Chapter 6
Improving Programming Skills Through an Innovative Collaborative Programming Model: A Case Study

Abstract In recent years, education has put considerable emphasis on the development of programming skills. However, students, especially, pupils often face challenges in programming. This study aims to improve pupils' programming skills through an innovative collaborative programming model. This model includes six phases, namely, understanding, designing, programming, sharing, evaluating, and refining. A case study was conducted to get a better understanding of participants' perceptions, programming skills, and collaborative problem-solving abilities. The results indicated that participants were interested in programming, and their programming skills as well as problem-solving abilities were improved. The implications for teachers and practitioners are also discussed in depth.

Keywords Collaborative programming · Collaborative problem solving · Programming skills

6.1 Introduction

The development of advanced technologies requires lots of human resources in programming skills (Lu et al., 2017). Computer science education has become more and more important in recent years (Chen et al., 2017). Thus, computer programming course has been a very fundamental course at all levels (Gordon & Brayshaw, 2008). Vaca-Ca´rdenas et al. (2015) proposed that programming skills are very crucial for preparing students for twenty-first-century success. Therefore, there is an urgent need to improve learners' programming skills.

Previous studies reported many strategies for improving computer programming skills, such as the use of robot (Noh & Lee, 2020), problem posing-based strategy (Wang & Hwang, 2017), developing a groupware system (Bravo et al., 2013), and flipped classroom model (Durak, 2020). However, the effectiveness of these strategies for pupils need to be investigated further. In addition, novice programmers like pupils encountered a number of problems and challenges. It was found that the first programming experience affected the interest and willingness of programming (Uysal, 2014). Therefore, it is necessary to develop a holistic model to help novice

L. Zheng, *Data-Driven Design for Computer-Supported Collaborative Learning*,
Lecture Notes in Educational Technology,
https://doi.org/10.1007/978-981-16-1718-8_6

programmers to improve their programming skills. This study sought to propose an innovative model to help pupils to improve collaborative programming skills in an Arduino course. The present study also validated the feasibility of the proposed model through a case study. Arduino provided an introduction to microcontrollers, and the graphical languages made it accessible to non-programmers (Reas & Fry, 2014). The following sections will illustrate literature review, method, results, and conclusions.

6.2 Literature Review

Collaborative programming has been considered as a very effective strategy for improving programming skills (Bailey & Mentz, 2017). Kanika et al. (2020) found that collaborative programming enabled students to learn from peers and write efficient programs. Teague and Roe (2008) believed that collaborative programming was very helpful for novice programmers in terms of establishing collective understanding of problems, receiving peers' feedback, and building knowledge. Many scholars investigated on how to improving programming skills through collaborative programming. For example, Lu et al. (2017) applied learning analytics to improve programming skills in a MOOCs collaborative programming course. Chorfi et al. (2020) adopted a computer-supported collaborative learning-based groupware to improve programming skills. Lu et al. (2020) proposed that a continuous inspection paradigm can serve as an effective method to ensure coding quality and improve programming skills. Wei et al. (2020) proposed a partial pair programming method, and they found that elementary school students' computational thinking skills and self-efficacy improved through the partial pair programming method.

In addition, it was found that computer programming is a problem-solving task (Piteira & Costa, 2013), and programming skills are closed related to problem-solving skills (Fessakis et al. 2013). Previous studies also indicated that problem-solving is a crucial aspect of programming (Deek et al., 1999; Shi et al., 2019). Beck and Chizhik (2013) proposed that collaborative programming helped learners to develop confidence in problem-solving abilities. Sun et al. (2020) analyzed three contrasting pairs' collaborative programming processes, and they demonstrated the complex relations among collaborative behaviors, discourses, and performances. However, most studies implemented collaborative programming in higher education context. Few studies conducted collaborative programming among elementary school students. To close this gap, this study aims to improve pupils' programming skills through the proposed model. The following sections will illustrate the proposed model in depth.

6.3 The Model of Collaborative Programming

This study proposed an innovative model of collaborative programming, including understanding, designing, programming, sharing, evaluating, and refining. This model aims to improve learners' collaborative problem-solving skills and programming skills, as shown in Fig. 6.1. It is a cycle, and there are six phases. The first phase is to understand the context, task requirements, and learning objectives of programming. The second phase is to design how to program and draw the flowchart to represent the thoughts. The third phase is to program collaboratively through online programming tools. In this phase, learners need to program and debug the code. The fourth phase is to share the group products with peers and teachers. The fifth phase is to evaluate the quality of group products by teachers and peers. The final phase is to refine and revise the program further based on teachers' and peers' suggestions.

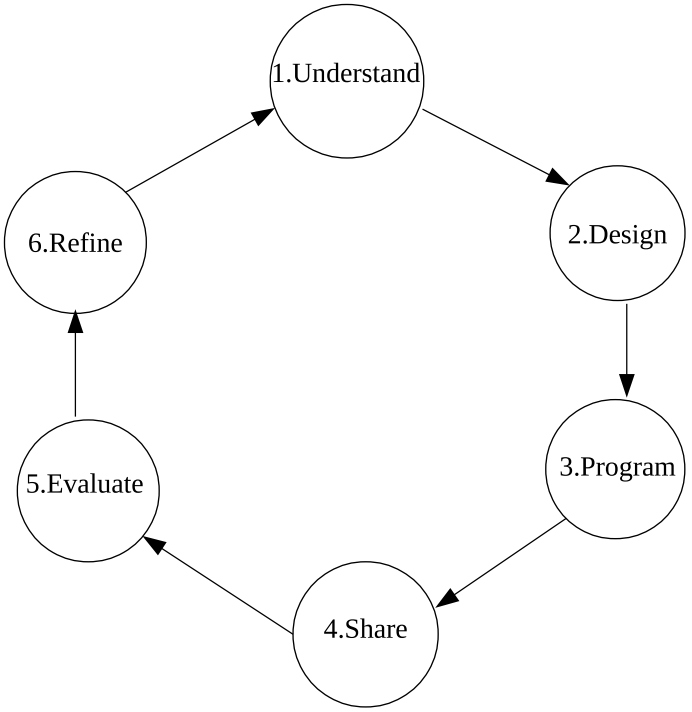

Fig. 6.1 The model of collaborative programming

6.4 Method

6.4.1 Participants

This study enrolled 9 pupils to voluntarily participate in this study. All the participants were divided into 4 groups of two or three students. Groups 1 and 3 were composed of girls. Groups 2 and 4 were composed of boys. The average age was 11 years. They were from the same elementary school. However, they had never collaboratively programmed before. They did not know how to program.

6.4.2 The Introduction to the Program

The tasks of the program were to make a fortune cat through a steering engine. The learning objectives of this program were to understand the principles and control method of a steering engine as well as acquire the applications of random number and key module. After participation in this study, students' interest in programming and programming skills was expected to be enhanced further. The learning materials include lecture notes, Arduino tools, the examples of programming, computers, scissors, gummed tape, colored paper, and colored pencil.

6.4.3 Procedures

This study followed the proposed collaborative programming model to design and implement the collaborative programming activities. Table 6.1 shows the procedures of the learning activity.

6.5 Results

6.5.1 Analysis of Programming Skills

Since each group collaboratively programmed and programming was the final group product, the rubric was designed to evaluate the programming skills. Table 6.2 shows the rubric of group products, and Table 6.3 shows the assessment results. It was found that group 4 achieved the highest score, followed by group 3, group 2, and group 1. Compared with previous programming skills, all the participants' programming skills were significantly improved (Fig. 6.2).

Table 6.1 The procedure of the first learning activity

Phases	Teachers' behaviors	Students' behaviors
1. Understand	1. The teacher demonstrated the pictures of various kinds of cats and asked a question "Do you know about the fortune cat?" 2. The teacher asked, "What are the major differences in these fortune cats?" 3. The teacher introduced the task and demonstrated the functionalities of made fortune cat. The principles and control methods of a steering engine as well as how to draw flowchart were also introduced by the teacher.	1. The students answered the question and linked it with their prior knowledge and experience. 2. The group members searched for information from the Internet and answered the question. 3. The students observed the made fortune cat, understood the task, and learned about the target knowledge.
2. Design	1. The teacher engaged students in designing the appearance of the fortune cat. 2. The teacher engaged students in designing the flowchart of the program.	1. The group members conducted collaborative learning to draw the fortune cat together. 2. The group members conducted collaborative learning to draw the flowchart of the program together.
3. Program	The teacher introduced how to program through the MIXLY (an open-source software) and the hardware.	1. The group members collaboratively program to make a fortune cat that can swing the arms. 2. Once they finished, they connected the hardware to test.
4. Share	The teacher organized the students to share their group products and discuss among all participants.	Each group shared their group products and discussed with peers.
5. Evaluate	The teacher evaluated the group product of each group.	Each group reflected and thought on how to improve the group products.
6. Refine	The teacher summarized and provided their suggestions to refine the group products.	Each group refined the group products further and demonstrated the revised group products.

Table 6.2 The rubric of group products

Dimensions	Explanations (10)	Explanations (15)	Explanations (20)
Originality (20)	The group product is not original and just followed the teachers' model.	The group product is a little bit innovative.	The group product is very original and innovative.
Programming (20)	The flowchart is incomplete, and there are some errors in programming.	The flowchart is complete and there is no error in programming.	The flowchart is perfect and the programming works well.
Hands-on skills (20)	The hands-on skills are low and the connection of hardware is loose and in chaos.	The hands-on skills are medium, and the connection of hardware is in order.	The hands-on skills are high, and the connection of hardware is firm and perfect.
Collaboration (20)	There is no communication and collaboration.	There is little bit communication and collaboration.	The group members communicated and collaborated closely.
Functionality (20)	The group product did not achieve the expected functionalities.	The group product achieved the expected functionalities.	The group product achieved the expected functionalities, and several new functionalities were added.

Table 6.3 The results of group products

Groups	Originality	Programming skills	Hands-on skills	Collaboration	Functionality	Total
Group 1	18	10	14	12	15	69
Group 2	15	18	16	13	15	77
Group 3	17	16	16	14	15	78
Group 4	17	18	18	17	15	85

6.5.2 Analysis of Collaborative Problem Solving

This study adopted the collaborative problem-solving framework developed by PISA (2017) to evaluate collaborative problem-solving competency. This assessment

framework is a matrix which is composed of vertical components and horizontal components. The vertical components were coded as (A) explore and understand (20 scores), (B) represent and formulate (25 scores), (C) plan and execute (30 scores), and (D) monitor and reflect (25 scores) (PISA, 2017; Song, 2018). The horizontal components were coded as (1) establish and maintain shared understanding (45 scores), (2) take appropriate action to solve the problem (25 scores), and (3) establish and maintain team organization (30 scores) (PISA, 2017; Song, 2018). The result of the collaborative problem-solving competency was the matrix of ABCD1, ABCD2, and ABCD3. Table 6.4 shows the results of collaborative problem solving for four groups. It was found that group 4 achieved the highest score in collaborative problem-solving competency, followed by group 3, group 2, and group 1.

Fig. 6.2 The programming of group 4

Table 6.4 The results of collaborative problem solving

Groups	Matrix1	Matrix2	Matrix3	Total
Group 1	31	12	20	63
Group 2	33	18	22	73
Group 3	35	20	25	80
Group 4	39	23	27	89

6.5.3 Interview Results

To get a better understanding of the participants' perceptions, all the participants were interviewed by researchers. The interview results indicated that participants were more interested in programming, and their programming skills as well as collaborative problem-solving skills were improved further.

First, the participants of the four groups addressed that this activity was very interesting, and their interests in programming increased. For example, one student said that "Before I believed that programming is very difficult. But now I believe that programming is very interesting and not difficult because I can program through the graph programming tool. The text programming tool is very boring." Another student also addressed that "I like this activity very much. I benefit a lot from it. I have a strong sense of fulfilment when I finish program. I feel very exciting when the fortune cat can swing the arms."

Second, the participants of four groups believed that the proposed model can improve their programming skills. For example, one student stated that "Before, I just programming directly without design. I never revise the program before. But now I learn how to programming in a proper way. I understand the task and requirements of program, then I begin to design the flow chart. And then our group program and share with peers." Another student also revealed that "Understanding, design, programming, sharing, evaluation, and refinement are very scientific and useful for improving programming skills. I learn a lot from this model."

Third, the participants of the four groups believed that the proposed model improved their collaborative problem-solving skills. As one student said "Initially, there are some grammar errors in programming. Later our group members collaboratively corrected the errors and tested again. Finally, the fortune cat's arms swing." Another student revealed that "I believe the evaluating and refining group products is very important for improving problem solving skills. I learn a lot from refinement and solve several problems."

6.5.4 Implications

This study had several implications for teachers and practitioners. First, the proposed collaborative programming model was very effective and useful for improving programming skills. This model includes six phases, namely, understanding, designing, programming, sharing, evaluating, and refining. These six phases were an iterative cycle with the aim of improving programming skills. Among these six phases, programming and refining programming are very important. In addition, debugging is a fundamental skill of programming (Beller et al., 2018), and novice programmers took significantly more time in debugging (Chiu & Huang, 2015). Therefore, teachers and practitioners should allocate enough time to debug for programmers.

Second, novice programmers need help from teachers or experts. Therefore, it is suggested that teachers guide novice programmers to follow the model to improve programming skills step by step. In addition, novice programmers may encounter various kinds of problems. Teachers should provide real-time feedback for novice programmers, including emotional and cognitive feedback. For example, Fwa (2018) developed an affective tutoring system to help novice programmers to regulate their negative affect.

Third, learning activities about programming need to be elaborately designed before implementation since programming is considered to be a creative activity (Grover & Pea, 2013). The programming tasks, requirements, questions, interactive strategies, programming environments, learning materials, and assessment methods need to be designed carefully. It was found that visual presentation (diagrams, video, animation) and verbal explanation contributed to learning programming (Zhang et al., 2014). The drag and drop type applications can help younger students to learn computer science and informatics concepts (Kalelioğlu, 2015). Therefore, appropriate and smart programming environments are very crucial for improving programming skills.

6.6 Conclusions

This study investigated how to improve programming skills through an innovative collaborative programming model. This model included six steps, namely, understanding, designing, programming, sharing, evaluating, and refining. A case study was conducted to examine the feasibility and effectiveness of this model. The results indicated that the proposed model was very helpful and insightful for improving programming skills. The best group achieved the highest scores in terms of group product's quality and collaborative problem-solving skills. The interview results revealed that all the participants were very interested in programming, and their programming skills as well as collaborative problem-solving skills improved further.

However, this study was constrained by several limitations. First, only four groups participated in this study. Therefore, cautions should be exercised when generalizing the results. Future study will expand the sample size to conduct the empirical study. Second, this study only examined the group product quality and collaborative problem-solving abilities. Future study will examine the effectiveness of the proposed model from other perspectives.

References

Bailey, R., & Mentz, E. (2017). The value of pair programming in the IT classroom. *The Independent Journal of Teaching and Learning, 12*(1), 90–103.

Beck, L., & Chizhik, A. (2013). Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. *ACM Transactions on Computing Education (TOCE), 13*(3), 1–21.

Beller, M., Spruit, N., Spinellis, D., & Zaidman, A. (2018). On the dichotomy of debugging behavior among programmers. In *Proceedings of the 40th International Conference on Software Engineering* (pp. 572–583). https://www.spinellis.gr/pubs/conf/2018-ICSE-debugging-analysis/html/BSSZ18.pdf.

Bravo, C., Duque, R., & Gallardo, J. (2013). A groupware system to support collaborative programming: Design and experiences. *Journal of Systems and Software, 86*(7), 1759–1771.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education, 109*, 162–175.

Chorfi, A., Hedjazi, D., Aouag, S., & Boubiche, D. (2020). Problem-based collaborative learning groupware to improve computer programming skills. *Behaviour & Information Technology*, 1–20. https://doi.org/10.1080/0144929X.2020.1795263.

Chiu, C. F., & Huang, H. Y. (2015). Guided debugging practices of game based programming for novice programmers. *International Journal of Information and Education Technology, 5*(5), 343–347.

Deek, F. P., Turoff, M., & McHugh, J. A. (1999). A common model for problem solving and program development. *IEEE Transactions on Education, 42*(4), 331–336.

Durak, H. Y. (2020). Modeling different variables in learning basic concepts of programming in flipped classrooms. *Journal of Educational Computing Research, 58*(1), 160–199.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Fwa, H. L. (2018). An architectural design and evaluation of an affective tutoring system for novice programmers. *International Journal of Educational Technology in Higher Education, 15*(1), 38. https://doi.org/10.1186/s41239-018-0121-2.

Gordon, N. A., & Brayshaw, M. (2008). Inquiry-based learning in computer science teaching in higher education. *Innovations in Teaching and Learning in Information and Computer Sciences, 7*(1), 22–33.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior, 52*, 200–210.

Kanika, Chakraverty, S., & Chakraborty, P. (2020). Tools and techniques for teaching computer programming: A review. *Journal of Educational Technology Systems*, *49* (2), 170–198. https://doi.org/10.1177/0047239520926971.

Lu, O. H., Huang, J. C., Huang, A. Y., & Yang, S. J. (2017). Applying learning analytics for improving students engagement and learning outcomes in an MOOCs enabled collaborative programming course. *Interactive Learning Environments, 25*(2), 220–234.

Lu, Y., Mao, X., Wang, T., Yin, G., & Li, Z. (2020). Improving students' programming quality with the continuous inspection process: a social coding perspective. *Frontiers of Computer Science, 14*(5), 1–18.

Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development, 68*(1), 463–484.

PISA. (2017). PISA 2015 collaborative problem-solving framework. Retrieved from https://www.goo.gl/Yp6j8L.

Piteira, M., & Costa, C. (2013). Learning computer programming: study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication* (pp. 75–80). ACM.

Reas, C., & Fry, B. (2014). *Processing: A programming handbook for visual designers and artists* (2nd ed.). Cambridge, MA, USA: MIT Press.

Shi, J., Shah, A., Hedman, G., & O'Rourke, E. (2019). Pyrus: Designing a collaborative programming game to promote problem solving behaviors. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). https://library.usc.edu.ph/ACM/CHI2019/1proc/paper656.pdf.

Song, Y. (2018). Improving primary students' collaborative problem solving competency in project-based science learning with productive failure instructional design in a seamless learning environment. *Educational Technology Research and Development, 66*(4), 979–1008.

Sun, D., Ouyang, F., Li, Y., & Chen, H. (2020). Three contrasting pairs' collaborative programming processes in China's secondary education. *Journal of Educational Computing Research.* https://doi.org/10.1177/0735633120973430.

Teague, D., & Roe, P. (2008). *Collaborative learning: Towards a solution for novice programmers.* Paper presented at the tenth conference on Australasian computing education. Retrieved from https://eprints.qut.edu.au/17818/1/c17818.pdf.

Uysal, M. P. (2014). Improving first computer programming experiences: The case of adapting a web-supported and well-structured problem-solving method to a traditional course. *Contemporary Educational Technology, 5*(3), 198–217.

Vaca-Ca´rdenas, L. A., Bertacchini, F., Tavernise, A., Gabriele, L., Valenti, A., Olmedo, D.E, . . . E. Bilotta (2015). Coding with Scratch: The design of an educational setting for Elementary pre-service teachers. In *2015 International Conference on Interactive Collaborative Learning (ICL)* (pp. 1171–1177). IEEE. Florence, Italy.

Wang, X. M., & Hwang, G. J. (2017). A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode. *Educational Technology Research and Development, 65*(6), 1655–1671.

Wei, X., Lin, L., Meng, N., Tan, W., & Kong, S. C. (2020). The effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-efficacy. *Computers & Education, 160*, 104023.

Zhang, J. X., Liu, L., Ordóñez de Pablos, P., & She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using alice. *International Journal of Engineering Education, 30*(3), 560–565.