Jiang, B. et al. (Eds.) (2025). Proceedings of the 33rd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Learning Support System Visualizing Behavior of Student Program in Object Oriented Language Based on Teacher's Intent of Instruction

Koichi YAMASHITA[a*], Kenzo KOBAYASHI[b], Takuma KOKUBO[b], Satoru KOGURE[b], Yasuhiro NOGUCHI[b], Raiya YAMAMOTO[b], Tatsuhiro KONISHI[b] & Yukihiro ITOH[c]
[a]*Faculty of Business Administration, Tokoha University, Japan*
[b]*Faculty of Informatics, Shizuoka University, Japan*
[c]*Shizuoka University, Japan*
*yamasita@hm.tokoha-u.ac.jp

Abstract: This study describes a learning support system developed by extending a system that visualizes program behavior based on teachers' intent of instruction. The extended system supports functions to visualize concepts specific to object-oriented languages and the behavior of learners' program code. Our system was evaluated based on subjective evaluations by learners in practical classes where it was introduced and implemented, and test results from an evaluation experiment conducted with recruited participants. The evaluation results suggested that although the system's impact was not uniform across all learners, it proved highly effective for students with insufficient understanding of the programs.

Keywords: Programming education, program visualization system, educational authoring tool, classroom practice

1. Introduction

Thus far, we developed a program visualization (PV) system called the Teacher's Explaining Design Visualization Tool (TEDViT) for the C language and conducted several classroom practices by introducing it into actual classes (Kogure et al., 2014). A distinctive feature of TEDViT is that it allows teachers to freely customize PV according to their intent of instruction. The evaluation results from the classroom practice suggest that TEDViT effectively supports programming learning for novice learners. We also developed extensions to support the understanding of learning objectives specific to object-oriented languages to deal with the trends in languages used for novice programming education (Yamashita et al., 2022) and extensions to visualize the behaviors of learners' program codes to improve learner engagement (Kobayashi et al., 2024). These extensions were evaluated individually to obtain positive results regarding visualization validity and learning effects.

However, these studies were independent of each other. In this study, we developed an extended TEDViT that incorporates functions to visualize the behaviors of learners' program codes, supporting visualizations of concepts specific to object-oriented languages. This paper describes the extended TEDViT and its evaluation experiments. We evaluated student engagement through questionnaire surveys administered during practical classes that incorporated our system, while learning effectiveness was evaluated based on test results from an experiment conducted with recruited participants.

2. Proposed System

We developed a PV system by extending TEDViT, which supports the visualization of an object-oriented conceptual model (OOCM) (Kogure et al., 2019) for *modified content*. OOCM is a visualization model based on the Unified Modeling Language (UML) that can visualize the relationships among objects and classes, including the concepts of inheritance and polymorphism. *Modified content* is one of the levels of ownership of target programs in PV systems categorized by Sorva et al. (2013), including *given content*, where learners cannot modify the target program at all; *own cases*, where learners can only modify input and variable values; *modified content*, where learners can edit the target program; and *own content*, where learners can observe their program code as the target program. Sorva et al. (2013) pointed out that the closer the content ownership of the target program is to *own content*, the stronger the learner's engagement, and the better the learning effectiveness of the system.

Bringing content ownership closer to *one's own content* is an inherently difficult approach in a PV system that cannot reflect a teacher's intent of instruction. Therefore, no similar attempts have been made to date. The reason for limiting the correspondence to *modified content* was to simplify the process of determining the correspondence between the teacher's intent of instruction and the elements of the learner's program. Although this reduces the coding flexibility of the target program, we believe that this constraint will not be a major problem when introducing the system in programming education for novice learners.

TEDViT displays PV and interaction on a web browser, considering the ease of integration into various classroom environments. This implies that the coding environment for learners and the PV observation environment were provided separately. However, as Brusilovsky et al. (1994) pointed out, in the visualization of *modified content*, PVs are considered as feedback for exploratory learning and problem solving. Therefore, in this study, we developed our system as an extension plugin to Microsoft Visual Studio Code (VSCode). VSCode is a widely used text editor for coding, and its use was recommended for students participating in the classroom practice described in the next section.

3. Evaluation and Conclusion

We conducted experiments to evaluate the effectiveness of the system developed in this study. We set two hypotheses to be verified: our system can improve student engagement, and our system can improve learners' program understanding. For simplicity, we had learners work on exercises using our system in actual classes and conducted questionnaire surveys to investigate whether they demanded the use of our system for learning. To measure the learners' understanding of the programs, we conducted an evaluation experiment in which we recruited participants and used our system to measure their understanding based on tests.

To investigate whether learners demanded the use of our system, we conducted a five-point scale questionnaire survey after students participated in exercises using the system during actual classes. The exercises were conducted in an actual programming class for second-year university students majoring in computer science, where they observed program behavior using the proposed system. The average score of the responses collected from the 63 students was 3.65. Of the 63 students, two responded with 1 or 2, indicating a negative attitude toward using our system, whereas 32 responded with 4 or 5, indicating a positive attitude. These results suggest that more than half of the users were inclined toward continued use of the system, indicating at least some positive impact on engagement.

An experiment to evaluate learners' understanding of the program was conducted with 22 participants, ranging from first-year university students to first-year graduate students, who responded to our call for participants. After a brief explanation of the evaluation experiment, a pre-test was conducted. The participants were divided into two groups, A and B, with 11 participants in each, so that the average pretest scores in each group were approximately the same. Both groups performed the same exercises. However, to reduce the order effect, the order of the exercises was reversed. Group A worked on exercises using our system, took a middle test, and worked on exercises without using our system before taking a post-test. Group B worked on exercises without our system, took a middle test, and then worked on exercises with our system before taking a post-test.

Although the learning order differed between the two groups, the correct answer rates (CARs) showed a similar trend, and the results did not show a clear advantage in learning using our system. Therefore, based on the results of the pre-test, we further divided the two groups into upper, middle, and lower groups according to their scores and investigated the changes in the scores of each group. Here, grouping based on pretest scores was performed by dividing the subjects into the lower group with scores below the first quartile, the upper group with scores above the third quartile, and the middle group with scores between the first and third quartiles. Table 1 shows the changes in CARs for each group. The underlined values represent the CARs before and after the implementation of the exercises using our system.

Table 1. *Average correct answer rates for each group*

	Upper A	Middle A	Lower A	Upper B	Middle B	Lower B
Pre-test	.784	.580	.417	.834	.570	.384
Middle-test	.884	.860	.750	.900	.740	.634
Post-test	.987	.950	.884	.934	.820	.850

According to the values in Table 1, the CAR for Group A, which used our system between the pretest and middle test, increased by .334, while that of Group B, which did not use our system, increased by .250. Between the middle and post-tests, the CAR for Group B, which used our system, increased by .217, while that for Group A, which did not use our system, increased by .134. Because both groups showed an increase in the CARs when using our system, these results suggest that our system is effective in supporting learners with insufficient understanding of programming concepts. When combined with the results from the subjective evaluations based on questionnaire surveys, the results suggest that our system improves student engagement and program understanding to a certain extent. Although the same trend was not observed in the upper and middle groups of the pretest CARs, we conclude that the two experimental hypotheses were supported to a certain degree overall, even if not fully.

Acknowledgements

This study was supported by JSPS KAKENHI Grant Numbers JP19K12259, JP22K12290 and JP24K15219.

References

Brusilovsky, P., Kouchnirenko, A., Miller, P., & Tomek, I. (1994). Teaching Programming to Novices: A Review of Approaches and Tools. Proceedings of World Conference on Educational Multimedia and Hypermedia (ED-MEDIA94), 103-110.

Kobayashi, K., Kogure, S., Noguchi, Y., Yamamoto, R., Yamashita, K., Konishi, T., & Itoh, Y. (2024). Program Learning Support System with Visualization Reflecting Teacher’s Intent for Learner’s Code. Proceedings of the 32nd International Conference on Computers in Education, 417-419.

Kogure, S., Fujioka, R., Noguchi, Y., Yamashita, K., Konishi, T., & Itoh, Y. (2014). Code reading environment according to visualizing both variable's memory image and target world's status. Proceeding of the 22nd International Conference on Computers in Education, 343-348.

Kogure, S., Ogasawara, K., Yamashita, K., Noguchi, Y., Konishi, T., & Itoh, Y. (2019). Application of Programming Learning Support System to Object-Oriented Language. Proceedings of the 26th International Conference on Computers in Education, 348-350.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of Generic Program Visualization Systems for Introductory Programming Education. ACM Transactions on Computing Education (TOCE), 13(4), 15.

Yamashita, K., Suzuki, Y., Kogure, S., Noguchi, Y., Yamamoto, R., Konishi, T., & Itoh, Y. (2022). Learning Support System Visualizing Relationships Among Classes and Objects Based on Teacher's Intent of Instruction. Proceedings of the 30th International Conference on Computers in Education, 314-316.