Education Tech Research Dev (2017) 65:1655–1671
https://doi.org/10.1007/s11423-017-9551-0

DEVELOPMENT ARTICLE

A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode

Xiao-Ming Wang[1] · Gwo-Jen Hwang[2]

Published online: 2 November 2017

Abstract Computer programming is a subject that requires problem-solving strategies and involves a great number of programming logic activities which pose challenges for learners. Therefore, providing learning support and guidance is important. Collaborative learning is widely believed to be an effective teaching approach; it can enhance learners' social interaction and offer a learning environment which provides rich learning experiences. However, the social interaction in collaborative learning does not occur automatically. Without proper guidance strategies or supporting tools for collaborative learning, the learning effects can be disappointing. To solve such a problem, a problem posing-based practicing strategy was proposed to support the development of a collaborative learning activity in a computer programming practice course. The students were guided to raise computer programming problems to boost the discussion among team members. The problems raised in each team were then exchanged and solved by another team to examine the coding and to provide feedback. To investigate the effectiveness of the proposed approach, an experiment was conducted in a C# programming course. Two classes of students from a university participated in the experiment. One class with 25 students was randomly assigned as the experimental group, and learned with a collaborative learning activity using the problem posing-based practicing strategy; the other class with 28 students was the control group, which learned with a conventional collaborative learning activity. The results show that the proposed strategy benefited the students in terms of improving their learning achievement, in particular, their programming skills. Moreover, it was found that the students who learned with the proposed approach had higher self-

✉ Gwo-Jen Hwang
gjhwang.academic@gmail.com

Xiao-Ming Wang
zsdwxm@126.com

[1] School of Teacher Education, Zhejiang Normal University, Jinhua 321004, China

[2] Graduate Institute of Digital Learning and Education, National Taiwan University of Science and Technology, Taipei 106, Taiwan, ROC

efficacy and lower cognitive load than those who learned with the conventional collaborative learning approach.

Keywords Computer programming · Problem posing · Cognitive load · Team-based learning · Self-efficacy of group learning

Introduction

Learning computer programming in teams is highly related to learning how to think creatively, infer systematically, and communicate and collaborate effectively with peers (Kalelioglu and Gülbahar 2014). Fostering computer programming skills is of increasing importance and is gradually becoming one of the core objectives of undergraduate and graduate programs in various domains, such as the software development and electronic engineering fields. However, for students, learning to program is a difficult process, especially for those novices without sufficient computer knowledge (Bravo et al. 2005). Therefore, how to provide a learning strategy or cognitive tool to help students learn computer programming is a challenging issue (Robins et al. 2003). In the past decades, researchers have conducted many studies to boost novices' understanding of computer programming and increase their programming skills (Robins et al. 2003; Wang et al. 2011). Mow (2008) pointed out that learning to program is a task with a high cognitive load, and to have great programming skills, students need to practice repeatedly. Several researchers have further indicated that students might lose enthusiasm and interest while programming, especially as a result of experiencing failure and frustration (Law et al. 2010; Isong 2014; Zhang and Yan 2014).

Learning with teamwork is deemed by educators to be a prospective approach; it can help students get through the difficult learning tasks with the help of social interaction (Kao et al. 2008). Several studies have shown the benefits of team-based computer programming (Sabin and Sabin 1994; Preston 2005). On the other hand, educators have also found that, on many occasions, learning in teams does not bring the expected social interactions; moreover, students' learning achievement might be disappointing due to the lack of learning strategies (Brush 1998; Serrano-Cámara et al. 2014). Therefore, how to increase students' social interactions in team-based learning environments has become an important issue.

Scholars have indicated that both cognitive and affective issues need to be taken into account when engaging students in complex tasks such as developing computer programs (Jonassen 2000). Regarding the cognitive issue, more complex tasks mean more cognitive operations, and hence more working memory is required (Antonenko and Niederhauser 2010). This implies that students' cognitive load could be a factor which affects their programming performance. Mayer's (1998) effort-based learning principle shows that students' self-efficacy could significantly affect their learning behaviors when dealing with complex tasks; that is, they might try harder and think more deeply if they have higher self-efficacy. Therefore, when engaging students in computer programming tasks, it is important to measure their self-efficacy and cognitive load as well as their programming outcomes.

Consequently, in this study, a problem posing-based practicing strategy was proposed to boost social interactions among learners in order to promote their team-based learning performance and self-efficacy as well as reduce their cognitive load when learning computer programming. Moreover, an experiment was conducted in a C# programming course to investigate the effectiveness of the proposed approach. During the learning activity, the team members took the programming tasks given by the teacher as a reference and started to pose problems collaboratively to design a new programming task. They then collaborated to complete the task. The teacher examined the results and gave feedback. Following that, the tasks were exchanged among teams; that is, each team tried to solve the tasks proposed by other teams. They then started a code review activity (i.e., examining the programs developed by others), and the teacher examined their tasks as well as the review results.

Literature review

Computer science education

Computer science education is considered as a fundamental curriculum at all levels of schooling around the globe (Gordon and Brayshaw 2008). In computer science courses, students are required to learn not only the basic knowledge of computers, but also the skills of using computers to deal with practical problems, such as the operation of computer application software and the development of computer programs (Esteves et al. 2011; Soloway 1993).

Scholars have indicated that computer programming is a problem-solving task that usually involves some semantically rich areas, meaning that it requires a great amount of domain knowledge and expertise to complete the task (Brooks 1983; Piteira and Costa 2013). In particular, the algorithm aspect of computer programming is highly related to learners' problem-solving skills (Papert 1980; Fessakis et al. 2013). For example, DiSessa and Abelson (1986) indicated that in the programming activity, students need to analyze, organize, convey, and evaluate their thoughts. Therefore, educators have emphasized the need to promote students' higher order thinking awareness in programming courses rather than just teaching them to memorize the programming code or instructions (Hawi 2010; Blignaut and Naude 2008).

In the past decades, researchers have helped learners gain computer programming skills via different paths, including building the theories of cognitive processing and understanding for computer programming (Brooks 1983), analyzing how programming novices learn from a psychological angle (Winslow 1996), evaluating the affective factors in learners' programming learning (Hawi 2010), and providing strategies, tools, and environments for learning (Hadjerrouit 2008; Esteves et al. 2011; Wang et al. 2011; Shaw 2012). Rogalski and Samurçay (1990) pointed out that gaining and enhancing knowledge in programming is a highly complex process, which involves cognitive activities and mental representations; such cognitive activities and mental representations have relations with programming design, understanding, adjustment, and testing. Mow (2008) indicated that programming is a task with a heavy cognitive load which requires students to practice many times in order to master it. Moons and de Backer (2013) also reported the importance of taking cognitive load into account when designing supporting mechanisms for computer programming. Researchers have further expressed that students might lose enthusiasm and interest in the process of learning to program, especially when experiencing failure (Law et al. 2010). Uysal (2014) emphasized that students' first programming experience could affect their interest in and willingness to take programming courses. Therefore, when designing activities for learning to program, it is important to provide learning support for students to increase their learning participation and reduce their cognitive load (Yang et al. 2015).

Collaborative learning

Collaborative learning means two or more people learn, or attempt to learn, something together, including knowledge, skills, abilities, and so on. It is usually seen as a process whereby people would be helped to finish a certain target or a product with designated goals (Serrano-Cámara et al. 2014). The term "collaborative learning" describes a scenario in which people hope that certain interactions would occur, and this would trigger the mechanism of group learning. In the past 10 years, researchers have reported the results of collaborative learning applied in different courses, such as writing (Wichmann and Rummel 2013), English (Rick et al. 2002; Law et al. 2015), natural science (Sung and Hwang 2013), and computer programming (Serrano-Cámara et al. 2014; Bravo et al. 2013; Preston 2005). Research has confirmed that collaborative learning encourages learners to use higher level cognitive strategies, critical thinking, in-depth learning and understanding, and positive attitudes towards learning and peers; such a learning approach provides them with a more open and flexible way to collaborate with their peers (Wang and Lin 2007; Laal and Ghodsi 2012; Laal et al. 2013). Meanwhile, researchers also believe that collaborative learning is an effective method to help learners solve difficult learning tasks with the support of peer interaction (Wang and Lin 2011).

However, the benefits of collaborative learning only appear in positive and well-functioning teams. Merely dividing students into learning groups cannot guarantee an effective collaborative learning activity (Soller 2001). Therefore, how to provide learning strategies, tools, and an environment to boost learners' social interaction is the key to their success.

Problem-posing and code review

Problem-posing refers to the cognitive activity in which a new problem is posed with a given question or certain situation; it also includes the process of problem solving to reform the problem (Lavy and Bershadsky 2003). The problem-posing strategy is a learning strategy which engages students in posing problems based on the issue specified by the teacher during the learning process; such a strategy has been widely applied in mathematics courses (Silver 2013; Shuk-kwan 2013; Singer and Voica 2013; Unal and Arikan 2015). In recent years, researchers have also implemented this strategy in other courses (Casagrande et al. 1998; Hung et al. 2014). For instance, they have adopted the problem-posing strategy to guide students to learn about local culture in a collaborative learning environment, with the results indicating that the strategy can effectively increase both the learning effects and group learning self-efficacy (Sung et al. 2016).

Many studies have shown that the problem-posing strategy can promote learners' curiosity regarding a certain concept, which is beneficial for knowledge construction and critical thinking ability improvement (Moses et al. 1990). Researchers have also found that when the teacher adopts the problem-posing strategy in school settings, the students usually more actively engage in creating and solving their problems, and hence have better learning performance (Barlow and Cates, 2006).

Furthermore, code review is an activity which engages students in examining programming code. It was originally proposed by Fagan (1976) in the field of software engineering to examine the quality of computer software (Mantyla and Lassenius 2009; Liu et al. 2001). The successful application of code review in software engineering has encouraged educators to use it as a learning strategy in computer science education (Wang et al. 2012). For instance, Wang et al. (2012) indicated that engaging students in peer code review could significantly improve their learning outcomes, including computer programming skills, collaborative learning, code norms, time management, and so on.

In this study, a problem posing-based practicing strategy was proposed to help learners experience collaborative learning in the practice of programming. The strategy is proposed based on the theories of constructivism and collaboration (Merrill 1991; Yakimovicz and Murphy 1995), and aims to engage students in higher order thinking activities, as indicated by Bloom (1994). During the learning process, the problem-posing strategy with follow-up code review tasks was used to promote the learners' involvement and interactions in the collaborative learning tasks. The theoretical basis of the approach can be further explained by the social constructivist theory, which emphasizes the role of social interactions in the learning process (Vygotsky 1962). Vygotsky indicated that knowledge is co-constructed; that is, students learn from interacting with peers. Therefore, he highlighted the importance of engaging learners in a process in which they can learn from each other and receive comments or assistance from peers. Au (1998) pointed out the effectiveness of applying the theory to school settings; for example, teachers can design learning activities to encourage those students who know the learning tasks and content better to work with those who do not.

In order to investigate the effectiveness of the proposed approach, an experiment was conducted in a C# programming course to answer the following research questions:

1. Compared to conventional collaborative learning: Can collaborative learning with the problem posing-based practicing strategy significantly improve learners' learning achievement?
2. Compared to conventional collaborative learning: Can collaborative learning with the problem posing-based practicing strategy significantly improve learners' self-efficacy of group learning?
3. Compared to conventional collaborative learning: Can collaborative learning with the problem posing-based practicing strategy significantly lower learners' cognitive load?

Problem posing-based practicing strategy for training computer programming skills

The online team-based learning environment with the problem posing-based practicing strategy was developed by integrating the Wiki function module in the Moodle learning management system. The learning environment allowed each team of students to edit and provide comments; in addition, it recorded each version of the edited results to enable the students to compare and see the changes between different versions.

In this study, the teacher used the grouping function to divide the students into learning teams. Besides, the teacher assigned two teams to work as a pair to exchange the programming tasks and conduct the coding review. Each team had two Wiki pages, one for problem-posing and the other for code review. Take Team i and Team j as an example; the problem-posing page for Team j was only visited by the members of Team j, but the code review page for Team j could be visited by the members of both Teams i and j. Each team needed to complete two programming tasks on these two pages: one was designed by themselves, and the other was designed by the other team. For instance, the members of

Team i not only posed a new programming task and provided the answer on the problem-posing page, but were also asked to complete the task proposed by Team j.

Figure 1 presents the application of the problem posing-based practicing strategy for Team i and Team j. When the students logged into the system and found their own group, they started posing problems on the problem-posing page.

There are five steps in the problem-posing activity. In the first step, the team members reviewed the learning contents based on the examples and problem-posing requirements provided by the teacher. The examples and learning contents given by the teacher for Team i and Team j were different. For instance, in Fig. 1, the content that Team i needed to pose problems for was an abstract class in C#, while for Team j, the students needed to pose problems about the interface of C#.

In the second step, after fully understanding the knowledge and the task requirements, the team members linked what they had learned with their life experience and voiced their opinions to settle the topic of the programming task. They then discussed the details of the task. For instance, the problem that Team i needed to pose was about an abstract class. They chose to calculate the area of a shape as the topic and requested to calculate the area of a circle and a sphere in their derived class. After posing the most preliminary problem, the team members entered the content of the task into the problem-posing page, as presented in Fig. 2. In should be noted that the learning content and system interface were in Chinese.

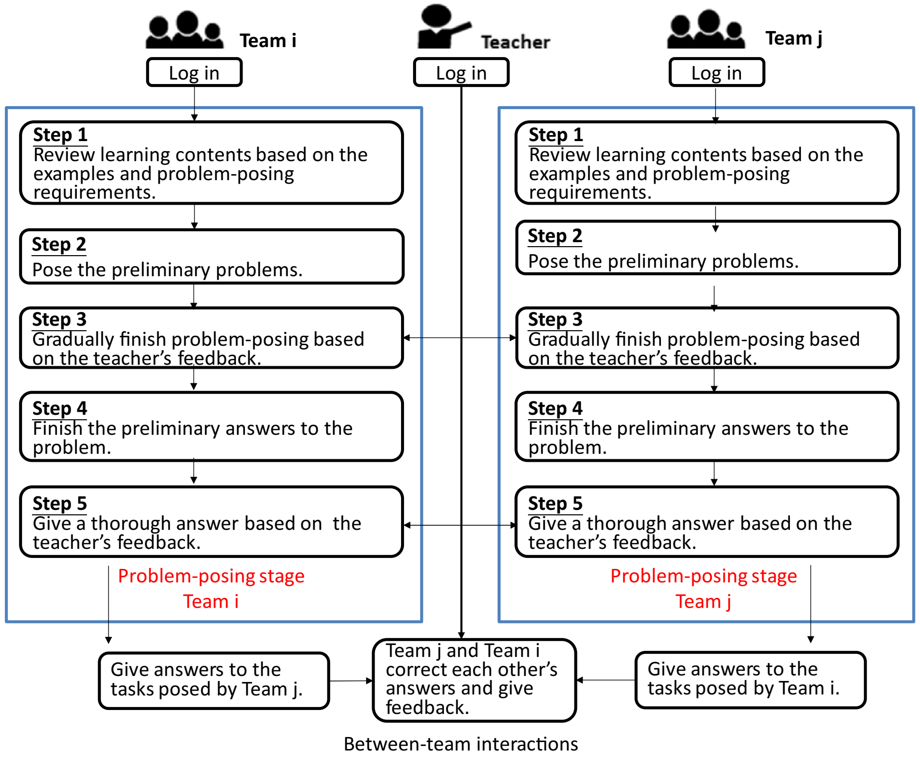
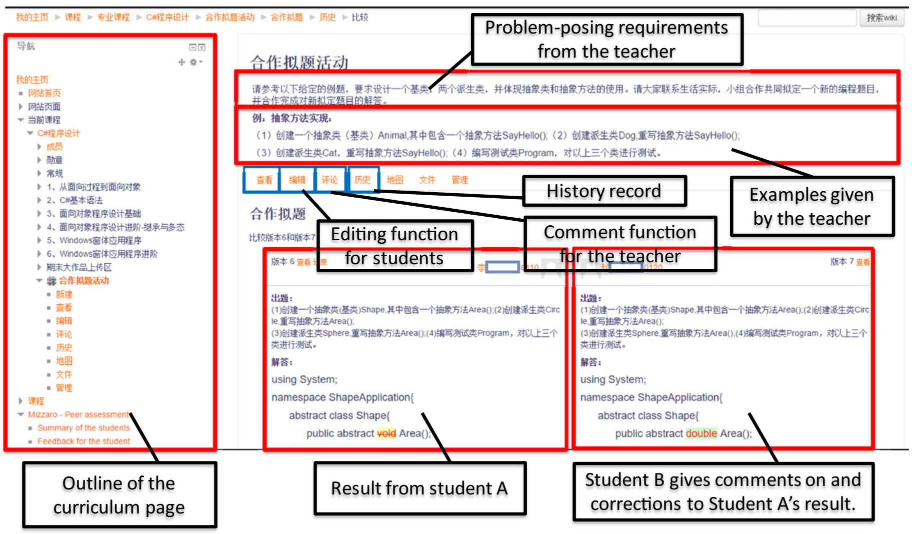

Team i
Teacher
Team j
Log in
Log in
Log in
Step 1
Review learning contents based on the examples and problem-posing requirements.
Step 2
Pose the preliminary problems.
Step 3
Gradually finish problem-posing based on the teacher's feedback.
Step 4
Finish the preliminary answers to the problem.
Step 5
Give a thorough answer based on the teacher's feedback.
Problem-posing stage
Team i
Step 1
Review learning contents based on the examples and problem-posing requirements.
Step 2
Pose the preliminary problems.
Step 3
Gradually finish problem-posing based on the teacher's feedback.
Step 4
Finish the preliminary answers to the problem.
Step 5
Give a thorough answer based on the teacher's feedback.
Problem-posing stage
Team j
Give answers to the tasks posed by Team j.
Team j and Team i correct each other's answers and give feedback.
Give answers to the tasks posed by Team i.
Between-team interactions

Fig. 1 Problem posing-based practicing strategy mechanism for Team i and Team j


我的主页 ▸ 课程 ▸ 专业课程 ▸ C#程序设计 ▸ 合作拟题活动 ▸ 合作拟题 ▸ 历史 ▸ 比较
搜索wiki
导航
我的主页
网站首页
网站页面
当前课程
C#程序设计
成员
勋章
常规
1、从面向过程到面向对象
2、C#基本语法
3、面向对象程序设计基础
4、面向对象程序设计进阶-继承与多态
5、Windows窗体应用程序
6、Windows窗体应用程序进阶
期末大作品上传区
合作拟题活动
新建
查看
编辑
评论
历史
地图
文件
管理
课程
Mizzaro - Peer assessment
Summary of the students
Feedback for the student
合作拟题活动
查看 编辑 评论 历史 地图 文件 管理
合作拟题
比较版本6和版本7
版本 6 查看 还原
版本 7 查看
Problem-posing requirements from the teacher
Examples given by the teacher
History record
Editing function for students
Comment function for the teacher
Outline of the curriculum page
Result from student A
Student B gives comments on and corrections to Student A's result.

请参考以下给定的例题，要求设计一个基类，两个派生类，并体现抽象类和抽象方法的使用。请大家联系生活实际，小组合作共同拟定一个新的编程题目，并合作完成对新拟定题目的解答。

例：抽象方法实现：

（1）创建一个抽象类（基类）Animal,其中包含一个抽象方法SayHello();（2）创建派生类Dog,重写抽象方法SayHello();
（3）创建派生类Cat，重写抽象方法SayHello();（4）编写测试类Program，对以上三个类进行测试。

出题：
(1)创建一个抽象类(基类)Shape,其中包含一个抽象方法Area();(2)创建派生类Circle,重写抽象方法Area();
(3)创建派生类Sphere,重写抽象方法Area();(4)编写测试类Program，对以上三个类进行测试。

解答：

```
using System;
namespace ShapeApplication{
    abstract class Shape{
        public abstract void Area();
```

出题：
(1)创建一个抽象类(基类)Shape,其中包含一个抽象方法Area();(2)创建派生类Circle,重写抽象方法Area();
(3)创建派生类Sphere,重写抽象方法Area();(4)编写测试类Program，对以上三个类进行测试。

解答：

```
using System;
namespace ShapeApplication{
    abstract class Shape{
        public abstract double Area();
```

Fig. 2 Problem-posing page

In the third step, after the team members had finished the preliminary problem-posing, using the comment function in Wiki, the teacher gave feedback for correction, checking the discreteness and difficulty level of the task, and if the posed problem met the requirements. The students subsequently worked together to improve the task based on the teacher's comments, using the Wiki collaboration function.

In the fourth step, after posing the programming task, the team members needed to complete the task. They could work on the programming alone and then collaborate to reach the final answer. It could also be conducted by having one member enter the coding and letting others do the corrections. No matter which way they decided to collaborate, they all needed to discuss the problem and reach a consensus. The history function in Wiki recorded the editing process for each member. As shown in Fig. 2, Student B could adjust the contents written by Student A and the system would record the adjustment for the users to compare the two versions.

In the fifth step, the teacher gave feedback on the programming, such as the correctness of the solution process and the code norms. The team members then adjusted the programming based on the suggestions given by other groups. Meanwhile, the teacher went around the class to observe the situation. Once each group had finished the problem-posing stage, it went on to the code review stage.

In the code review activity, two matched groups exchanged their programming tasks. For instance, Team i would put their task about abstract class on Team j's code review Wiki page, while Team j would put their task about the interface on Team i's code review Wiki page. Then, the members of the two groups worked on each other's tasks. After finishing the tasks, the two groups gave each other feedback. If the results were too diverse, the students could discuss them and the teacher could also make comments.

Methodology

Learning materials

To evaluate the effectiveness of the problem posing-based practicing strategy, a quasi-experiment was conducted in a C# course. This course requires students to gain C# object-oriented programming knowledge (i.e., knowing how a computer program works and the syntax of the programming language) and programming skills (i.e., knowing how to analyze a task and develop a program for dealing with the task). The object-oriented concept is closely linked to the real world by providing several features in programming, such as "encapsulation," "inheritance," and "polymorphism" (Winslow 1996).

Participants and design

A quasi-experiment was adopted, and the participants were 53 sophomores (second year students) from two classes at a university located in eastern China. The learning activity was conducted in a C# programming course. They had already studied C programming in their freshman year and knew the procedure-oriented programming method. One class with 25 students (5 males and 20 females) was assigned to be the treatment group and the other class with 28 students (6 males and 22 females) was the control group. The ratios of male and female students for the two groups were very close; that is, 20 and 80% for the treatment group, and 21 and 79% for the control group. The students in both groups were majoring in Educational Information Technology, and had an average age of 20. All of the students in the treatment group and the control group were taught by the same teacher who has 15 years of teaching experience. During the quasi-experiment, the students' personal information was hidden so as to protect their privacy; moreover, they were informed that joining the experiment was voluntary and would not affect their grades, and they were allowed to withdraw from the experiment at any stage. It should be noted that in the department of Educational Information Technology in many Asian countries, the students need to take courses in both computer technology and educational technology.

In this experiment, through the implementation of the problem posing-based practicing strategy in the collaborative learning activity, learners were guided to link the programming learning with the real world to cultivate their problem-solving conceptions in the real world and to enhance their object-oriented programming thinking. Meanwhile, compared to conventional collaborative learning, whether such an approach can improve students' learning was observed, in terms of learning achievement, cognitive load, and self-efficacy for group learning.

Experimental procedure

Figure 3 presents the experiment process. Before the learning activity, all of the students were instructed by the same teacher for 7 weeks in the C# programming course, including the basic syntax and class of C#. Then, all of the students took a pre-test and completed a pre-questionnaire. The purpose of the pre-test was to evaluate the students' prior knowledge of C# programming and programming skills, while the pre-questionnaire included an assessment of the students' self-efficacy for group learning.

Following that, all of the students received the instruction on the "inheritance" concept in C# programming with corresponding examples for 2 weeks. Then, the treatment group

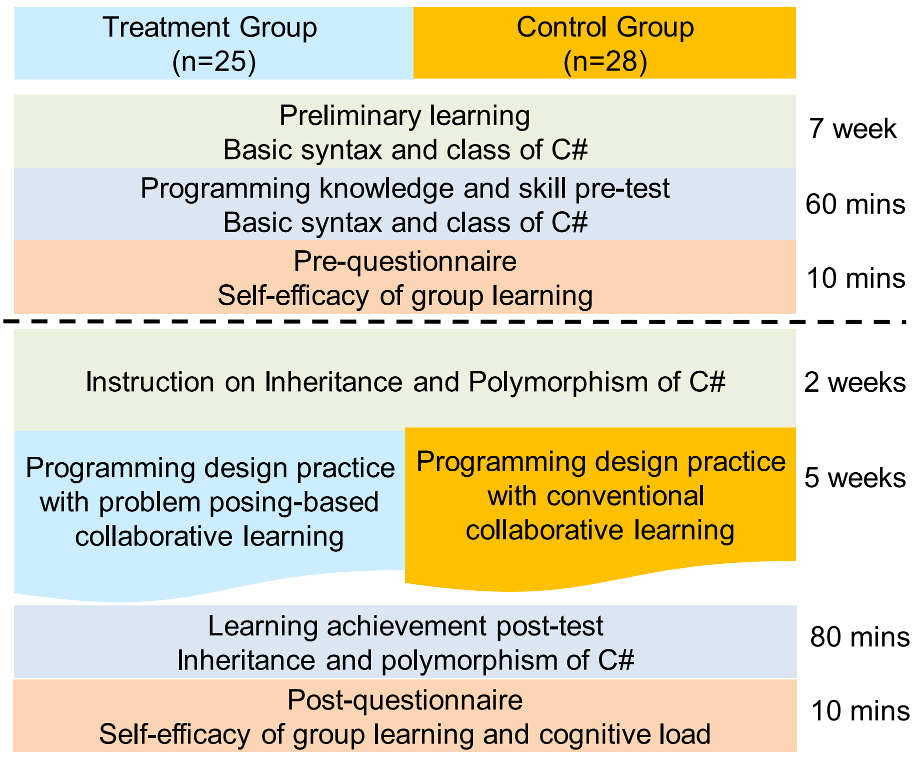

Treatment Group
(n=25)
Control Group
(n=28)
Preliminary learning
Basic syntax and class of C#
7 week
Programming knowledge and skill pre-test
Basic syntax and class of C#
60 mins
Pre-questionnaire
Self-efficacy of group learning
10 mins
Instruction on Inheritance and Polymorphism of C#
2 weeks
Programming design practice with problem posing-based collaborative learning
Programming design practice with conventional collaborative learning
5 weeks
Learning achievement post-test
Inheritance and polymorphism of C#
80 mins
Post-questionnaire
Self-efficacy of group learning and cognitive load
10 mins

Fig. 3 Experimental design of the learning activity

and the control group learned with the two different teaching strategies in the coming 5 weeks.

The students in the treatment group learned with the problem posing-based practicing strategy, using the Wiki page in Moodle to engage in the collaborative problem-posing and code review activities. During the learning activity, the members of the treatment group were requested to finish two programming tasks on inheritance, one of which was proposed by the team members, based on the examples given by the teacher in class, while the other was proposed by the other matching team.

On the other hand, the students in the control group were guided to complete the learning tasks in teams with the conventional collaborative learning approach. After logging into the Moodle system and collaborating with their team members via discussion, they directly completed two programming tasks given by the teacher, and then submitted their results. The teacher then gave feedback on the students' programs and provided example programs to them. The students then reviewed the code provided by the teacher, and examined and corrected their own programs by referring to the feedback provided by the teacher and the example programs. Following that, the teacher answered the questions raised by the students. Such a collaborative learning approach had been employed in the selected school for years.

After the learning activity, a post-test and a post-questionnaire were conducted. The content of the post-test included inheritance and polymorphism of C#, while that of the post-questionnaire included group learning self-efficacy and cognitive load.

Measuring tools

The measuring tools in this study were composed of the pre-test and post-test of learning achievement, the pre-questionnaire of self-efficacy for group learning, and the post-questionnaire of self-efficacy for group learning and cognitive load.

The purpose of the pre-test was to evaluate whether the students in both the treatment group and the control group met the same minimum standard for learning the programming knowledge and skills, while that of the post-test was to investigate whether any difference existed as a result of the two groups of students adopting different learning strategies. There were 14 multiple-choice items (70%) and one design item (30%) in the pre-test, and 8 multiple-choice items (40%) and two design items (60%) in the post-test, both with a perfect score of 100. The multiple-choice items aimed to test the students' C# programming knowledge, while the design items were to test the students' problem-solving programming skills. Both the pre-test and the post-test were designed by two experienced teachers, each with 15 years of programming teaching experience. The Cohen's kappa coefficients of the scores graded by the two teachers were 0.97 and 0.98 for the pre-test and post-test, respectively, showing high consistency between their grading.

The self-efficacy of the group learning questionnaire was adopted from the one developed by Wang and Lin (2007), which was revised from the original questionnaire of Pintrich et al. (1991). There were eight items in the questionnaire, which was used to test the individuals' judgement of group ability and the evaluation of the group ability for the upcoming tasks, such as "I am certain I can understand the most difficult material presented in the readings for this course" and "I am confident I can understand the basic concepts taught in this course." A 5-point Likert scheme was used, with 5 representing *Strongly agree*, 4 representing *Agree*, 3 representing *Neutral,* 2 representing *Disagree*, and 1 representing *Strongly disagree*. The Cronbach's alpha value for the self-efficacy of group learning is .93, showing good internal consistency of the items, as indicated by Christmann and van Aelst (2006) and Sijtsma (2009).

The cognitive load questionnaire was modified by Hwang et al. (2013) based on the cognitive load measures proposed by Paas (1992) and Sweller et al. (1998). It includes two dimensions, mental load (e.g., "The learning content in this learning activity was difficult for me" and "I had to put a lot of effort into answering the questions in this learning activity") and mental efforts (e.g., "The instructional way in the learning activity was difficult to follow and understand"). The former refers to the intrinsic cognitive load caused by the difficulty or large amount of the learning content, while the latter refers to the extraneous cognitive load caused by the teaching strategy (i.e., the way the learning content is organized and presented). There are five items for mental load and three for mental efforts. A 7-point Likert scheme was adopted for each item, with 7 representing *Strongly agree* and 1 representing *Disagree*. A higher value means that the students have higher cognitive loading in the learning process. The Cronbach's alpha values for the two dimensions are .92 and .84, respectively.

Results

Analysis of learning achievement

The analysis of covariance (ANCOVA) was employed by using the pre-test scores as the covariant, the post-test scores as the dependent variable, and the learning approach as the independent variable in order to investigate whether the treatment and control groups showed significant differences in their learning achievements by excluding the interaction effects of the pre-test.

The test of the homogeneity of regression coefficients on the two groups' pre-test scores showed that the homogeneity of variance was not violated with $F = 0.000$ and $p = 0.998 > 0.05$. Following that, ANCOVA was then conducted on the two groups' post-test scores by excluding the impact of the pre-test scores, as shown in Table 1. The results show that there was a significant difference in the scores of the students who adopted the different learning strategies for collaborative learning ($F = 5.09$, $p = 0.028 < 0.05$), with the treatment group scoring higher than the control group. Therefore, it is concluded that the problem posing-based practicing strategy used in collaborative learning can significantly improve students' learning achievement compared with conventional collaborative learning. The Eta squared value $\eta^2 = 0.09$ implies a medium to large effect size.

To further investigate the effects of the different collaborative learning strategies on the students' learning of programming knowledge and programming skills, the scores for the multiple-choice items and design items in the pre- and post-tests were analyzed.

The test of the homogeneity of regression coefficients on the two groups' pre-test scores of programming knowledge and skills showed that the homogeneity of variance was not violated with $F = 0.68$ ($p = 0.42 > 0.05$) and $F = 0.32$ ($p = 0.57 > 0.05$), respectively. That is, ANCOVA can be used to compare the two groups' scores of programming knowledge and skills.

Table 2 shows the ANCOVA results of the two groups' programming knowledge and skills. It can be seen that the adjusted mean scores of the post-test for the multiple-choice items in the treatment group and the control group were 25.77 and 24.52, respectively, meaning that with the different collaborative learning strategies, the students still had similar basic knowledge of C# ($F = 0.686$, $p = 0.41 > 0.05$). On the other hand, the ANCOVA results showed that the adjusted mean scores of the post-test for the design items in the treatment group and the control group were 32.46 and 24.31, meaning that with the different collaborative learning strategies, a significant difference was reached in the design items of the post-test score, with the mean score of the treatment group higher than that of the control group ($F = 5.67$, $p = 0.021 < 0.05$). The Eta squared value $\eta^2 = 0.1$ implies a medium to large effect size.

Table 1 The ANCOVA result of the learning achievements post-test for the two groups

Group	N	Mean	SD	Adjusted Mean	SE	*F*	η^2
Treatment group	25	57.88	15.48	57.85	2.83	5.09*	0.09
Control group	28	49.11	17.27	49.14	2.68		

* p <.05

Table 2 The ANCOVA result of the programming knowledge and skills post-test for the two groups

Variance	Group	N	Mean	SD	Adjusted mean	SE	*F*	η^2
Programming knowledge	Treatment	25	25.77	5.72	25.75	1.09	0.69	0.01
	Control	28	24.52	5.50	24.51	1.03		
Programming skill	Treatment	25	32.08	12.47	32.46	2.49	5.67*	0.10
	Control	28	24.64	14.65	24.31	2.35		

* p <.05

Analysis of cognitive load

An independent *t* test analysis was conducted on the post-questionnaire of cognitive load, as presented in Table 3. It can be seen that the mean score of cognitive load for the control team was 4.66, while that for the treatment group was 3.70, meaning that the students who learned with the problem posing-based practicing strategy had significantly lower cognitive load than the students in the control group ($t = 2.69$, $p = 0.01 < 0.05$). The Cohen's d is 0.75, showing a medium to large effect size.

Analysis of self-efficacy of group learning

ANCOVA was employed by using the pre-questionnaire ratings as the covariant, the post-questionnaire ratings as the dependent variable, and the learning approach as the independent variable in order to investigate whether the treatment and control groups showed significant differences in their group learning self-efficacy by excluding the interaction effects of the pre-questionnaire ratings.

The test of the homogeneity of regression coefficients on the two groups' pre-questionnaire ratings of self-efficacy of group learning showed that the homogeneity of variance was not violated with $F = 0.453$ with $p > 0.05$. ANCOVA was then conducted on the post-questionnaire ratings of self-efficacy of group learning. Table 4 shows the results for the two groups. It can be seen that the adjusted mean scores for the treatment group and the control group were 4.18 and 3.76, meaning that the treatment group significantly improved their self-efficacy of group learning, compared to the control group ($F = 4.90$, $p = 0.03 < 0.05$). The Eta squared value $\eta^2 = 0.09$ implies a medium to large effect size. The results showed that the students learning with the problem posing-based practicing strategy collaborative learning showed better self-efficacy for group learning than those learning with the conventional approach. This result matches the result that the treatment group students had higher learning achievement and lower cognitive load.

Table 3 Independent sample *t* test of the cognitive loads of the two groups

Group	N	Mean	SD	*t*	*p*	*d*
Treatment	25	3.70	1.21	2.69*	0.01	0.75
Control	28	4.66	1.39			

* p <.05

Table 4 The ANCOVA result of the self-efficacy of group learning post-test for the two groups

Group	N	Mean	SD	Adjusted mean	SE	*F*	η^2
Treatment	25	4.20	0.50	4.18	0.14	4.90*	0.09
Control	28	3.74	0.77	3.76	0.13		

* $p < .05$

Discussion and conclusions

A problem posing-based practicing strategy was proposed in this study. The results present that, compared to conventional collaborative learning, the problem posing-based practicing strategy can significantly improve students' programming skills and group learning self-efficacy, while also lowering their cognitive load.

In terms of learning achievement, some studies showed that learners can gain positive learning achievement by having course content-related learning tasks and guiding them to actively engage in construction, creation, and reflection activities to promote peer interaction as well as their problem-solving abilities (Lai and Hwang 2015; Hardy et al. 2014; Auttawutikul et al. 2014). It is inferred that the problem-posing with follow-up code review activities was in line with the social constructivist theory, which emphasizes the impacts of social interactions on students' cognitive process. In this study, the aim of the proposed approach was to engage learners in a process whereby they could learn from each other and receive comments or assistance from their peers. Such an approach offered the treatment group students not only opportunities to deal with the problems raised by their peers and to receive comments from their peers, but also to review their peers' work, which situated them in a learning environment to learn from each other. Such an activity reinforced the learners' active participation and responsible attitude towards engaging in learning. Furthermore, the code review activity improved the learners' interaction and gave them a chance to reflect on and reinforce their understanding of programming problems and coding. As indicated by Wang et al. (2012), engaging students in peer-review tasks enables them to reflect on and reinforce their understanding of the learning content. In the learning process of the problem posing-based practicing strategy, the learners continued participating in the interaction of knowledge construction, problem creation, problem solving, comment giving, and reflection. This might be the reason why the treatment group outperformed the control group in terms of learning achievement.

In terms of cognitive load, Antonenko and Niederhauser (2010) indicated that, in addition to intrinsic load and extraneous load, germane load also contributes to the total load caused by learning materials or tasks. They also indicated that, while germane load improved cognition and learning, extraneous load did the opposite owing to the limitations caused by the learning design. In this study, after teacher instruction in the control group, the students followed the programming tasks given by the teacher to find the answer; following that, they reviewed their own programming code as well as the feedback and standard answer provided by the teacher to correct their code. On the other hand, the students in the treatment group were provided with example tasks at the beginning of the learning activity. With such scaffolding, they could pose a problem within a certain knowledge range and collaborate to solve it. During the process, the learners were not limited to one single task and could be more actively engaged and creative, which can meet the goal-free effect of cognitive load. This might be the reason why the treatment group was able to have not only better learning achievement, but also lower cognitive load. The study of Antonenko and Niederhauser (2010) also supports this point; that is, the goal-free effect allows learners to express themselves without limitation, triggers creation, and hence lowers their cognitive load and helps them achieve meaningful learning (Sweller 1994, 2010).

It should be noted that there are several limitations to this study. First, the sample size of the participants was not large, and hence the findings cannot be generalized to represent the cases with a large number of learners. Second, in the present study, the proposed approach was applied to the selected units of the programming course for 7 weeks; therefore, the impacts of the strategy cannot be generalized to all of the computer programming units. Third, the effect sizes for the statistical results of cognitive load and self-efficacy were not large, implying that the generalization of the findings in these dimensions is limited. In addition, in the proposed strategy, it is assumed that the learners have equivalent basic programming knowledge for posing problems in the first stage as well as reviewing peer work in the second stage. For other courses in which learners are from different departments with different levels of basic knowledge or skills, additional considerations and treatments could be required.

For future research, it is suggested that long-term experiments with large sample sizes are required. It is also worth applying the strategy to other computer programming units to investigate its impacts from different aspects, such as the tendencies of collaboration and communication in team-based programming.

Acknowledgments This study is supported in part by the Key Research Base of Humanities and Social Sciences in Colleges and Universities of Zhejiang Province (Zhejiang Normal University Education Base).

Compliance with ethical standards

Conflict of interest The authors would like to declare that there is no conflict of interest in this study.

References

Antonenko, P. D., & Niederhauser, D. S. (2010). The influence of leads on cognitive load and learning in a hypertext environment. *Computers in Human Behavior, 26*(2), 140–150.

Au, K. H. (1998). Social constructivism and the school literacy learning of students of diverse backgrounds. *Journal of literacy research, 30*(2), 297–319.

Auttawutikul, S., Wiwitkunkasem, K., & Smith, D. R. (2014). Use of weblogs to enhance group learning and design creativity amongst students at a Thai University. *Innovations in Education and Teaching International, 51*(4), 378–388.

Barlow, A. T., & Cates, J. M. (2006). The impacts of problem posing on elementary teachers' belief about mathematics and mathematics teaching. *School Science and Mathematics, 106,* 64–73.

Blignaut, P., & Naude, A. (2008). The influence of temperament style on a student's choice of and performance in a computer programming course. *Computers in Human Behavior, 24*(3), 1010–1020.

Bloom, B. S. (1994). Reflections on the development and use of the taxonomy. *Yearbook: National Society for the Study of Education, 92*(2), 1–8.

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.

Bravo, C., Marcelino, M. J., Gomes, A. J., Esteves, M., & Mendes, A. J. (2005). Integrating educational tools for collaborative computer programming learning. *J. UCS, 11*(9), 1505–1517.

Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies, 18*(6), 543–554.

Brush, T. A. (1998). Embedding cooperative learning into the design of integrated learning systems: rationale and guidelines. *Educational Technology Research and Development, 46*(3), 5–18.

Casagrande, L. D. R., Caron-Ruffino, M., Rodrigues, R. A. P., Vendrusculo, D. M. S., Takayanagui, A. M. M., Zago, M. M. F., et al. (1998). Problem-posing in education: Transformation of the practice of the health professional. *Patient Education and Counseling, 33*(2), 161–167.

Christmann, A., & Van Aelst, S. (2006). Robust estimation of Cronbach's alpha. *Journal of Multivariate Analysis, 97*(7), 1660–1674.

diSessa, A. A., & Abelson, H. (1986). Boxer: A reconstructible computational medium. *Communications of the ACM, 29*(9), 859–868.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and learning of computer programming through the use of the Second Life virtual world. *British Journal of Educational Technology, 42*(4), 624–637.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Journal of Research and Development, 15*(3), 182.

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.

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.

Hadjerrouit, S. (2008). Towards a blended learning model for teaching and learning computer programming: A case study. *Informatics in Education, 7*(2), 181–210.

Hardy, J., Bates, S. P., Casey, M. M., Galloway, K. W., Galloway, R. K., Kay, A. E.,… & McQueen, H. A. (2014). Student-generated content: Enhancing learning through sharing multiple-choice questions. *International Journal of Science Education*, *36*(13), 2180-2194

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education, 54*(4), 1127–1136.

Hung, C. M., Hwang, G. J., & Wang, S. Y. (2014). Effects of an integrated mind–mapping and problem–posing approach on students' in–field mobile learning performance in a natural science course. *International Journal of Mobile Learning and Organisation, 8*(3–4), 187–200.

Hwang, G. J., Yang, L. H., & Wang, S. Y. (2013). A concept map-embedded educational computer game for improving students' learning performance in natural science courses. *Computers & Education, 69*, 121–130.

Isong, B. (2014). A methodology for teaching computer programming: first year students' perspective. *International Journal of Modern Education and Computer Science, 6*(9), 15.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational technology research and development*, *48*(4), 63–85." based on this comment.

Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*(1), 33.

Kao, G. Y. M., Lin, S. S., & Sun, C. T. (2008). Beyond sharing: Engaging students in cooperative and competitive active learning. *Educational Technology & Society, 11*(3), 82–96.

Laal, M., & Ghodsi, S. M. (2012). Benefits of collaborative learning. *Procedia-Social and Behavioral Sciences, 31,* 486–490.

Laal, M., Naseri, A. S., Laal, M., & Khattami-Kermanshahi, Z. (2013). What do we achieve from learning in collaboration? *Procedia-Social and Behavioral Sciences, 93,* 1427–1432.

Lai, C. L., & Hwang, G. J. (2015). An interactive peer-assessment criteria development approach to improving students' art design performance using handheld devices. *Computers & Education, 85*, 149–159.

Lavy, I., & Bershadsky, I. (2003). Problem posing via "what if not?" strategy in solid geometry—a case study. *The Journal of Mathematical Behavior, 22*(4), 369–387.

Law, Q. P. S., Chung, J. W. Y., Leung, C. C., & Wong, T. K. S. (2015). Enhancement of self-efficacy and interest in learning english of undergraduate students with low english proficiency through a collaborative learning programme. *American Journal of Educational Research, 3*(10), 1284–1290.

Law, K. M., Lee, V. C., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education, 55*(1), 218–228.

Liu, E. Z. F., Lin, S. S., Chiu, C. H., & Yuan, S. M. (2001). Web-based peer review: the learner as both adapter and reviewer. *IEEE Transactions on Education, 44*(3), 246–251.

Mantyla, M. V., & Lassenius, C. (2009). What types of defects are really discovered in code reviews? *IEEE Transactions on Software Engineering, 35*(3), 430–448.

Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional Science, 26*(1–2), 49–63.

Merrill, D. (1991). Constructivism and instructional design. *Educational Technology, 31*(5), 45–53.

Moons, J., & De Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education, 60*(1), 368–384.

Moses, B., Bjork, E., & Goldenberg, E. P. (1990). Beyond problem solving: Problem posing. In T. J. Cooney (Ed.), *Teaching and learning mathematics in the 1990s* (pp. 82–91). Reston, VA: National Council of Teachers of Mathematics.

Mow, I. C. (2008). *Issues and difficulties in teaching novice computer programming. In Innovative techniques in instruction technology, e-learning, e-assessment, and education* (pp. 199–204). Berlin: Springer.

Paas, F. G. W. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: a cognitive load approach. *Journal of Educational Psychology, 84*(4), 429–434.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Pintrich, P.R., Smith, D.A.F., Garcia, T., & McKeachie, W.J. (1991). A manual for the use of the motivated strategies for learning questionnaire (MSLQ). MI: National Center for Research to Improve Postsecondary Teaching and Learning. (ERIC Document Reproduction Service No. ED 338122)

Piteira, M., & Costa, C. (2013, July). 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.

Preston, D. (2005). Pair programming as a model of collaborative learning: a review of the research. *Journal of Computing Sciences in Colleges, 20*(4), 39–45.

Rick, J., Guzdial, M., Holloway-Attaway, K. C. L., & Walker, B. (2002, January). Collaborative learning at low cost: CoWeb use in English composition. In: *Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community* (pp. 435–442). International Society of the Learning Sciences.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172.

Rogalski, J., & Samurçay, R. (1990). Acquisition of programming knowledge and skills. *Psychology of Programming, 18,* 157–174.

Sabin, R. E., & Sabin, E. P. (1994, March). Collaborative learning in an introductory computer science course. In: ACM SIGCSE Bulletin (Vol. 26, No. 1, pp. 304–308). ACM.

Serrano-Cámara, L. M., Paredes-Velasco, M., Alcover, C. M., & Velazquez-Iturbide, J. Á. (2014). An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. *Computers in Human Behavior, 31,* 499–508.

Shaw, R. S. (2012). A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums. *Computers & Education, 58*(1), 111–120.

Shuk-kwan, S. L. (2013). Teachers implementing mathematical problem posing in the classroom: challenges and strategies. *Educational Studies in Mathematics, 83*(1), 103–116.

Sijtsma, K. (2009). On the use, the misuse, and the very limited usefulness of Cronbach's alpha. *Psychometrika, 74*(1), 107.

Silver, E. A. (2013). Problem-posing research in mathematics education: Looking back, looking around, and looking ahead. *Educational Studies in Mathematics, 83*(1), 157–162.

Singer, F. M., & Voica, C. (2013). A problem-solving conceptual framework and its implications in designing problem-posing tasks. *Educational Studies in Mathematics, 83*(1), 9–26.

Soller, A. (2001). Supporting social interaction in an intelligent collaborative learning system. *International Journal of Artificial Intelligence in Education, 12,* 40–62.

Soloway, E. (1993). Should we teach students to program? *Communications of the ACM, 36*(10), 21–25.

Sung, H. Y., & Hwang, G. J. (2013). A collaborative game-based learning approach to improving students' learning performance in science courses. *Computers & Education, 63*, 43–51.

Sung, H. Y., Hwang, G. J., & Chang, Y. C. (2016). Development of a mobile learning system based on a collaborative problem-posing strategy. *Interactive Learning Environments, 24*(3), 456–471.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction, 4*(4), 295–312.

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational psychology review, 22*(2), 123–138.

Sweller, J., Van Merriënboer, J. J. G., & Paas, F. G. W. C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review, 10*(3), 251–297.

Unal, H., & Arikan, E. E. (2015). An investigation of eighth grade students' problem posing skills (Turkey sample). *International Journal of Research in Education and Science, 1*(1), 23–30.

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.

Vygotsky, L. S. (1962). *Thought and language*. Cambridge, MA: MIT Press.

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education, 59*(2), 412–422.

Wang, S. L., & Lin, S. S. (2007). The effects of group composition of self-efficacy and collective efficacy on computer-supported collaborative learning. *Computers in Human Behavior, 23*(5), 2256–2268.

Wang, T., Su, X., Ma, P., Wang, Y., & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. *Computers & Education, 56*(1), 220–226.

Wichmann, A., & Rummel, N. (2013). Improving revision in wiki-based writing: Coordination pays off. *Computers & Education, 62,* 262–270.

Winslow, L. E. (1996). Programming pedagogy-a psychological overview. *ACM SIGCSE Bulletin, 28*(3), 17–22.

Yakimovicz, A. D., & Murphy, K. L. (1995). Constructivism and collaboration on the Internet: case study of a graduate class experience. *Computers & Education, 24*(3), 203–209.

Yang, T. C., Hwang, G. J., Yang, S. J., & Hwang, G. H. (2015). A two-tier test-based approach to improving students' computer-programming skills in a web-based learning environment. *Journal of Educational Technology & Society, 18*(1), 198.

Zhang, Y., & Yan, D. (2014). Curriculum reform of C language programming and cultivation of computational thinking. *Advances in Natural Science, 7*(4), 49–53.

Xiao-Ming Wang is a lecture in the School of Teacher Education, Zhejiang Normal University, China. His research interests include computer education, computational thinking and web-based learning.

Gwo-Jen Hwang is a Chair Professor at National Taiwan University of Science and Technology, Taipei, Taiwan. Dr. Hwang has published more than 330 academic papers. His research interests include e-learning, computer mediated communication, intelligent tutoring system, and expert system. Owing to the good reputation in academic research and innovative inventions of e-learning, in 2007 and 2011, he received the annual Most Outstanding Researcher Award from the National Science Council in Taiwan.