Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

Design of a Learning Support System and Lecture to Teach Systematic Debugging to Novice Programmers

Raiya YAMAMOTO[a*], Yasuhiro NOGUCHI[b], Satoru KOGURE[b], Koichi YAMASHITA[c], Tatsuhiro KONISHI[b] & Yukihiro ITOH[d]
[a]*Graduate School of Science and Technology, Shizuoka University, Japan*
[b]*Faculty of Informatics, Shizuoka University, Japan*
[c]*Faculty of Business Administration, Tokoha University, Japan*
[d]*Shizuoka University, Japan*
[*]dgs15009@s.inf.shizuoka.ac.jp

Abstract: In our previous research, we developed a learning support system for teaching a systematic debugging process. However, it was observed that all the subjects were unable to learn the entire process but some managed to learn it partially at experimental evaluation. We considered following two reasons: (1) the level of debugging skills we expected for subjects was higher than their actual level and (2) subjects were not offered lectures about the debugging process and required skills but were provided with only some exercises. For (1), we consider that they need to gain more basic debugging skills. For example, we decided to teach a skill to observe variables' values. For (2), we designed a lecture that learners could attend to learn the process and debugging skills. The lecture has instruction and exercise parts. Next, we extended our learning support system to be adapted to the lecture and assist learners who face difficulties in the exercises. In this paper, we report a design of the learning support system and the lecture.

Keywords: Programming education, debugging learning, interactive learning support system

1. Introduction

In previous research, we developed a learning support system for learners who do hit-or-miss debugging to assist learning a systematic debugging process (Yamamoto et al., 2015). However, from a result of an evaluation, all subjects were unable to learn the entire debugging process but managed to learn it partially. We considered following two main reasons; The level of debugging skills we expected from the subjects was higher than their actual level (**Reason 1**). Subjects were not offered lectures on the debugging process and skills required but were provided with only three exercises (**Reason 2**).

First, as for Reason 1, we designed the learning support system by considering that the subjects have adequate skills to achieve the process; however, they did not. Judging from the experimental evaluation, we considered they required training in obtaining basic debugging skills. Therefore, in the proposed process, they learned not only the debugging process but also basic debugging skills required in the process. They need to gain skills for arranging function relations and for checking data flow for selecting a function that can contain bugs. In addition, they needed skills to evaluate values of variables and to focus on a statement that requires to be checked.

Second, as for Reason 2, the learning situation in the experimental evaluation corresponded to learning discovery. It is obvious that learning with the help of only three exercises is difficult for subjects. Therefore, we explicitly designed a lecture for teaching the debugging process and the skills required. Miljanovic (2015) devised a game-based learning support system called RoboBUG to learn debugging. In his teaching method, he conducted a lecture to teach the concept of debugging before learners started learning debugging by using the system. However, his method imparts partial debugging skills and does not focus on the entire debugging process. Thus, we devised a lecture that

teaches learners the entire debugging process and skills. The goal is to learn the debugging process and skills required to debug programs that contain several or various functions. However, it can be difficult for some learners to start learning through debugging programs containing some functions. Therefore, we divided the process and the skills into two parts and designed two lectures: A lecture for teaching the debugging process and skills required to debug programs that contain only the main function (**Lecture 1**) and a lecture for teaching the debugging process and skills required to debug programs that contain several or various functions (**Lecture 2**).

In addition, we designed exercises to help learners practice using the processes and debugging skills they learned in the lecture. Next, we analyzed when learners experienced difficulty in the exercises and developed a learning support system to assist those learners. We extended the learning support system we developed in the previous research because the previous system was unable to support the learning of some skills in the lecture. The system should work as a scaffold, that is, it should adjust its assistance according to the student's skill level.

In this paper, we report the design of a learning support system and lecture for learners in the desired level.

2. Designing Lectures

2.1 Basic Ideas

In this section, we describe the design of the lectures as discussed in chapter 1. We need to design two lectures for novice learners to be able to learn the entire debugging process and acquire skills. First, we designed the process as the goal of the lecture, as shown in Figure 1, by combining our experiences and knowledge from the references (Myers et al., 2012, Zeller, 2009). The debugging process and skills taught in the lectures correspond to the figure. Further, we must start teaching the debugging process and skills for debugging programs containing only a main function, so we must divide the process in Figure 1 to fit each lecture. Section 2.2.1 describes the process that is scaled down for the lecture to teach debugging program that contains only a main program.

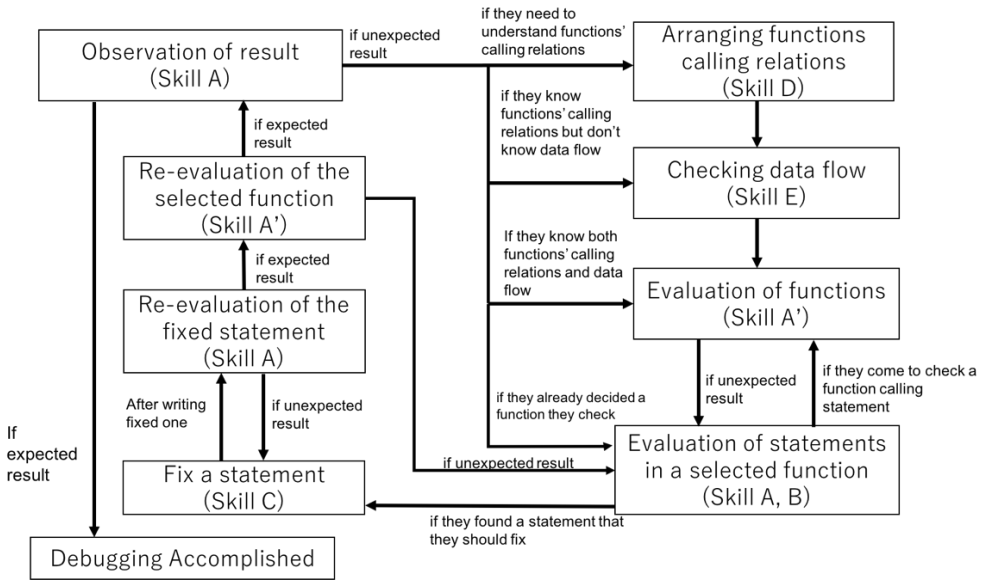

Figure 1. Process established as the goal of the lecture

In addition, learners must have sufficient practice to learn the process and skills. We decided that each lecture must be of 90 min. The instruction part is of approximately 30 min and the exercise part is of 60 min. We considered approximately 5–10 min for each exercise so that the learners may practice with at least five or six exercises at every lecture. This number of exercises is approximately twice more than the number of exercises in the previous experimental evaluation, so we think this

number can be enough. The learning support system is provided to assist learners who have difficulties during exercises. The concrete features of the system are described in Chapter 3.

2.2 Details of the Lectures

2.2.1 Detail of Lecture 1

In this lecture, learners learn a process to debug programs that contain only a main function. Figure 2 shows a scaled down process for this lecture. Learners must learn three skills to achieve the process shown in Figure 2: Skills A, B, and C.

Skill A involves the evaluation of variable values and execution of branch and iterative statements in the program. It includes two subskills: Skill A-1 involves the consideration of expected values and execution of branch and iterative statements; Skill A-2 involves the observation of the actual value and execution process. For Skill A-1, learners learn that they must consider expected values and execution process. Learners who are unable to consider expected values will be asked to understand the algorithm and specification before starting debugging. For Skill A-2, they learn to insert output statements to inspect actual value and actual execution process.

Skill B involves the verification of correctly executed area and unexpected executed area in a source code. This skill consists of two subskills: Skill B-1, to focus on a particular statement and SkillB-2, to evaluate the focused statement using Skill A. They learn how to shift their focus to next statement. For Skill B-1, various strategies exist; therefore, we selected a simple moving strategy, in which program statements are checked from top to bottom. The learners learn this strategy in this lecture. For Skill B-2, they learn to evaluate the statement selected using Skill B-1 and limit possible areas that may have a bug according to the evaluation results.

Skill C involves the fixing a statement containing a bug. By using Skills A and B, the learners can find a statement that functions unexpectedly. There are various methods for fixing codes; therefore, we did not investigate this skill intensely in this lecture. Instead, we provide some typical examples for fixing statements.

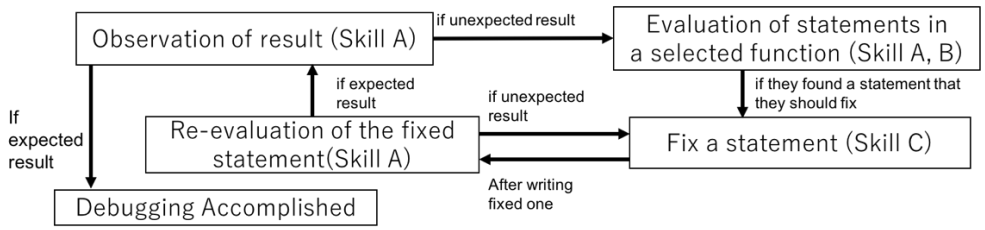

Figure 2. Process that learners learn in the first lecture

Finally, teachers demonstrate an example of managing an entire process to learners using the acquired skills. The students then practice to debug programs containing only a main program through exercises.

2.2.2 Detail of Lecture 2

In this lecture, learners learn the process of debugging programs containing functions in addition to a main function. Figure 1 shows the goal for this lecture in the learning process. The process consists of six skills: Skills A, B, C, D, E, and A'. The learners must learn these skills to complete the process; however, Skills A, B, and C are the same skills that they learned in the first lecture. Therefore, they learn Skills D, E, and A' additionally in this lecture.

Skill D involves the arranging of the program structure. It consists of two subskills. In Skill D-1, functions in a source code are listed. In Skill D-2, function calls are arranged and a diagram of function's calling relations in a program must be drawn. In this lecture, we explain skill D-1 can be used if they know the syntax of functions and find a function block. Skill D-2 can also be used if they know the syntax and can recognize which function and their calling relations they must check.

In Skill E, the data flow among functions is checked. Skill E has two subskills: Skill E-1 involves the determining of functions being called in the current execution. Skill E-2 involves the tracing of functions called through a diagram. Skill E can be learned if learners know function syntax and can follow the flow in the diagram of function's calling relations.

After selecting a function, the learners have to check the function. Next, a skill is required for evaluating the working of the function: Skill A'. We explained that the use of Skill A' is similar to Skill A except that a statement-calling function must be determined. Next, an argument(s) and return value must be evaluated to confirm the called function works correctly. The learners acquire knowledge of inserting an output statement to check the values of arguments and return value.

In the last part of the lecture, teachers demonstrate an example of managing an entire process to learners like the first lecture. They have exercises in the second lecture, too.

3. Design of a Learning Support System

We designed a learning support system to support learners in practicing debugging through exercises. The three WorkSheets (WS) are provided. WS has structures that correspond to the process. Learners can learn the process by working on WS repeatedly. In addition, all WS can work as scaffolds, that is, they reduce their support functions according to learners' skill level. The basic strategy of reduction is to cut a part of the structures of support functions that corresponds to a skill that the learners have already acquired. Then, functions that WS can provide to the learners will become almost same with the ones that ordinary programming environments have. We expect that they move to the environment when they notice it.

3.1 WS for Learning Debugging Process and Skills to Debug a Selected Function (WS0)

This WS (Figure 3) assists in learning the debugging of a selected function. The system corresponds to the process and skills learned in the first lecture. Furthermore, this WS encompasses Skill A'. In exercises for these skills, learners may face the following difficulties; They cannot shift their focus to next statement (**Difficulty 1-1**). They cannot consider expected values and execution process of branch and iterative statements correctly (**Difficulty 1-2**). They cannot observe the actual values and execution process of branch and iterative statements correctly(**Difficulty 1-3**). They are unable to identify the process they must perform (**Difficulty 1-4**).

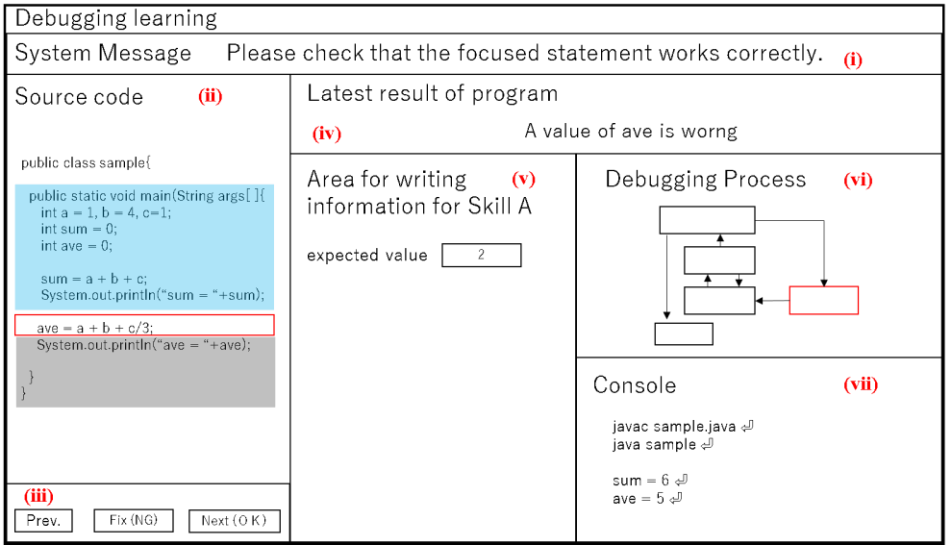

Figure 3. WS0 user interface

The WS shows system message on area (i) at each step. Reading the message, they know what to do next. For Difficulty 1-1, the WS provides the function that learners can color backgrounds of source codes on area (ii). They can learn how to classify statements in their source

code by this function. For Difficulty 1-2, the WS show the type of information they must consider to evaluate on area (v). When a statement is focused on, a data input box(es) appears on the area. Learners learn that they must consider the expected values and execution process of branch and iterative statements by filling the box(es). For Difficulty 1-3, the WS can insert a correct statement(s) to source code on area (ii). Learners refer to it to learn how to insert one. For Difficulty 1-4, the WS shows the step of the debugging process they are at on area (vi).

3.2 WS for Learning to Understand a Program Structure (WS1)

This WS (Figure 4) is used for learning Skill D that is used for arranging functions calling relations. In the exercise for Skill D, learners may experience the following difficulties; They may not remember what they need to do in the step involving Skill D (**Difficulty 2-1**). They may not be able to confirm the correctness of the diagrams they constructed (**Difficulty 2-2**).

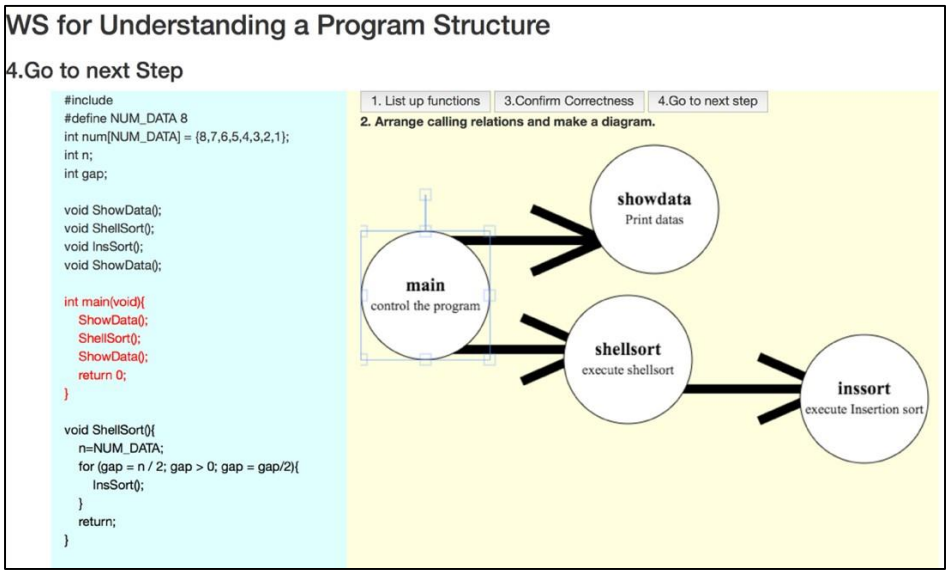

Figure 4. WS1 user interface

For Difficulty 2-1, they learn the following procedure using WS1. Phase 1 involves listing of functions in the source code, phase 2 involves arranging the functions calling relations and constructing a diagram of calling relations, and phase 3 involves the confirming of the correctness of the created structure model. WS1 consists of buttons and areas that correspond to each phase (Figure 4). Button 1 starts phase 1. Working area 2 displays suggestions to learners to perform phase 2. Button 3 initiates the system to start phase 3. For Difficulty 2-2, the system evaluates the diagram by pushing Button 3. Learners can check for the correctness of their diagram. To achieve this support function, the system has corresponds the correct diagram with the source code.

3.3 WS for Learning to Check Data Flow (WS2)

This WS (Figure 5) supports the learning of Skill E that is used for checking data flow. In the exercise for practicing Skill E, learners may experience following difficulties; They may not be able to remember what they need to do for Skill E (**Difficulty 3-1**). They may not be able to trace the data flow on their diagram, which they check in their source code (**Difficulty 3-2**). They may not be able to confirm the correctness of data flow they constructed (**Difficulty 3-3**).

For Difficulty 3-1, they learn the following procedure by using this WS. In Phase 1, they select statement-calling function that is executed next. In Phase 2, they trace the executed flow. Next, they move to the called function and check the calling statements or return statement in the called function and perform Phases 1 and 2 repeatedly. Finally, in Phase 3, they confirm the correctness of their constructed data flow. This WS consists of a button and areas that correspond to each phase, similar to WS 1. Working areas 1 and 2 suggest Phases 1 and 2, respectively. Button 3 initiates the

system to start Phase 3. For Difficulty 3-2, the system labels the diagram by using numbers according to the flow they checked in the source code. The numbers indicate the order of data flow, thus helping the learners confirm the data flow. For Difficulty 3-3, the system evaluates the data flow through clicking of Button 3 as in WS1. Learners can check the correctness of their data flow. To achieve this support function, the system corresponds the correct data flow with the source code.

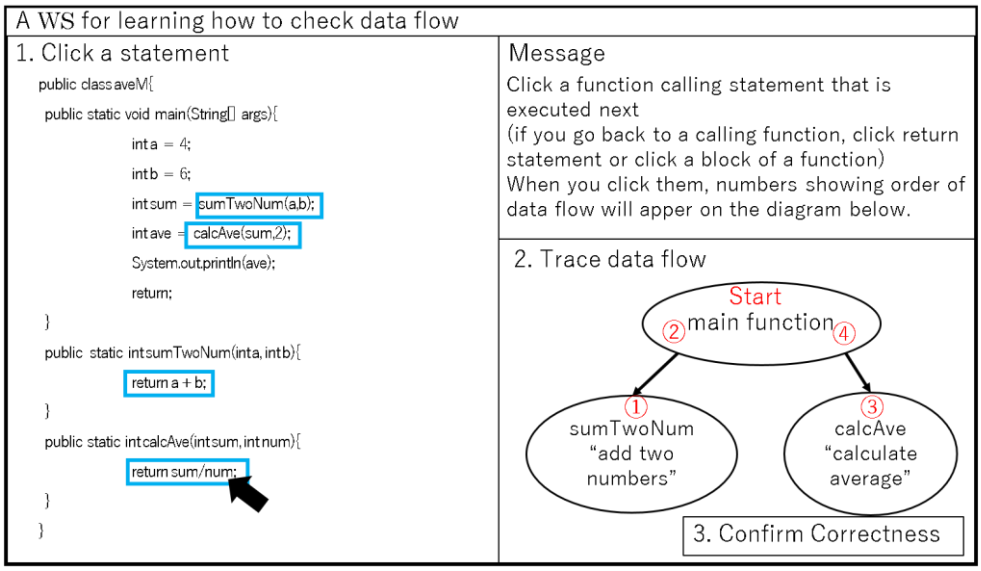

Figure 5. WS2 user interface

4. Conclusion

In this study, we designed a lecture with a learning support system for teaching systematic debugging to novice programmers. Learners can gain the required knowledge through the lecture and practice using the system support. Currently, we are evaluating the effectiveness of Lecture 1 and WS0. In future, we plan to implement remaining WS and lectures and carry out experimentally evaluate the effectiveness of whole debugging learning system that includes the lecture and the learning support system.

Acknowledgements

This research is supported by Japanese Grant-in-Aid for Scientific Research (B) 24300282.

References

Yamamoto, R., Noguchi, Y., Kogure, S., Yamashita, K., Konishi, T., & Itoh, Y. (2015). Construction of an Environment to Support Learning Systematic Debugging Process with Worksheets and Synchronized Observation Tool. *Proceedings of International Conference on Computers in Education 2015*, 269-274.

Myers, J, G., Badgett, T., & Sandler, C. (2012). The Art of Software Testing 3rd Edition. Hoboken, NJ: John Wiley & Sons, Inc.

Zeller, A. (2009). Why Programs Fail: A Guide to Systematic Debugging Second Edition. Burlington, MA: Morgan Kaufmann Publishers.

Miljanovic, A, M. (2015). RoboBUG: A Game-Based Approach to Learning Debugging Techniques (Master's thesis, University of Ontario Information and Technology, Oshawa, Canada). Retrieved from http://hdl.handle.net/10155/536