IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.006, LIIR439
DOI: https://doi.org/10.52731/liir.v006.439

Exploring the Effects of a Scaffolded Programming Learning Environment Integrated with ChatGPT on Students' Learning Performance and Self-Efficacy

Chia-Jung Chang [*], Li-Heng Hung [*]

Abstract

This study designed an online scaffolded learning programming environment, integrating self-regulated learning model with ChatGPT, to facilitate students learning programming and explored its impact on learning performance, cognitive load, and self-efficacy. A one-group pretest-posttest experimental design, the experiment involved 58 university students who took part in three weeks learning activity. The results showed that the environment benefitted students on the enhancement of conceptual understanding of web programming, the enhancement of learning self-efficacy on logical thinking, algorithm, and debugging. Moreover, it increased students' germane cognitive load and reduced their perceived task difficulty.

Keywords: learning programming, metacognition, learning self-efficacy, ChatGPT

1 Introduction

Programming competencies is regarded as one of the significant competencies of the 21st century and is often viewed as a tool to help individuals solve complex problems [1]. Hence, how to cultivate students' programming competencies has long been an important issue of concern among scholars and educators in the field of computer science education worldwide. For most learners, programming remains a challenging subject, especially for novices [2]. Programming concepts are often too abstract to grasp [3], and repeated failures during debugging can discourage learners to keep learning [4]. Therefore, it is necessary to provide additional scaffolding or supportive tools in instructional design to help students overcome learning difficulties.

In recent years, the emergence of Generative Artificial Intelligence (GenAI) tools has provided users with various solutions to complex problems. Among these tools, ChatGPT has become particularly well-known for its ability to interact with users through natural language conversation, offering answers and suggestions in response to their queries. Compared to traditional conversational chatbots—which mainly process user input and generate fixed responses—ChatGPT is considered a tool for augmented intelligence [1]. The concept emphasizes that learners can work more efficiently and effectively with the aid of AI, compensating for gaps in knowledge and decision-making skills. Consequently, numerous studies have explored the application of ChatGPT in diverse educational contexts, including engineering, physics, and English learning.

Research suggests that ChatGPT can serve as a powerful learning support tool by offering immediate feedback and guidance [1]. Unlike traditional programming learning environments,

[*] Department of Information Communication, Yuan Ze University, Taiwan

ChatGPT does not require users to have advanced programming skills or prior knowledge. This makes it especially valuable for learners with limited programming backgrounds, providing a more flexible and approachable way to solve problems [4]. Specifically, studies have shown that ChatGPT delivers timely feedback that helps students understand programming concepts more quickly and effectively [5]. Moreover, ChatGPT supports personalized learning by adapting to each learner's individual progress and challenges, much like a personal tutor. It can assist with querying syntax, providing code examples, explaining code, and debugging errors [5]. Given these empowering and multifaceted features, ChatGPT can be considered a scaffolded learning aid that helps students more easily reach their zone of proximal development.

However, some studies have pointed out potential limitations and hidden challenges associated with ChatGPT. These include the lack of a structured learning process, insufficient integration with programming applications or development environments, and limited support for data structures or algorithms [1]. Rahman and Watanobe [6] also highlighted issues such as a lack of common sense, absence of visual representations, and difficulties in handling complex reasoning tasks. Other studies have reported negative impacts of GenAI tools on students' academic performance, abilities, and metacognitive engagement. For example, Niloy et al. [7] found that students' writing abilities declined with the use of ChatGPT, while Song and Song [8] observed that although ChatGPT enhanced students' English writing skills, it also led to over-reliance. Similarly, Fan et al. [9] reported that ChatGPT induced metacognitive laziness, as students failed to monitor and reflect on their learning processes. In other words, while GenAI tools help lighten the cognitive load, their prolonged use may shape learners' habits in ways that discourage cognitive effort.

In educational research, metacognition is widely recognized as a key factor in successful learning and the core of self-regulated learning. However, excessive use of AI in the learning process may reduce students' metacognitive engagement, leading to lower autonomy and a weaker ability to learn independently. How to leverage ChatGPT to support deep learning and minimize the problem of metacognitive laziness is a critical issue worthy of further exploration. To address this, the study developed an online scaffolded programming learning environment that integrates metacognitive questions and ChatGPT to support students in learning programming, and explored its impact on their learning performance, cognitive load, and self-efficacy. Specifically, the research questions are as follows:

- Does the scaffolded programming learning environment facilitate students' conceptual understanding?
- Does the scaffolded programming learning environment improve students' programming self-efficacy?
- What is the impact of the scaffolded programming learning environment on students' cognitive load?

2 Method

2.1 Participants

This study adopted a one-group pretest-posttest experimental design to investigate the effect of the scaffolded programming learning environment on students' programming performance, cognitive load, and perceptions of self-efficacy in learning programming. A total of 58 first-year students from a private university in northern Taiwan participated in the study. They had learned basic web programming knowledge but had no prior experience using AI tools to assist in programming learning. Therefore, they were suitable as research samples.

2.2 Learning Programming Activity

This study developed an online programming learning environment that integrated the self-regulated learning model with ChatGPT. The learning activity was based on the four phases of the self-regulated learning model - experience, planning, execution, and reflection - to guide students in building a to-do list web application. The task was divided into three subtasks: layout design, data access, and CRUD functionality, all of which were related to system structure and algorithms. In the experience phase, students explored and interacted with example to understand the objectives of each subtask. During the planning phase, they needed to lay out learning strategies. In the execution phase, students implemented these strategies to write code using an online code editor developed by the study. It allowed them to run their code immediately and monitor their progress. In the reflection phase, students were prompted to consider what they had learned, identify any difficulties encountered, and consider how they could improve their work. ChatGPT was embedded within the environment, allowing students to interact with it throughout the learning process. To promote deeper learning, metacognitive questions were incorporated into the execution and reflection phases. For instance, prompts like "How do you plan to complete this task?" and "If you want to add a new to-do item, what steps are needed?" encouraged students to reflect critically instead of depending solely on ChatGPT for answers.

2.3 Procedure

Before the activity, students completed two questionnaires—the Programming Self-Efficacy Scale and the Cognitive Load Questionnaire—as well as a pretest, which took 50 minutes. The learning activity lasted for three weeks, during which students were required to complete one subtask each week outside of class. They were required to plan and manage their own learning plans and complete the assigned tasks within the learning environment. After each subtask, the same learning test was administered to assess their conceptual understanding of web programming. Additionally, the instructor provided whole-class feedback to reinforce correct concepts. After completing all three subtasks, the same two questionnaires were administered again as a posttest.

2.4 Data Collection and Analysis

Learning tests were developed by the researchers to assess students' understanding of web programming concepts including web layout design, data access, and CRUD operations units. The Computer Programming Self-Efficacy Scale developed by Tsai, Wang, and Hsu [2] was used to measure students' self-efficacy in both traditional and scaffolded learning environments. The scale includes five dimensions: logical thinking, algorithms, debugging, control, and collaboration. The collaboration dimension was excluded from this study since the learning activity was conducted individually. In addition, the Cognitive Load Scale developed by Leppink et al. [10] was used to measure students' mental effort. The questionnaire consists of ten items measuring three types of cognitive load: intrinsic, extraneous, and germane. The Cronbach's alpha for the scales was 0.94 and 0.92, indicating high reliability. Data from the scales were analyzed using paired t-tests. Furthermore, interviews were conducted to identify potential problems students encountered during the activity.

3 Result and Discussion

3.1 The Conceptual Understanding of Learning Programming

Table 1 shows the results of students' learning performance across three tests using a paired t-test. The results revealed significant differences in all three tests after the activity compared to conventional instruction. These findings suggest that the learning environment effectively enhanced students' conceptual understanding in three areas: web layout design, data access, and CRUD operations. However, it is worth noting that students' test scores still require further improvement.

Table 1: The result of students learning performance in the conventional and the environment

Task test	Setting	Mean	SD	t-value	p-value
Layout design	Conventional	30.2308	9.97774	-9.68***	0.00
	Scaffolded	55.7692	18.75481		
Data Access	Conventional	29.0517	10.81977	-9.55***	0.00
	Scaffolded	55.7759	20.6432		
CRUD	Conventional	35.8036	13.87531	-7.12***	0.00
	Scaffolded	55.3571	22.64061		

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

3.2 Students' Cognitive Load

Table 2 presents the results of students' cognitive load in the two learning environments using a paired t-test. The results showed a significant difference in effort (t = -2.26, p < .05). Additionally, there were marginally significant differences in germane cognitive load (t = -1.68, p = .09) and perceived task difficulty (t = -1.80, p = .07). These findings suggest that students exerted more effort in the scaffolded environment than in the conventional instruction. Furthermore, the scaffolded environment may have increased students' germane cognitive load and reduced their perceived task difficulty. In other words, students in the scaffolded environment invested more effort in learning programming and perceived the tasks as less difficult.

Table 2: The result of students' cognitive load in the two environments

	Setting	Mean	SD	t-value	p-value
Intrinsic load	Conventional	3.3626	0.74344	-0.34	0.73
	Scaffolded	3.4035	0.81342		
Extraneous load	Conventional	2.7076	0.75873	1.56	0.12
	Scaffolded	2.5088	0.90437		
Germane load	Conventional	3.5614	0.64828	-1.68	0.09
	Scaffolded	3.7281	0.65853		
Effort	Conventional	3.2982	0.77839	-2.26*	0.03
	Scaffolded	3.6491	0.85547		
Difficulties	Conventional	3.4035	0.72849	-1.80	0.07
	Scaffolded	3.614	0.83995		

*<.05

3.3 Learning Programming Self-efficacy

Table 3 presents the results of students' programming self-efficacy in the two environments using a paired t-test. Significant differences were found in logical thinking (t = -2.07, p < .05) and algorithmic thinking (t = -1.88, p = .06). Additionally, there was a marginally significant difference in debugging ability (t = -1.88, p = .06). However, no significant difference was observed in control competency between the two environments. These results indicate that the scaffolded environment enhanced students' programming self-efficacy, particularly in logical thinking and algorithmic skills. Moreover, the environment may also contribute to improvements in students' debugging abilities.

Table 3: The result of students' learning programming self-efficacy in the two environments

	Setting	Mean	SD	t-value	p-value
Logical thinking	Conventional	3.5647	0.98515	-2.07*	0.04
	Scaffolded	3.8664	1.03895		
Algorithm	Conventional	2.6782	1.20907	-2.38*	0.02
	Scaffolded	3.0172	1.1788		
Debugging	Conventional	3.3046	1.02652	-1.88	0.06
	Scaffolded	3.5575	1.07676		
Control	Conventional	3.6379	1.06748	-0.86	0.39
	Scaffolded	3.7874	1.14911		

*<.05

4 Conclusion

This study designed an online scaffolded learning programming environment to facilitate students' programming learning and explored its impact on learning performance, cognitive load, and self-efficacy. The results indicated that the environment not only enhanced students' conceptual understanding of web programming but also strengthened their logical thinking, algorithmic problem-solving, and debugging. Moreover, the environment facilitated students to voluntarily invest greater effort in learning programming and may have helped them use more cognitive resources to processing information.

However, it is worth noting that many students still experienced difficulties in understanding programming concepts. Future studies could develop pedagogical strategies to enhance students' conceptual understanding. The findings of this study can serve as a reference for researchers and instructors in GenAI-supported instruction.

Nonetheless, several limitations should be acknowledged. Due to the constraints of the experimental design, the results may not fully verify the effectiveness of the scaffolded environment. Additionally, the findings were based primarily on questionnaire data and learning test outcomes. Furthermore, the activity was conducted over a relatively short-term activity. Future research should consider conducting a longer-term intervention, as well as evaluating students' programming processes to better identify potential learning difficulties.

Acknowledgement

This research was partially funded by the National Science and Technology Council, Taiwan under contract numbers NSTC 113-2410-H-155-009- and MOE Teaching Practice Research Program (PEE1137615).

References

[1] R. Yilmaz and F. G. K. Yilmaz, "The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation," Computers and Education: Artificial Intelligence, vol. 100147, 2023.

[2] C. Y. Tsai, "Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy," Computers in Human Behavior, vol. 95, pp. 224–232, 2019.

[3] J. C. Y. Sun and K. Y. C. Hsu, "A smart eye-tracking feedback scaffolding approach to improving students' learning self-efficacy and performance in a C programming course," Computers in Human Behavior, vol. 95, pp. 66–72, 2019.

[4] N. M. S. Surameery and M. Y. Shakor, "Use ChatGPT to solve programming bugs," Int'l J. of Information Technology & Computer Engineering (IJITC), vol. 3, no. 1, pp. 17–22, 2023.

[5] E. Chen, R. Huang, H. S. Chen, Y. H. Tseng, and L. Y. Li, "GPTutor: A ChatGPT-powered programming tool for code explanation," arXiv preprint, arXiv:2305.01863, 2023.

[6] M. M. Rahman and Y. Watanobe, "ChatGPT for education and research: Opportunities, threats, and strategies," Applied Sciences, vol. 13, no. 9, p. 5783, 2023.

[7] A. C. Niloy, S. Akter, N. Sultana, J. Sultana, and S. I. U. Rahman, "Is ChatGPT a menace for creative writing ability? An experiment," J. of Computer Assisted Learning, vol. 40, no. 2, pp. 919–930, 2024.

[8] C. Song and Y. Song, "Enhancing academic writing skills and motivation: Assessing the efficacy of ChatGPT in AI-assisted language learning for EFL students," Front. Psychol., vol. 14, p. 1260843, 2023.

[9] Y. Fan et al., "Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance," Br. J. Educ. Technol., vol. 00, pp. 1–42, 2024.

[10] J. Leppink, F. Paas, C. P. van der Vleuten, T. van Gog, and J. J. van Merriënboer, "Development of an instrument for measuring different types of cognitive load," Behav. Res. Methods, vol. 45, pp. 1058–1072, 2013.