Journal of Computing in Higher Education
https://doi.org/10.1007/s12528-024-09407-3

The effects of pre-training types on cognitive load, self-efficacy, and problem-solving in computer programming

Jaewon Jung[1] · **Yoonhee Shin**[2] · **HaeJin Chung**[3] · **Mik Fanguy**[4]

Accepted: 10 June 2024

Abstract
This study investigated the effects of pre-training types on cognitive load, self-efficacy, and problem-solving in computer programming. Pre-training was provided to help learners acquire schemas related to problem-solving strategies. 84 undergraduate students were randomly assigned to one of three groups and each group received three different types of pre-training: 1) WOE (worked-out example) and metacognitive scaffolding, 2) faded WOE and metacognitive scaffolding, and 3) WOE and faded metacognitive scaffolding. After the pre-training phase, the participants' cognitive load, self-efficacy, and programming problem-solving skills were analyzed. Then, during the training phase, the participants were asked to attempt a programming problem-solving task with faded WOE and faded metacognitive scaffoldings. After the training phase, the participants' cognitive load, self-efficacy, and programming problem-solving were analyzed again. The findings revealed that providing both cognitive scaffolding (i.e., WOE or faded WOE) and non-faded metacognitive scaffolding during the pre-training phase is effective for novice learners for optimizing cognitive load, promoting self-efficacy, and enhancing programming problem-solving skills.

Keywords Metacognitive scaffolding · Faded WOE · Cognitive load · Self-efficacy · Problem-solving programming

Introduction

With the increasing importance of computer programming skills in twenty-first century critical competencies (Wu et al., 2020), various instructional strategies have been proposed for effective programming instruction. According to previous studies, self-regulation skills, including cognitive and metacognitive skills, are important to solve complex tasks when programming (Loksa et al., 2016; Shin et al.,

Extended author information available on the last page of the article

Published online: 27 June 2024

2023; Zimmerman & Schunk, 2011). Self-regulation skills refer to strategies learners employ in order to systematize their thinking and behavior to apply them to achievement of a learning goal (Berk, 2003). Specifically, learners with high academic achievement use more self-regulated learning strategies than learners with low achievement (Hwang & Vrongistnos, 2002), and self-regulation increases learning motivation or self-efficacy, giving learners an autonomous and active attitude (Deweck & Leggett, 1988). However, novice learners may experience difficulties in solving programming problems because they lack the self-regulation ability to understand computational principles and apply them effectively to programming (Loksa et al., 2020; Magana et al., 2019) and because low self-efficacy may prevent successful learning. Pre-training is one of the ways to overcome such difficulties, and it is necessary to understand important concepts and practice reflective thinking in the problem-solving process through pre-training.

Instructional strategies that consider both cognitive and metacognitive aspects are important when providing pre-training for improving the self-regulation skill of novice computer programmers. Specifically, if cognitive skills are necessary for understanding computational principles, metacognitive skills can be seen as necessary for overall programming problem-solving (Shin & Song, 2022; Shin et al., 2023). Worked-out example (WOE) is a type of instructional strategy that in which the learner is guided through a stepwise demonstration of how to solve a problem or complete at task (Kirschner et al., 2006). Shin et al. (2023) found that providing metacognitive scaffolding to promote metacognitive skills together with a WOE to promote cognitive skills in programming learning was more effective in programming knowledge acquisition and problem-solving than providing a WOE alone. In addition, providing a WOE with metacognitive scaffolding was found to be effective in reducing unnecessary cognitive load and promoting germane cognitive load (Molenaar et al., 2011; Prather et al., 2020). In terms of metacognition, metacognitive scaffolding is effective in structuring the solution process through planning in the early phase of problem-solving and applying metacognitive strategies through monitoring and reflection in the problem-solving phase (Molenaar et al., 2011; Prather et al., 2020). In terms of cognition, providing well-structured WOE can help learners learn the computational principles necessary for programming problem-solving (Garner, 2002). However, it is important to apply appropriate fading, meaning a reduction in the level of assistance or guidance provided to learners in completing a task or solving a problem, to WOE in order optimize cognitive load and effectively apply acquired knowledge (Garner, 2001; Renkl, 2002; Salleh et al., 2018).

As mentioned above, despite research findings that cognitive and metacognitive support need to be provided together to improve self-regulation abilities in programming learning, so far, most studies have provided only cognitive or metacognitive scaffolding. Moreover, even if cognitive and metacognitive scaffoldings are provided together, there is a lack of research on which types of scaffolding provide more synergistic effects to novice learners' understanding of computer programming. In previous research, Zheng et al. (2022) revealed findings indicating that scaffolding strategies where scaffolding is gradually introduced are more effective than scaffolding strategies where scaffolding is gradually removed. However, there

remains a scarcity of studies investigating the effectiveness of faded metacognitive scaffolding strategies. Furthermore, despite the importance influence of self-efficacy on self-regulated learning (Shell et al., 1995), there is a paucity of research exploring the relationship between scaffolding types aimed at supporting self-regulation in programming education and self-efficacy. Therefore, the present study focuses on pre-training to support the promotion of cognitive and metacognitive skills, which are components of learners' self-regulation ability, and seeks to explore how pre-training types affect novice learners' programming learning. Through this, we intend to present empirical evidence to identify the type of pre-training that is most effective for programming instruction.

Literature review

Programming and pre-training

Recently, as interest in computational thinking has increased, the importance of problem-solving in programming has increased. In order to successfully solve complex problems when learning to program, self-regulation skill to apply various solutions based on the principles necessary for problem-solving is required (Loksa et al., 2016; Zimmerman & Schunk, 2011). Self-regulation skill enhances learners' self-efficacy (Deweck & Leggett, 1988), and learners with high self-efficacy use more self-regulated learning strategies (Zimmerman & Martinez-Pons, 1990). However, novice learners lack self-regulation skill, which can make it difficult to apply appropriate programming-related knowledge to solve programming problems (Magana et al., 2019).

For example, a study examining the problem-solving strategies of experts in programming has revealed that experts adopt a problem-solving approach focused on knowledge clarification and strategic planning. During the problem-analysis stage, experts tend to demonstrate a clear understanding of the problem and effectively developed a well-structured plan to arrive at a solution (Chao, 2016; Lister et al., 2006; Loksa et al., 2016; Xie et al., 2019). Novices, on the other hand, tend to approach the solution without employing any specific problem-solving strategies, resulting in the manifestation of simplistic and repetitive errors during the implementation phase (Loksa et al., 2020; Magana et al., 2019). In other words, novice learners lack the skills to solve problems through a series of processes such as systematic planning of their programming, testing the programming output, and evaluating programming solutions. A possible way to assist novice learners during the problem-solving process of programming is to provide instruction that aids in their understanding of computational principles and enables them to practice applying metacognitive strategies through pre-training in the process of problem-solving.

Previous studies have shown that for complex tasks, pre-training can improve learners' expertise and effectively support problem-solving (Jung et al., 2021; Kalyuga et al., 2001). Pre-training has been shown to reduce unnecessary cognitive load and improve comprehension of learning content by enabling learners to pre-learn core concepts or core content in the process of solving complex

problems (Jung et al., 2016). Pre-training has been applied to diverse fields such as multimedia learning (Mayer et al., 2002) and computer-based collaborative learning (CSCL) (Jung et al., 2021). In the present study, we intend to apply pre-training to help learners solve computer programming problems.

Worked-out examples and cognitive load

Cognitive load occurs because the amount of information that can be processed at one time by human working memory is fixed, and in order to lead learning effectively, an instructional strategy that considers cognitive load is required (Mayer & Moreno, 2003; Sweller et al., 1998). Cognitive load can be divided into intrinsic cognitive load, extraneous cognitive load, and germane cognitive load. Intrinsic load primarily pertains to the complexity of the content that students need to comprehend, extraneous load relates to factors that hinder comprehension during information transfer, and germane load refers to the information that learners have absorbed (Sweller et al., 1998).

Previous studies have proposed various instructional design strategies to reduce extraneous cognitive load and promote germane cognitive load beneficial to learning (Paas & Sweller, 2012; Renkl & Atkinson, 2003). Among them, a representative example for controlling the cognitive load of novice learners is providing WOE, which presents an expert's problem-solving process (Clarke et al., 2005; Renkl & Atkinson, 2003). In learning to program, novice learners can experience cognitive load in the process of attempting to comprehend basic programming-related concepts and applying appropriate problem solutions to problems. Accordingly, in previous studies, WOE was provided to support the cognitive aspect to help acquire computational principles (Garner, 2002). Faded WOE, in particular, can maximize learning outcomes by eliminating the key parts from expert's problem-solving process and allowing learners to figure out the solution on their own (Hancock-Niemi et al., 2016; Renkl et al., 2000; Salleh et al., 2018; van Merriënboer & de Croock, 1992). However, when the use of a faded WOE that does not match the learner's prior knowledge level, unnecessary information is provided redundantly or necessary information is not provided, causing extraneous cognitive load for the learner (Seta et al., 2007). Therefore, when using faded WOE, it is important to apply fading appropriately in consideration of the learner's expertise (Merriënboer & Kirschner, 2012; de Jong, 2010).

Recently, some studies have found that providing cognitive and metacognitive support together in programming instruction for novice learners is more effective in controlling cognitive load, acquiring programming-related knowledge, and solving problems than providing only cognitive support (Pol et al., 2010; Chen et al., 2023; Shin & Song, 2022; Shin et al., 2023). Therefore, programming learning for novice learners should be provided with instructional strategies that consider both cognitive and metacognitive aspects. Currently, research is needed to ascertain effective combinations of cognitive and metacognitive support.

Metacognitive scaffolding in programming

Recently, the provision of metacognitive scaffolding has been utilized as a strategy to promote learners' metacognition and support metacognitive behaviors as they engage in problem-solving during programming. Previous research on programming instruction has shown that metacognitive scaffolding is effective in improving learners' problem-solving skills in that it provides a means to check what learners already know and supports them in selecting appropriate problem-solving strategies (Shin & Song, 2022). Specifically, metacognitive scaffolding plays an important role in leveraging metacognitive strategies such as planning the programming, monitoring programming output, and reflecting the programming solutions necessary for the entire process of programming troubleshooting (Shin et al., 2023).

According to Mohd Rum and Ismail (2017), novice learners provided with metacognitive scaffolding exhibited superior programming performance to learners not provided with metacognitive scaffolding. This indicates that by providing novice learners with an expert's fine problem-solving strategies through metacognitive scaffolding, the process of goal setting, organizing knowledge, enacting strategies, evaluating a potential solution, and implementing a solution of novice learners was effectively supported. In particular, given that proper fading of a WOE promotes cognition (Renkl et al., 2000), providing a faded WOE and metacognitive scaffolding together can be expected to be effective in solving programming problems. Moreover, it is expected that this approach has a positive effect on the improvement of learners' problem-solving and self-efficacy because learners can concentrate on essential information from experts' problem-solving processes (Sweller, 2010) and infer the principles required at each stage of the problem-solving process through the eliminated parts of the WOE (Chi et al., 1989; Renkl, 2002).

In a study exploring effective metacognitive scaffolding types for novice learners in collaborative programming settings (Zheng et al., 2022), fade-in scaffolding (where scaffolding is gradually introduced) was found to be more effective than fade-out scaffolding (where scaffolding is gradually removed). The study provided empirical support for the notion that fade-in scaffolding is an effective metacognitive scaffolding type for collaborative programming of novice learners. However, research on the effect of faded metacognitive scaffolding is still lacking. Despite various studies on metacognitive scaffolding for novice learners in programming instruction, the literature remains rather porous. Although there are many studies that apply fading to cognitive scaffolding such as WOE in individual programming learning, few attempts have investigated how to apply fading to metacognitive scaffolding.

The present study

In the present study, we aim to provide both cognitive and metacognitive scaffolding together to promote learners' cognitive and metacognitive skills, thereby optimizing cognitive load, promoting self-efficacy, and enhancing problem-solving skills. Specifically, we will examine which types of metacognitive scaffolding are

most effective for programming instruction among novice learners. Additionally, we will explore the potential synergistic effect of providing cognitive and metacognitive scaffolding together. Specifically, 84 undergraduate students were randomly assigned to one of three groups respectively receiving three different types of pre-training: 1) WOE (worked out example) and metacognitive scaffolding, 2) faded WOE and metacognitive scaffolding (FWM), and 3) WOE and faded metacognitive scaffolding (WFM). The participants' levels of cognitive load, self-efficacy, and programming problem-solving were assessed after the pre-training phase, and then during the training phase, the participants were asked to attempt a programming problem-solving task with faded WOE and faded metacognitive scaffoldings. After the training phase, the participants' cognitive load, self-efficacy, and programming problem-solving were assessed again.

Research questions and hypotheses

The present study was guided by the following research questions and corresponding hypotheses:

RQ1 Does the use of fading in pre-training programs containing both WOE and metacognitive scaffolding help control learners' cognitive load?H1 Learners engaged in pre-training that includes fading will exhibit more optimized cognitive load compared to learners engaged in pre-training that omits fading.H1a Learners in the FWM and WFM conditions will exhibit lower levels of intrinsic cognitive load than learners in the WM condition.H1b Learners in the FWM and WFM conditions will exhibit lower levels of extraneous cognitive load than learners in the WM condition.

RQ2 If so, is it more beneficial to learners' cognitive load to fade the WOE or to fade the metacognitive scaffolding?H2 Fading the WOE will be more beneficial to learners' cognitive load than fading the metacognitive scaffolding.H2a Learners in the FWM conditions will exhibit lower levels of intrinsic cognitive load than learners in the WFM condition.H2b Learners in the FWM conditions will exhibit lower levels of extraneous cognitive load than learners in the WFM condition.H2c Learners in the FWM conditions will exhibit higher levels of germane cognitive load than learners in the WFM condition.

RQ3 Does the use of fading in pre-training programs containing both WOE and metacognitive scaffolding promote learners' self-efficacy?H3 Learners engaged in pre-training that includes fading will exhibit improved self-efficacy compared to learners engaged in pre-training that omits fading.H3a Learners in the FWM and WFM conditions will exhibit higher levels of self-efficacy than learners in the WM condition.

RQ4 If so, is it more beneficial to learners' self-efficacy to fade the WOE or to fade the metacognitive scaffolding?H4 Fading the WOE will be more beneficial

to learners' self-efficacy than fading the metacognitive scaffolding.H4a Learners in the FWM conditions will exhibit higher levels of self-efficacy than learners in the WFM condition.

RQ5 Does the use of fading in pre-training programs containing both WOE and metacognitive scaffolding benefit learners' programming problem-solving?H5 Learners engaged in pre-training that includes fading will exhibit enhanced programming problem-solving compared to learners engaged in pre-training that omits fading.

RQ6 If so, is it more beneficial to learners' programming problem-solving to fade the WOE or to fade the metacognitive scaffolding?H6 Fading the WOE will be more beneficial to learners' programming problem-solving than fading the metacognitive scaffolding.H6a Learners in the FWM condition will exhibit higher levels of programming problem-solving than learners in the WFM condition.

Method

Participants

The participants in this study were 84 female first-year undergraduate students majoring in science and technology at a 4-year women's university. The participants were enrolled in a 2-credit course titled "Introduction to Programming," and none had experience taking courses related to Python programming. In addition, students had no prior experience taking courses related to computational thinking skills and problem-solving skills at a university. The average age of the students was 20.35 years. All participants were randomly assigned to one of three groups, and each group was provided with different types of pre-training: traditional WOE and metacognitive scaffolding (WM), faded WOE and metacognitive scaffolding (FWM), and traditional WOE and faded metacognitive scaffolding (WFM). All of groups carried out programming problem-solving tasks for three weeks using a web-based programming tool.

Experimental materials

Three types of pre-training

During their pre-training, the participants performed basic-level programming problem-solving tasks. The purpose of the pre-training was to demonstrate an understanding of the basic concepts and accurately arrive at a solution through a process of planning, monitoring, and reflection. The three types of pre-training were constructed as WM, FWM, and WFM.

Worked-out example

The WOE used in this study was designed based on WOE developed by Shin et al. (2023). The WOE used herein included problem statements and objectives in a "problem" section as well as a basic program structure and problem-solving context, including input data and output data, in a "solution" section. While traditional WOE included a guide with all the code, faded WOE was provided with core and important concepts being removed (e.g., variables, control statements, functions, etc.) with 2–3 concepts per problem (see Fig. 1).

Metacognitive scaffolding

Metacognitive scaffolding was provided to assist novices in planning, monitoring, and reflecting while solving programming problems. The planning guide aimed to help learners construct a plan to reach the group's goal such as the choosing learning strategies and managing timelines. The monitoring guide provided instructions on conducting a systematic analysis of solutions, and the reflection guide facilitated novice learners in discerning their acquired knowledge and areas of oversight. The metacognitive questions were developed based on Loksa et al. (2016) and Shin et al. (2023) (see Table 1). Metacognitive scaffolding included a guide with metacognitive questions and a sample of experts' metacognitive strategies with core metacognitive concepts regarding planning, monitoring,

Problem: You are a sports event organizer, hosting a running race. In this race, to ensure an accurate start, a countdown program is required that starts counting down from the number entered by the user and signals the start when the count reaches zero. **Step 1. Write what form to declare the variable to store the item's name.** count = int (input("Please enter the initial value.: ")) **Step 2. Set up a loop that starts the countdown from the number entered by the user and prints it.** for x in range (count, 0, -1) : print(x) **Step 3. After the loop ends, write code to output a message saying "Start".** print("Start")	**Problem:** You are a sports event organizer, hosting a running race. In this race, to ensure an accurate start, a countdown program is required that starts counting down from the number entered by the user and signals the start when the count reaches zero. **Step 1. Write what form to declare the variable to store the item's name.** count = int (input("Please enter the initial value.: ")) **Step 2. Set up a loop that starts the countdown from the number entered by the user and prints it.** for x in range (count, __(a)__, __(b)__) : print(_(c)_) **Step 3. After the loop ends, write code to output a message saying "Start".** print("Start")

Fig. 1 Problem-solving WOE examples (left: traditional WOE (WM), right: faded WOE (FWM))

Table 1 Sample metacognitive scaffolding questions

Strategies	Metacognitive questions
Planning	What variables and key phrases are needed to solve the problem?
Monitoring	Has the detailed goal setting for problem-solving been accomplished?
Reflection	What did you learn from this task?

and reflecting, while faded metacognitive scaffolding included a guide with only metacognitive questions with a sample of experts' metacognitive strategies with some core concepts faded.

Pre-test

Ten multiple-choice questions about basic concepts and syntax of Python programming were used to measure the participants' prior knowledge (e.g., "Choose the appropriate principle when declaring a variable"). One point was provided for a correct answer to each problem, with a maximum score of 10 points (see Table 2).

There were no significant differences in prior knowledge across the three conditions. Specifically, the result shows that there was no statistically significant difference among three groups ($F = 0.96$, $p = 0.39$, $\eta 2 = 0.02$) (see Table 3).

Cognitive load measures

A ten-point Likert scale was used to measure cognitive load, based on previous research by Leppink et al., 2013, ranging from "Extremely Easy" (0) to "Extremely Difficult" (10) (see Table 4). The cognitive load measurement consisted of 10

Table 2 Sample pre-test items to measure prior knowledge

Items#	Problems
1	Choose the one that is appropriate as a principle when declaring a variable
2	Choose the one that is appropriate for the description of the conditional statement

Table 3 The group means of prior knowledge

Groups	N	Mean	SD
WM (Traditional WOE with metacognitive scaffolding)	27	5.59	2.24
FWM (Fade WOE with metacognitive scaffolding)	30	4.83	2.72
WFM (Traditional WOE with faded metacognitive scaffolding)	27	5.56	2.01

Table 4 Sample questions for cognitive load measures

Type of load	Questions
Intrinsic load	The content of this learning task was very complex
Extraneous load	The instructions and explanations during the activity were very unclear
Germane load	The activity really enhanced my understanding of the topic(s) covered

multiple-choice items: three items for intrinsic load, three for extraneous load, and four for germane load. Cognitive load was measured after the pre-training and training phases (Cronbach's Alpha = 0.87 and 0.82, respectively).

Measurement of level of self-efficacy

Self-efficacy was measured using a subscale of the Motivated Strategies for Learning Questionnaire (MSLQ) (Pintrich et al., 1991). The questionnaires consisted of eight multiple-choice questions (e.g., "I'm confident I can learn the basic concepts taught in this course.") (see Table 5). Self-efficacy was measured after the pre-training and training phases (Cronbach's Alpha = 0.79 and 0.85, respectively).

Measurement of level of programming problem-solving

Programming problem-solving tests were conducted to measure the participants' programming-related knowledge after the pre-training and training phases. Each test comprised a contextual and intricate programming problem-solving task. Participants were asked to create a grade calculator using Python after the pre-training phase and then create a course registration program after the training phase. They were given 50 min for each problem. Both problems were semi-open structured, with certain programming concepts that had to be included, but were presented as real-life-based problem-solving scenarios. The difficulty level of the second task was higher than the first. Programming problem-solving tests were assessed by two experts with more than five years of experience in teaching and researching computer programming. The evaluation comprised a set of two questions, with each question being worth 10 points, resulting in a total score of 20

Table 5 Sample self-efficacy items to measure prior knowledge

Items#	Problems
1	I'm certain I can master the skills being taught in this class
2	I'm confident I can learn the basic concepts taught in this course

points. The total grade points of the course are 100 points, and the problem-solving test score measured in the experiment accounts for 20 points out of the total 100 points (attendance 10 points, problem-solving test 20 points, midterm exam 30 points, final exam 40 points).

Procedure

The participants were divided into three groups, with each group receiving a different type of pre-training to perform their tasks. The experiment was conducted once a week for three consecutive weeks during the three-hour class meetings, so that the procedures took a total of nine hours. The participants had spent one week learning basic programming concepts (e.g., variables, loops, conditionals, and functions) and then took a pre-test. The instructor delivered a comprehensive briefing to the participants regarding the study's objectives, methodologies, advantages, data utilization, and the rights bestowed upon them as research participants. Eligibility for participation in this study was contingent upon individuals' voluntary consent subsequent to receive a comprehensive briefing on the study's nature and objectives. All 84 learners in the class agreed to participate in the study.

After the pre-test, the participants were asked to individually perform a programming problem-solving task as part of their assigned type of pre-training (WM, FWM, and WFM). After completing the pre-training, each participant took initial measurements of cognitive load, self-efficacy, and programming problem-solving. During the training phase, the participants were asked to carry out a programming problem-solving task using the faded WOE and faded metacognitive scaffolding. After completing the training, all of the participants took final measurements of cognitive load, self-efficacy, and programming problem-solving (see Fig. 2).

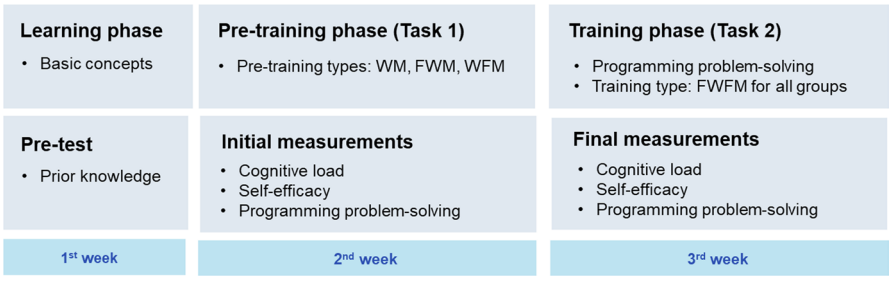

Fig. 2 Experiment Design. *Note* WM = traditional WOE with metacognitive scaffolding. FWM = faded WOE with metacognitive scaffolding. WFM = WOE with faded metacognitive scaffolding. WFWM = faded WOE with faded metacognitive scaffolding

Data collection and analysis

The independent variables were the three different types of pre-training, and the dependent variables were cognitive load, self-efficacy, and programming problem-solving. While there was no significant difference in prior knowledge among the three groups, it was considered as a covariate due to its potential correlation with problem-solving skills. The analysis was conducted using PSAW Statistics 21, with the significance level set at 0.05 and 0.01 for statistical verification.

Results

Cognitive load

The descriptive statistics of the participants' cognitive load levels are shown in Table 6. A two-way repeated measures ANOVA was conducted using the type of task in the pre-training and training as an intra-subject factor and the scaffolding type as an inter-subject factor.

Intrinsic load. There was a statistically significant difference in intrinsic cognitive load among the three groups in the pre-training and training, though the magnitude of this difference was not substantial, $F(2, 81) = 7.851$, $p < 0.1$. The effect size was $\eta^2 = 0.058$, which is on the upper end of what is typically considered a small effect, approaching medium (Cohen, 1988). An increase in intrinsic cognitive load was observed in the training in comparison to in the pre-training for both the FWM and WFM groups, while the WM group conversely exhibited a reduction in intrinsic cognitive load in the training. Post-hoc analysis revealed a significant difference only between FWM and WFM ($p < 0.05$).

Extraneous load. There was no statistically significant difference in extraneous load among the three groups in the pre-training and training, $F(2, 81) = 1.315$, $p = 0.53$. The effect size was $\eta^2 = 0.016$, which is considered a small effect, indicating minimal differences among the groups.

Table 6 The descriptive statistics of cognitive load

Cognitive load type		WM (n = 27)		FWM (n = 30)		WFM (n = 27)	
		Pre-training	Training	Pre-training	Training	Pre-training	Training
Intrinsic Load	M	4.25	3.65	2.98	3.38	4.32	4.56
	SD	1.79	2.10	2.06	2.34	1.97	2.18
Extraneous Load	M	3.00	2.80	2.32	2.50	3.10	3.28
	SD	1.26	1.56	1.48	1.53	1.91	1.94
Germane Load	M	7.29	7.98	7.51	7.22	7.25	7.27
	SD	1.46	1.42	2.13	1.87	1.32	1.41

Pre-training: Task 1, Training: Task 2

Germane load. There was a statistically significant difference in germane load among the three groups in the pre-training and training, F (2, 81) = 3.567, $p < 0.05$. The effect size was $\eta^2 = 0.074$, which is on the upper end of what is typically considered a small effect, approaching medium (Cohen, 1988), suggesting meaningful differences in how the groups processed the learning materials. Both the WM and WFM groups exhibited an increase in germane load in the training as compared to in the pre-training. However, no statistically meaningful difference was observed in the post-hoc analysis between the WM and WFM groups.

Self-efficacy

The influence of the type of task in the pre-training and training and scaffolding on self-efficacy was examined through a two-way repeated measures ANOVA. Statistically significant differences were observed among the three groups in terms of student self-efficacy, contingent upon the type of task in the pre-training and training, F (2, 81) = 4.923, $p < 0.01$. The effect size was $\eta^2 = 0.10$, which is considered a small effect, approaching medium (Cohen, 1988), suggesting a substantial impact of the type of task on student self-efficacy. In both the WM and FWM groups, self-efficacy was observed to be higher in the training as compared to in the pre-training. However, no statistically meaningful difference was identified in the post-hoc analysis between the WM and FWM groups. In the WFM group, despite having higher self-efficacy than the other two groups in the pre-training, the WFM group exhibited lower self-efficacy in the training (see Table 7).

Problem-solving skills

This study investigated the impact of scaffolding types on problem-solving ability, with the problem-solving tests being differentiated into two levels of difficulty: low and high. Considering pre-test scores, the results of the one-way ANCOVA for each test are as follows (see Table 8).

For the low-difficulty condition, a test measured after pre-training revealed no significant difference among the three groups, F (2, 80) = 0.063, $p = 0.94$. The effect size was $\eta^2 = 0.002$, indicating a very small effect, suggesting that the type of scaffolding had negligible impact on problem-solving ability in the low-difficulty tasks (see Table 9). On the other hand, for the high-difficulty condition, a test measured after training revealed significant differences among the three

Table 7 The Descriptive Statistics of Self-efficacy

Self-efficacy	WM (n = 27)		FWM (n = 30)		WFM (n = 27)	
	Pre-training	Training	Pre-training	Training	Pre-training	Training
M	4.58	5.04	4.85	5.13	5.17	4.81
SD	1.08	1.00	1.33	1.26	1.05	1.19

Pre-training: Task 1, Training: Task 2

Table 8 The descriptive statistics of problem-solving skills

Problem-Solving Skills	WM (n = 27)		FWM (n = 30)		WFM (n = 27)	
	Pre-training	Training	Pre-training	Training	Pre-training	Training
M	8.89	6.93	9.17	5.25	8.89	4.46
SD	2.89	3.57	1.90	3.95	3.20	3.75
AM	8.91 (SE: 0.52)	7.02 (SE: 0.72)	9.13 (SE: 0.50)	5.09 (SE: 0.52)	8.91 (SE: 0.52)	4.54 (SE: 0.72)

AM: adjusted mean, Pre-training: Task 1, Training: Task 2

Table 9 Tests of between-subjects effects dependent variable: pre-training

	Sum of square	*df*	Mean of square	*F*	*p*	η^2
Corrected model	4.171	3	1.390	0.190	.90	.007
Intercept	1175.019	1	1175.019	160.737	.00	.668
Pre-test	2.683	1	2.683	0.367	.55	.005
Group	0.917	2	0.458	0.063	.94	.002
Error	584.817	80	7.310			
Total	7375.000	84				
Corrected Total	588.988	83				

*$p < .05$. **$p < .01$

Table 10 Tests of between-subjects effects dependent variable: training

	Sum of square	*df*	Mean of square	*F*	*p*	η^2
Corrected model	134.126	3	44.709	3.249	.03	.109
Intercept	708.731	1	708.731	51.508	.00	.392
Pre-test	48.423	1	48.423	3.519	.06	.042
Group	91.556	2	45.778	3.327	.04*	.077
Error	1100.767	80	13.760			
Total	3809.000	84				
Corrected Total	1234.893	83				

*$p < .05$. **$p < .01$

groups, F (2, 80) = 3.327, $p < 0.05$. The effect size was $\eta^2 = 0.077$, which is on the upper end of what is typically considered a small effect, approaching medium (Cohen, 1988), indicating that the type of scaffolding had a more pronounced effect on the problem-solving ability in high-difficulty tasks (see Table 10) with the WM group scoring the highest (AM = 7.02, SE = 0.72). Furthermore, post-hoc analysis revealed a significant difference only between the WM and WFM groups ($p < 0.05$).

Discussion

The effects of three types of pre-training on cognitive load

In this study, we provided cognitive scaffolding and metacognitive scaffolding together, considering the importance of self-regulation skills, including cognitive and metacognitive skills, to address complex tasks in programming education (Loksa et al., 2016; Shin et al., 2023; Zimmerman & Schunk, 2011). Specifically, our aim was to identify the most effective type of pre-training when providing cognitive scaffolding and metacognitive scaffolding together, with the purpose of optimizing cognitive load and promoting learning. During the training phase, it was observed that the FWM type was the most effective in reducing intrinsic cognitive load and extraneous cognitive load and WFM type was the least effective. Conversely, the WM type was found to be the second-most effective in reducing intrinsic and extraneous cognitive load, following the FWM type, and it was the most effective in promoting germane cognitive load. This indicates that the WM type not only effectively minimized unnecessary cognitive load, securing space for germane cognitive load, but also promoted germane cognitive load most effectively. Based on findings, Hypothesis 1 was partially supported. Specifically, Hypothesis 1a, 1b, and 1c were each partially supported. In addition, Hypothesis 2 was also partially supported. Hypothesis 1a and 2b were supported, but Hypothesis 2c was unsupported. Faded WOE in a manner that does not match learners' levels of prior knowledge can lead to cognitive overload (Hancock-Niemi et al., 2016; Salleh et al., 2018). However, this study reveals that, for novice learners, the most effective pre-training approach in reducing unnecessary cognitive load involves providing faded WOE with core concepts eliminated, along with metacognitive scaffolding, as was the case with the FWM pre-training type.

Cognitive load occurs due to the limited capacity of human working memory to process a certain amount of information within a given time (Mayer & Moreno, 2003). Previous studies have aimed to reduce extraneous cognitive load and promote germane cognitive load, which is beneficial for learning (Renkl & Atkinson, 2003). In particular, pre-training can effectively reduce unnecessary cognitive load during the problem-solving process of complex tasks by providing learners with prior exposure to the core content. In this study, the FWM type exhibited the lowest intrinsic and extraneous cognitive load and the highest germane load during the pre-training phase. During the training phase, it showed the lowest intrinsic and extraneous cognitive load and the second-highest germane load. This finding demonstrates that the FWM type serves as an effective pre-training approach, not only in reducing unnecessary cognitive load but also in facilitating germane cognitive load, which benefits learning. On the other hand, the WM type showed lower intrinsic cognitive load and extraneous cognitive load during the training phase compared to the pre-training phase, while germane cognitive load increased. Considering that the difficulty of the learning tasks is higher during the training phase compared to the pre-training phase, it is predicted that the WM type could be applied as an effective pre-training approach for learning tasks with high difficulty.

The effects of three types of pre-training on self-efficacy

Self-efficacy is the degree of confidence that learners have in their abilities and effectiveness (Bandura, 1977). Those with higher self-efficacy take a challenging approach to tasks that are considered somewhat difficult and think that they can control and handle them (Schunk & Pajares, 2002). Therefore, self-efficacy is an important factor influencing learning. In training, self-efficacy was highest in the FWM group with fading applied to cognitive scaffolding and lowest in the WFM group. This suggests that providing the WM or FWM types in pre-training is effective in promoting self-efficacy in novice learners. Based on findings, Hypothesis 3 was partially supported. The WM type is more effective in enhancing self-efficacy than the WFM type. In addition, Hypothesis 4 was supported. Considering previous studies on self-efficacy, which showed that learners with higher self-regulatory ability had higher self-efficacy (Deweck & Leggett, 1988), the research results imply that the FWM pre-training strategy, which applies fading to cognitive scaffolding, is the most effective strategy for improving self-regulation skill.

When we examined changes in self-efficacy during the pre-training and training stages, self-efficacy increased in all groups except the WFM group. In the case of the WFM group, which provided fading to the expert's metacognitive strategy, self-efficacy appeared to be high in pre-training with low task difficulty, but it was confirmed that self-efficacy was the lowest in the training stage with high task difficulty. This indicates that providing expert metacognitive strategies without fading to pre-training is effective in improving self-efficacy for novice learners who lack self-regulation ability. In other words, it can be suggested that providing both cognitive and metacognitive scaffolding together, which includes all the metacognitive strategies of experts required for each stage of problem-solving, is effective in promoting self-efficacy.

The effects of three types of pre-training on problem-solving

Because prior research has shown that novice learners provided with metacognitive scaffolding exhibited better programming performance than learners not provided with metacognitive scaffolding (Mohd Rum & Ismail, 2017), we explored pre-training types that are effective for problem-solving by providing WOE and metacognitive scaffolding together in training, the WM type was most effective in problem-solving, and the FWM type was second-most effective in problem-solving. Based on findings, Hypothesis 5 was unsupported, and Hypothesis 6 was supported. Although WM was most effective in problem-solving, FWM was more effective than WFM in problem-solving. Given that the level of prior knowledge in related areas affects problem-solving skills (Sweller, 1988), it can be predicted that the FWM type, which was the most effective in problem-solving in pre-training, had a positive effect on problem-solving in training by facilitating learners' prior knowledge. On the other hand, the WM group showed lower problem-solving skills than the FWM group in pre-training, but the highest problem-solving skills in training. This shows

that providing WM-type pre-training without using a faded strategy to novice learners is effective in solving difficult programming problems. On the other hand, comparing the problem-solving scores of pre-training and training, the scores of the pre-training tasks were higher than those of the training tasks. It can be inferred that this is because the difficulty level of the pre-training tasks is relatively low compared to the difficulty level of the training tasks.

In general, having a high level of prior knowledge indicates having numerous schemas for learning content in related domains, and prior knowledge is widely recognized as a crucial element in enhancing problem-solving skills (Sweller, 1988). In this study, we were able to confirm that providing cognitive scaffolding along with metacognitive scaffolding in programming problem-solving for novice learners is an effective pre-training approach that facilitates schema acquisition and enhance problem-solving skills. Metacognitive scaffolding plays an important role in the problem-solving process by helping learners check what and how much they know and refer to expert metacognitive strategies to select appropriate problem-solving strategies (Shin & Song, 2022). In this study, we aimed to utilize a faded metacognitive scaffolding that partially erased the core concept within the expert's metacognition strategy to help promote learners' metacognition. However, the study provided empirical evidence that providing non-faded metacognitive scaffolding together with cognitive scaffolding such as WOE or faded WOE was the most effective approach for problem-solving in novice learners.

On the other hand, the FWM type, which had the lowest intrinsic and extraneous cognitive load and the highest germane cognitive load in pre-training, was the most effective in solving pre-training problems. In addition, the WM type with the second-lowest intrinsic and extraneous cognitive load and the highest germane cognitive load in training was most effective in problem solving in training. These findings support previous research (Sweller et al., 1998) that optimizing the cognitive load during the problem-solving process enhances problem-solving skill and results in successful problem-solving learning. In addition, the findings suggest that when providing a scaffolding strategy to improve learning performance, instructional should be designed considering the optimization of cognitive load. On the other hand, compared to pre-training, the WM and FWM groups, whose self-efficacy improved in the training stage, also exhibited effective problem-solving in the training stage, implying that self-efficacy is one of the important factors affecting learning.

Conclusion

In this study, we tried to derive an effective pre-training type to support novice learners' problem-solving in programming. In particular, we explored which types of scaffolding provision caused synergistic effects when providing cognitive and metacognitive scaffolding together. As a result, although the effect size was statistically small effect, approaching medium (Cohen, 1988), it was founded that it is effective to provide both cognitive scaffolding such as WOE or faded WOE and non-faded metacognitive scaffolding to novice learners. In other words, providing all the

core contents of the expert's metacognitive strategy has a positive effect on novice learners' cognitive load control, self-efficacy improvement, and problem-solving in programming. For cognitive scaffolding, providing WOE with fading was effective in reducing unnecessary cognitive load, improving self-efficacy, and solving low-difficulty problems. WOEs without fading strategies were effective in facilitating germane cognitive load during the training phase and performing intermediate-level learning tasks. So far, there have been many studies that have applied fading to cognitive scaffolding, but it was difficult to find an attempt to apply fading to metacognitive scaffolding. In this study, it is meaningful that we not only explored the effectiveness of faded metacognitive scaffolding, but also obtained empirical results on scaffolding types that can enhance the effect when cognitive scaffolding and metacognitive scaffolding are provided together in pre-training. In addition, while pre-training has been applied mainly in the fields of multimedia learning (Mayer et al., 2002) and computer-supported collaborative learning (Jung et al., 2021), the present study presents an empirical case of applying pre-training to programming problem-solving for novice learners.

Despite these contributions, the present study has several limitations that must be acknowledged. First, in this study, a strategy to erase core concepts was used when developing faded WOE. Further studies should explore effective faded WOE types in depth utilizing more diverse elimination strategies. Second, in this study, a simplified form of WOE was utilized. However, depending on the task content, WOE may be presented in more complex forms than those designed in this study. Therefore, the results of this study conducted using these simplified forms of WOE may not generalize to more complex WOE. Third, this study provided learners with the problem-solving strategies of experts through cognitive and metacognitive scaffolding. However, in order to effectively facilitate problem-solving, it is essential to further enhance and elaborate the integration between cognitive scaffolding and metacognitive scaffolding. Future studies should focus on seamlessly incorporating the contents of cognitive scaffolding and metacognitive scaffolding in the scaffolding design process, considering the various stages of problem-solving. Despite these limitations, the present paper goes some way in designing an effective cognitive scaffolding and metacognitive scaffolding strategy for novice learners to support programming problem-solving.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval All respondents of collected surveys were informed of their rights and participated voluntarily.

References

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review, 84*(2), 191. https://doi.org/10.1037/0033-295X.84.2.191

Berk, L. (2003). Child development. Pearson Higher Education AU. Retrieved from https://www.pearson.com/en-au/media/yfcpe1ax/9780205149766.pdf

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202–215. https://doi.org/10.1016/j.compedu.2016.01.010

Chen, C. H., Liu, T. K., & Huang, K. (2023). Scaffolding vocational high school students' computational thinking with cognitive and metacognitive prompts in learning about programmable logic controllers. *Journal of Research on Technology in Education, 55*(3), 527–544.

Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*(2), 145–182. https://doi.org/10.1016/0364-0213(89)90002-5

Clarke, T., Ayres, P., & Sweller, J. (2005). The Impact of sequencing and prior knowledge on learning mathematics through spreadsheet applications. *Educational Technology Research and Development, 53*(3), 15–24. https://doi.org/10.1007/BF02504794

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Lawrence Erlbaum Associates.

de Jong, T. (2010). Cognitive load theory, educational research, and instructional design: Some food for thought. *Instructional Science, 38*(2), 105–134. https://doi.org/10.1007/s11251-009-9110-0

Dweck, C. S., & Leggett, E. L. (1988). A social-cognitive approach to motivation and personality. *Psychological Review, 95*(2), 256. https://doi.org/10.1037/0033-295X.95.2.256

Garner, S. (2001). A tool to support the use of part-complete solutions in the learning of programming. In *Proceedings of the 2001 Informing Science Conference* (pp. 222–228). https://doi.org/10.28945/2385

Garner, S. (2002). Reducing the Cognitive Load on Novice Programmers. In P. Barker & S. Rebelsky (Eds.), *Proceedings of ED-MEDIA 2002--World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 578–583). Denver, Colorado, USA: Association for the Advancement of Computing in Education (AACE). Retrieved October 6, 2022 from https://www.learntechlib.org/primary/p/10329/

Hancock-Niemic, M. A., Lin, L., Atkinson, R. K., Renkl, A., & Wittwer, J. (2016). Example-based learning: Exploring the use of matrices and problem variability. *Educational Technology Research and Development, 64*(1), 115–136. https://doi.org/10.1007/s11423-015-9403-8

Hwang, Y. S., & Vrongistinos, K. (2002). Elementary In-Service Teachers' Self-Regulated Learning Strategies Related to Their Academic Achievements. *Journal of Instructional Psychology, 29*(3). Retrieved from https://www.proquest.com/scholarly-journals/elementary-service-teachers-self-regulated/docview/1416365195/se-2

Jung, J., Shin, Y., & Zumbach, J. (2021). The effects of pre-training types on cognitive load, collaborative knowledge construction and deep learning in a computer-supported collaborative learning environment. *Interactive Learning Environments, 29*(7), 1163–1175. https://doi.org/10.1080/10494820.2019.1619592

Jung, J., Kim, D., & Na, C. (2016). Effects of WOE presentation types used in pre-training on the cognitive load and comprehension of content in animation-based learning environments. *Journal of Educational Technology & Society, 19*(4), 75–86.

Kalyuga, S., Chandler, P., Tuovinen, J., & Sweller, J. (2001). When problem solving is superior to studying worked examples. *Journal of Educational Psychology, 93*(3), 579–588. https://doi.org/10.1037/0022-0663.93.3.579

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: Novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin, 38*(3), 118–122. https://doi.org/10.1145/1140124.1140157

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). https://doi.org/10.1145/2858036.2858252

Loksa, D., Xie, B., Kwik, H., & Ko, A. J. (2020, February). Investigating novices' in situ reflections on their programming process. In *Proceedings of the 51st ACM Technical. Symposium on Computer Science Education* (pp. 149–155). https://doi.org/10.1145/3328778.3366846

Magana, A. J., Fennell, H. W., Vieira, C., & Falk, M. L. (2019). Characterizing the interplay of cognitive and metacognitive knowledge in computational modeling and simulation practices. *Journal of Engineering Education, 108*(2), 276–303.

Mayer, R. E., & Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. *Educational Psychologist, 38*(1), 43–52. https://doi.org/10.1207/S15326985EP3801_6

Mayer, R. E., Mathias, A., & Wetzell, K. (2002). Fostering understanding of multimedia messages through pre-training: Evidence for a two-stage theory of mental model construction. *Journal of Experimental Psychology: Applied, 8*(3), 147–154. https://doi.org/10.1037/1076-898X.8.3.147

van Merriënboer, J. J., & de Croock, M. B. (1992). Strategies for computer-based programming instruction: Program completion vs. program generation. *Journal of Educational Computing Research, 8*(3), 365–394.

van Merriënboer, J. J., & Kirschner, P. (2012). *Ten steps to complex learning: A systematic approach to four-component instructional design* (2nd ed.). New York: Routledge/Taylor & Francis Group. https://doi.org/10.4324/9781315113210

Mohd Rum, S. N., & Ismail, M. A. (2017). Metacognitive support accelerates computer assisted learning for novice programmers. *Journal of Educational Technology & Society, 20*(3), 170–181.

Molenaar, I., van Boxtel, C. A., & Sleegers, P. J. (2011). Metacognitive scaffolding in an innovative learning arrangement. *Instructional Science, 39*(6), 785–803. https://doi.org/10.1007/s11251-010-9154-1

Paas, F., & Sweller, J. (2012). An Evolutionary upgrade of cognitive load theory: Using the human motor system and collaboration to support the learning of complex cognitive tasks. *Educational Psychology Review, 24*(1), 27–45. https://doi.org/10.1007/s10648-011-9179-2

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). https://eric.ed.gov/?id=ED338122

Prather, J., Becker, B. A., Craig, M., Denny, P., Loksa, D., & Margulieux, L. (2020, August). What do we think we think we are doing? Metacognition and self-regulation in programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (pp. 2–13). https://doi.org/10.1145/3487050

Renkl, A. (2002). Worked-out examples: Instructional explanations support learning by self-explanations. *Learning and Instruction, 12*(5), 529–556. https://doi.org/10.1016/S0959-4752(01)00030-5

Renkl, A., & Atkinson, R. K. (2003). Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. *Educational Psychologist, 38*(1), 15–22. https://doi.org/10.1207/S15326985EP3801_3

Renkl, A., Atkinson, R. K., Maier, U. H., & Staley, R. (2002). From example study to problem solving: Smooth transitions help learning. *The Journal of Experimental Education, 70*(4), 293–315. https://doi.org/10.1080/00220970209599510

Renkl, A., Atkinson, R. K., & Maier, U. H. (2000, August). From studying examples to solving problems: Fading worked-out solution steps helps learning. In L. Gleitman & A. K. Joshi (Eds.). In *Proceeding of the 22nd Annual Conference of the Cognitive Science Society* (pp. 393–398). Mahwah, NJ: Lawrence Erlbaum Associates, Inc. https://doi.org/10.1037/0022-0663.95.4.774

Salleh, S. M., Shukur, Z., & Judi, H. M. (2018). Scaffolding model for efficient programming learning based on cognitive load theory. *International Journal of Pure and Applied Mathematics, 118*(7), 77–83.

Schunk, D. H., & Pajares, F. (2002). The development of academic self-efficacy. In *Development of achievement motivation* (pp. 15–31). Academic Press.

Seta, K., Satake, H., Umano, M., & Ikeda, M. (2007, September). Learning phase model based scaffolding and its fading to facilitate collaborative learning of critical thinking. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems* (pp. 590–599). Berlin, Heidelberg: Springer

Shell, D. F., Colvin, C., & Bruning, R. H. (1995). Self-efficacy, attribution, and outcome expectancy mechanisms in reading and writing achievement: Grade-level and achievement-level differences. *Journal of Educational Psychology, 87*(3), 386. https://doi.org/10.1037/0022-0663.87.3.386

Shin, Y., & Song, D. (2022). The effects of self-regulated learning support on learners'. Task. performance and cognitive load in computer programing. *Journal of Educational Computing Research, 60*(6), 1490–1513.

Shin, Y., Jung, J., Zumbach, J., & Yi, E. (2023). The Effects of worked-out example and metacognitive scaffolding on problem-solving programming. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331231174454

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review, 22*(2), 123–138. https://doi.org/10.1007/s10648-010-9128-5

Sweller, J., van Merriënboer, J. J., & Paas, F. G. (1998). Cognitive architecture and instructional design. *Educational Psychology Review, 10*(3), 251–296. https://doi.org/10.1023/A:1022193728205

van de Pol, J., Volman, M., & Beishuizen, J. (2010). Scaffolding in teacher–student interaction: A decade of research. *Educational Psychology Review, 22*, 271–296. https://doi.org/10.1007/s10648-010-9127-6

Wu, L., Looi, C. K., Multisilta, J., How, M. L., Choi, H., Hsu, T. C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: A comparative study between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific Education Researcher, 29*(1), 21–34. https://doi.org/10.1007/s40299-019-00485-x

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education, 29*(2–3), 205–253.

Zheng, L., Zhen, Y., Niu, J., & Zhong, L. (2022). An exploratory study on fade-in versus fade-out scaffolding for novice programmers in online collaborative programming settings. *Journal of Computing in Higher Education, 34*(2), 489–516. https://doi.org/10.1007/s12528-021-09307-w

Zimmerman, B. J., & Martinez-Pons, M. (1990). Student differences in self-regulated learning: Relating grade, sex, and giftedness to self-efficacy and strategy use. *Journal of Educational Psychology, 82*(1), 51. https://doi.org/10.1037/0022-0663.82.1.51

Zimmerman, B., & Schunk, D. H. (2011). *Handbook of self-regulation of learning and performance*. Taylor & Francis. https://doi.org/10.4324/9780203839010

Jaewon Jung is a research fellow of Office of Higher Education Research at Korean Educational Development Institute, South Korea. She holds a Ph.D. in Educational Technology from Hanyang University, South Korea. Her current research interests include exploring the educational issues such as multimedia learning, computer-supported collaborative learning (CSCL), problem-based learning (PBL), and instructional design to provide effective learning strategies and enhance learning outcomes.

Yoonhee Shin is a professor in the Department of Educational Technology at Hanyang University. Professor Shin's research focus is on instructional design, learning analytics using multimodal data, computer-supported collaborative learning (CSCL), and software education including AI for everyone. Her work aims to enhance problem-solving skills, promote meaningful peer interaction, and foster computational thinking.

Haejin Chung is a professor in the Cha Mirisa College at Duksung women's university, South Korea. She holds a Ph.D. in Computer Science and Enginnering from Dankook University, South Korea. Her research interests include big data, problem-based learning (PBL) and software education.

Mik Fanguy is an invited professor in the School of Digital Humanities and Computational Social Sciences at the Korea Advanced Institute of Science and Technology (KAIST) in South Korea. His research interests include online collaborative writing and note-taking.

Authors and Affiliations

Jaewon Jung[1] · Yoonhee Shin[2] · HaeJin Chung[3] · Mik Fanguy[4]

✉ Yoonhee Shin
yoonheeshin@hanyang.ac.kr

Jaewon Jung
jjungj5@gmail.com

HaeJin Chung
hjchung@duksung.ac.kr

Mik Fanguy
mik@kaist.edu

[1] Korean Educational Development Institute (KEDI), Jincheon-gun, Chungcheongbuk-Do, Republic of Korea

[2] Hanyang University, Seoul, Republic of Korea

[3] Duksung Women's University, Seoul, Republic of Korea

[4] Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea