Kashihara, A. et al. (Eds.) (2024). Proceedings of the 32nd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Learning Support Environment with Fill-in-Blank Exercise Based on Program Visualization System

Koichi YAMASHITA[a*], Shuya SUZUKI[b], Satoru KOGURE[b], Yasuhiro NOGUCHI[b], Raiya YAMAMOTO[a] , 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 paper describes a learning support system that allows learners to answer code fragments fitting blanks in target program code while observing program visualization (PV), and an evaluation experiment conducted to measure the effectiveness of the system. Most existing PV systems have functions to visualize changes in target domain world caused by each execution of statement and to allow learners to answer questions provided by the system, which is not sufficient in terms of learner engagement. In this study, we extended the PV system developed in our previous study by adding a function to provide fill-in-blank exercise to learners and allow learners to answer it, and a function to visualize the PV according to the learner's answer. The effectiveness of our system was measured by conducting a comparison experiment with a previous system. Although the sample size was not sufficient, the experimental results suggest that learning with our system could contribute to improve learners' understanding of program and that our system would be more effective than the previous system.

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

1. Introduction

Thus far, we have developed a program visualization (PV) system called Teacher's Explaining Design Visualization Tool (TEDViT) (Kogure et al., 2014) and have implemented it in several real classes (Yamashita et al., 2023). TEDViT is a PV system characterized by functions that teachers can freely define the visualization policy, which allows teachers to provide learners with PV consistent with the explanations in their class. However, like most typical PV systems (Sorva, Karavirta, & Malmi, 2013), TEDViT did not have sufficient functions to interact between learners and PV. Therefore, from the viewpoint of learner engagement (Naps et al., 2002), it has not been possible to construct a learning environment with sufficient learning effects.

In this study, we extended TEDViT and developed a system that visualizes program code with some parts are blanked and that provides exercises for learners to answer code fragments fitting the blanks by observing the PVs. Providing such quiz and the answer to it means providing new learning tasks other than observing the PVs to learners, and thus, increases learner engagement. In addition, providing PVs in response to the learner's own answers is expected to support learners forming a hypothesis testing cycle and hence to contribute enhancing learning effectiveness. To evaluate the effectiveness of our system, we conducted a preliminary experiment in which actual university students learned sorting program. The evaluation results suggested that the learners could cultivate better understanding of the program by learning with our system, and that our system would have better learning effectiveness than unextended TEDViT.

2. Proposed System

Styles of quizzes or questions that has long been used in programming education include one in which some parts of the program code are blanked, and learners is asked to answer code fragments that fits the blanks. By presenting this type of question using the PV system, learners can be asked to perform a learning activity of predicting code fragments from changes in PVs, rather than predicting changes in PVs from code fragments. Moreover, it supports learners to form a hypothesis testing cycle in which they form a hypothesis about code fragments based on their knowledge obtained from classes, immediately test their hypotheses by obtaining visual feedback through PVs and revise their hypotheses if there are errors. Therefore, incorporating this function into a PV system is expected to further increase the effectiveness of the learners' learning.

In this study, we developed a learning support system that can provide fill-in-blank questions in which some parts of target program code are blanked. Our system generates a learning environment consisting of four main visualization areas, (A), (B), (C), and (D). Area (A) displays text descriptions navigating learners through the entire learning process. The learner follows the messages in this area to observe PVs and answer questions. In area (B), the target program code is visualized, and the statement being executed is highlighted. Some parts of the code are displayed as blank. Area (C) visualizes the algorithm representation based on PAD. Area (D) is visualized either to provide the target domain world (i.e. the PV) or to input the answers to the fill-in-blank questions, depending on stage of the learning process. Since our system is aimed for novice learners, the answers are not freely written code fragments, but rather are selected from a list of prepared options to prevent grammatical errors from interfering with essential learning activities.

The entire learning process navigated by area (A) consists of the following five steps:

1. Learners learn the behavior of the target program including some blanks in area (B) by observing PVs visualized in area (D).
2. Learners answer code fragments corresponding to the blanks in area (B) by using the input form provided in area (D) in multiple choice style.
3. Learners confirm the behavior of the program with answered code fragment in Step 2 by observing visualization of program filled in the blanks with their own answers in area (B) and its behavior visualized in area (D).
4. If the behavior confirmed in Step 3 is what learners intended, learners finalize their answer and proceed to Step 5; otherwise, cancel the answer and return to Step 1.
5. If the answer is incorrect, our system provides some explanations of their answer in area (D) as feedback, and learners return to Step 1; If the answer is correct, an entire explanation of the question is visualized in area (D).

3. Evaluation

To assess the effectiveness of our system, we conducted an evaluation experiment using our system and TEDViT (hereafter referred to as base system) which is the basis of our system. Twelve participants were recruited from second-year university students majoring in computer science. The subjects were randomly divided into two groups: a group of 6 subjects who learn merge sort program and a group of 6 subjects who learn quick sort program. Subjects in both groups preformed both of learning activity with the base system and with our system in turn. To reduce order effects, subjects in each group were further divided into two groups, A and B, with 3 subjects in each group, and the order in which they used the system was reversed. Subjects took the test three times: before learning (pre), after the first learning (middle), and after the second learning (post), and the learning effect was measured by the change in correct answer rates of those tests.

For simplicity, we do not consider the difference between two target programs here, but compare the change in the correct answer rates between the six subjects in Group A and the six subjects in Group B. The averages of the pre-, middle-, and post-test correct answer rates for the two groups are shown in Table 1. The group of subjects who used the base system in

the first learning (Group A) improved their correct answer rate in middle test immediately after the first learning by an average of 0.410 from the pre-test. The group of subjects who used our system in the first learning (Group B) improved them by an average of 0.525, indicating better improvement in learning with our system. In addition, the group of subjects who used the base system in the second learning (Group B) improved their correct rates in post-test immediately after the second learning by 0.060 on average from the middle test. The group of subjects who used our system in the second learning (Group A) improved them by an average of 0.106 points, again indicating better improvement with our system.

Table 1. *Averages of Correct Answer Rates for Groups A and B*

Group	First system	Pre	Middle	Post
A	Base system	0.217	0.627	0.733
B	Our system	0.228	0.753	0.813

4. Conclusion

In this study, we developed a learning support system that have learners perform fill-in-blank exercises by answering code fragments fitting blanks in target program code while observing PVs, by extending the existing PV system. The effectiveness of our system was measured by conducting an experiment to compare it with a preceding system. The participants in the experiment were 12 second-year university students majoring in computer science, all of whom performed in two learning activities, one using our system developed in this study and the other using the preceding system. Learning effectiveness was evaluated based on the correct answer rates on pre-test conducted before learning, middle-test conducted after the first learning activity, and post-test conducted after the second learning activity. The results of the correct answer rate analysis showed that our system improved the correct answer rates more than the preceding system. The responses to the questionnaire survey indicated that the function to provide PV based on learners' own answers was effective, as we had intended. From these results, we can conclude that our learning support system developed in this study is more effective in supporting novice learners understand the program than preceding systems to a certain degree.

Acknowledgements

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

References

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.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Kohonen, A., Malmi, L., McNally, M., Rodger, S., & Velázquez-Iturbide, J. (2002). Exploring the Role of Visualization and Engagement in Computer Science Education. *Working Group Reports from the 2002 Conference on Innovation and Technology in Computer Science Education*, 131-152.

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., Soma, H., Kogure, S., Noguchi, Y., Yamamoto, R., Konishi, T., & Itoh, Y. (2023). Program Visualization System Supporting Teacher-Intended Stepwise Refinement. *Proceedings of the 31st International Conference on Computers in Education*, 435-440.