized. Future studies could employ large sample size and quasi-experiment to verify the results of this study.

Acknowledgements

The research is partially funded by National Science and Technology Council of Taiwan under NSTC 113-2410-H-155-009- and 113-2410-H-431-006-MY2.

References

Boyer, K. E., Dwight, A. A., Fondren, R. T., Vouk, M. A., & Lester, J. C. (2008, June). A development environment for distributed synchronous collaborative programming. *In Proceedings of the 13th annual conference on Innovation and technology in computer science education* (pp. 158-162).

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2), ep272.

Derus, S. R. M. & Ali, A. Z.M (2012). Difficulties in learning programming: Views of learners. *The proceeding of 1st International Conference on Current Issues in Education*, Yogyakarta, Indonesia

Dillenbourg, P., & Traum, D. (2006). Sharing solutions: Persistence and grounding in multimodal collaborative problem solving. *The Journal of the Learning Sciences, 15*(1), 121-151.

Hsu, T. C., & Hwang, G. J. (2017). Effects of a structured resource-based web issue-quest approach on students' learning performances in computer programming courses. *Journal of Educational Technology & Society, 20*(3), 82-94.

Jonsson, M., & Tholander, J. (2022, June). Cracking the code: Co-coding with AI in creative programming education. *In Proceedings of the 14th Conference on Creativity and Cognition* (pp. 5-14).

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into practice, 41*(4), 212-218.

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016, May). Programming, problem solving, and self-awareness: Effects of explicit guidance. *In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 1449-1461).

Nainan, M., & Balakrishnan, B. (2019). Design and Evaluation of Worked Examples for Teaching and Learning Introductory Programming at Tertiary Level. *Malaysian Online Journal of Educational Technology, 7*(4), 30-44.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Educational Computing Research, 56*(8), 1345-1360.

Wei, X., Lin, L., Meng, N., Tan, W., & Kong, S. C. (2021). The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. Computers & education, 160, 104023.

Wu, B., Hu, Y., Ruis, A. R., & Wang, M. (2019). Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning, 35*(3), 421-434.

Zhi, R., Price, T. W., Marwan, S., Milliken, A., Barnes, T., & Chi, M. (2019, February). Exploring the impact of worked examples in a novice programming environment. *In Proceedings of the 50th acm technical symposium on computer science education* (pp. 98-104).