2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI) | 978-1-6654-9755-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IIAIAAI55812.2022.00044

A Support System for Programming Exercises Using Test-first Approach

Kana Suetake
Graduate School of Engineering
Tamagawa Univresity
Tokyo， Japan
t5jc22232@stu.tamagawa.ac.jp

Takafumi Tanaka
Callage of Engineering
Tamagawa University
Tokyo， Japan
tanaka_t@eng.tamagawa.ac.jp

***Abstract*—In programming exercises for novice learners, the learners often satisfied with that their programs 'work' at a first glance, and they neglect to test the programs work properly. Making test cases and modifying programs are necessary training for the learners to comprehend behavior of the programs. Therefore, we propose a method for learning programming using test-first approach and a system to support it. Test-first approach is an approach for programming that a programmer makes tests before making programs. Therefore, we decided to apply the approach to the exercises in order to get the learners in habit of testing their programs. The system supports the learners to make test cases and evaluate them. When a learner submits her/his test cases for a question, the system points out missing test cases and gives hints based on a correct answer by a teacher. When the learner submits her/his program, the system executes the tests made by the learner and the teacher and displays the results. This paper describes the method, system, and a preliminary experiment to evaluate the system.**

Keywords—Programming exercise, test-first approach, unit testing

I. INTRODUCTION

In programming exercises for novice learners, the learners often satisfied with that their programs 'work' at a first glance, and they neglect to test the programs work properly. Generally, few examples of inputs and expected outputs are given for a question. A learner read the question and write a program which satisfies requirements of the question. However, the learner tends to judge a correctness of the program simply by checking the given samples of inputs and outputs in the question. To determine whether the program works correctly or not, it must be verified that it produces an appropriate output for each of the various inputs. To master programming, it is important for learners to be able to find the pairs of inputs and expected outputs by themselves early in the learning. However, these kinds of training are not emphasized in the early stage of programming learning.

On the other hand, agile development including a test-first approach has been increasingly adopted in software development in recent years. The test-first approach is an approach for programming that a programmer makes tests before making programs. It is effective to comprehend requirements to be satisfied.

Therefore, we apply the test-first approach to programming education for novice learners.

II. RELATED WORKS

Test-Driven Development (TDD) is a method including the test-first approach which is adopted in agile development [1]. TDD is a practice in coding process that developers make test cases first and then make a code which passes the tests. Then, the developer makes the code by repeating making test cases, modifying the code, and refactoring. Although our method dose note require the learners to be advanced in the exercises, a concept of the test-first approach has much in common with our method.

Fukuyama et al. proposed a Java programming education support system based on a test-driven development method to support the students' learning and reduce teachers' workload [2]. This system is a web application that the students create their programs and verify them online for programming exercises. To verify the programs, the students create and run test codes that for determining the programs satisfy requirements described in specifications for the exercises.

Fukuyama et al. developed a programming learning support system based on the TDD method. However, it has a problem that it is difficult to have novice learners create test codes. In this study, since the system is intended for use by novice learners, the learners make pairs of input values and an expected output value (hereinafter referred to as "test cases") and evaluate them with the system.

Hachisu et al. developed a support system for evaluating learners' test cases. The system provide a web IDE and collect learners' test codes from the IDE. Then, the system evaluates test cases in the test codes based on criteria that are defined by a teacher based on objectives of a question and leaners' levels of understanding [3]. Then, the system gives feedback including results of the evaluation. Hachisu et al. aim to enable the learners to design their test cases properly. When a learner enters a program and test cases, the system generates a test driver and executes tests using the test cases. It reduces the learner's time and burden to make the same input for each time to execute the tests. The study by Hachisu et al. is similar to our study in a concept of automatic test evaluation. However, it does not mention instructions for test-first development procedure.

There are several challenges for novice learners in learning programming. An example is that they may not be able to adequately predict expected output of a function for a set of

978-1-6654-9755-8/22/$31.00 ©2022 IEEE
DOI 10.1109/IIAIAAI55812.2022.00044

arguments. Katagiri et al. said that this can be avoided by organizing pairs of input values and an expected output from specifications and making programs accordingly [4]. Therefore, Katagiri et al. propose "Neko," a coding support tool for beginners. In the system, a learner gets pairs of inputs and expected outputs that to be satisfied. The learner makes a program that satisfies the pairs. Then, the system determines the learner's code satisfies the pairs.

This study proposes a method and a support system of the method that learners make test cases and programs in test-first approach without making test codes.

III. Proposed Method

This section describes a programming education method for novice learners using a test-first approach. In general, a class in a basic programming course is carried out in the following steps: (1) Lecture on the syntaxes and typical usage of them by a teacher, (2) The learner does basic exercises using the syntaxes, (3) The learner does advanced exercises and submits their programs, and (4) The teacher gives the learners feedback on her/his programs.

However, the learner often neglects to check her/his programs to satisfy the requirements by the exercises in the step (3). Therefore, we decided to apply the test-first approach to the programming class. 'Test-first' is an approach of programming that a developer makes test cases before she/he makes a program. A 'test case' consists of inputs and an expected output. Then, she/he makes a program that passes the test. After that, she/he considers new test cases and modifies the program to passes them. Then, she/he repeat it until the program satisfies the requirements.

A benefit of this approach is that the learner can realize goals of the programs and how to validate the program before they make the programs.

To make the learner understand the test-first approach, in the step (1), the teacher explains what kind of test cases are necessary to test the programs including the syntaxes taught in the class.

In the step (2), the teacher shows the learner concrete examples of valid test cases. After that, the teacher explains why the test cases are needed to check the programs' behavior and how to make test cases.

In step (3), the teacher makes the learner to do the exercises using test-first approach. While she/he is making a program, she/he executes the tests frequently and refactors the program. Then, she/he submits the program after she/he confirms that the program passes all of the test cases.

In this study, we develop a system for the novice learners to support the step (3) that provides supports for them to make test cases, execute the tests, and get feedbacks for a sufficiency of the test cases and results of the test executed.

Procedure of the learners' activities in the step (3) with the system is as follows.

(1) Making test cases.

The learner reads a problem description through the system. The description is displayed in the frame above the input form in Fig.1. Then, she/he makes test cases that are necessary to validate the program to make. The system supports it by providing input forms for sets of 'input values' and 'expected output value' by a simple user interface. Initially, the system shows an input form of a test case. The form consists of a form of "input values" and a form of "expected output value". A user is able to add a form of a test case by pressing "Add Line ". She/he is also able to add a form of "Input Values". (Fig.1).

(2) Making program.

The learner makes a program in her/his own development environment. Then, she/he submits the program to the system. In this study, the program restricted that it has a main method and one or more methods to that takes arguments and outputs a return value. In addition, the methods are limited to pure functions that their return values are determined only by their arguments. It is important for the novice learners to understand features of a pure function and how to make it.

When the learner submits her/his program, she/he select a method to test from the program. The system executes tests using her/his test cases and the teacher's test cases registered with the question in advance. Then, the system gives the learner feedbacks including hints on a sufficiency of the learner's test cases and results of the executed tests (Fig.2).

(3) Modifying the program based on the feedbacks by the system.

The learner modifies her/his program and test cases based on the feedbacks. She/he repeat this procedure (1) to (3) until the program and test cases completed.

IV. System Development

This section describes a developed of our system to support the proposed method.

A. Overview

This system is intended to support novice learners in programming. When the learners work on an exercise, the system supports them to make a program using the test-first approach. Figure 3 shows an architecture of the system.

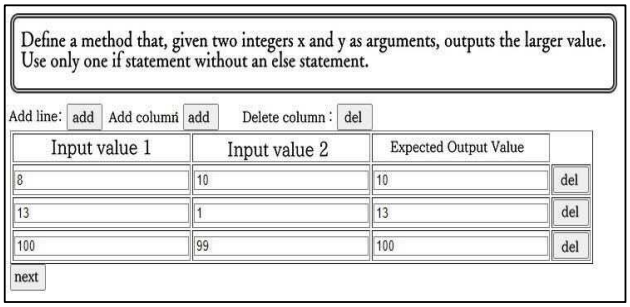

Fig.1 Example of test case creation.

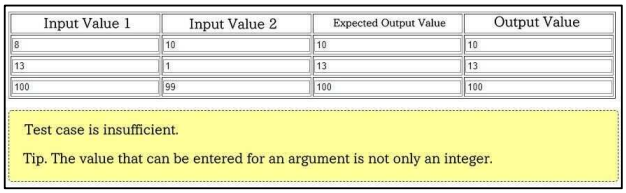

Fig.2 Execution result screen.

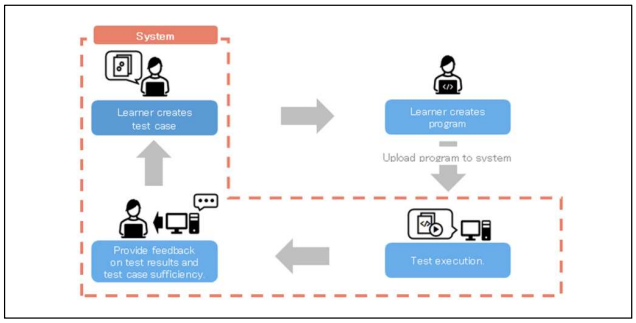

Fig.3 Overview of our method and system.

Learners send test cases and a source code to test to the system. When the system tests a code, the system extracts methods from the source code. Then, the system assigns values of the test case to the argument of the method and executes it. Finally, the system displays results of the test.

B. Features of the system

(1) Comparison of test cases

Test cases made by learners are compared with the model test cases prepared by a teacher to confirm that the learners, made the necessary test cases.

(2) Execution of tests

The system extracts only specified methods from the uploaded source code and executes the test cases, made by the learner. The model test cases prepared by the teacher will also be executed. Then, the system compares actual results obtained from the executions with the expected results described in the test cases.

(3) Visualization of results of the test

The system displays the results for the learners. If a test case is insufficient, a "hint" is displayed to help the learner figure out what the missing test case is. The "hints" section allows the faculty member to set the content of the hints when creating a model test case. Hints are displayed in a yellow frame (Fig.2).

V. EXPERIMENT

This section describes a preliminary experiment to evaluate effectiveness of our method and system to support the test-first approach. The number of participants in the experiment was four. There were three undergraduate students majoring in software science at Tamagawa University and a teacher who teaches programming at the university.

A. Procedure

Procedure of the experiment is as follows.

(1) Programming without the test-first approach and the system (30 min)

The participants answer a couple of programming exercises.

(2) Understanding the test-first approach and operation of the system.

We explain the test-first approach and operation of the system.

(3) Programming with the test-first approach and the system (60 min)

The participants answer a couple of programming exercises which have similar difficulty to the exercises in the step (1). Time limit of this step was set at 60 minute for two reasons: first, participants were not accustomed to the test-first approach; second, they were not accustomed to using the system.

(4) Questionnaire

The learners answer questionnaire that asks effectiveness of our method and system.

TABLE I shows a part of questions in the questionnaire. Q1 to Q5 are answered on a scale of 1 (disagree) to 4 (agree) while Q6 is answered on a description format.

TABLE I Questions of the questionnaire.

No.	Question
Q1	Making test cases was difficult.
Q2	Test-execution support was useful.
Q3	Hints shown in the result of tests were useful.
Q4	I understood the test-first approach.
Q5	A function of automatic test execution is useful to learn software test.
Q6	Where is the point that you feel difficult in making test cases?

TABLE II Answers for Q1 to Q5.

No.	1	2	3	4
Q1	2	1	1	0
Q2	0	0	0	4
Q3	0	1	2	1
Q4	0	0	0	4
Q5	0	0	1	3

TABLE III Answers for Q6.

No.	Answer
1	To determine inputs was difficult.
2	I didn't think it is difficult. However, I thought it takes time and effort. It would be better that the system remember my input because I need to use the system repeatedly.
3	It is difficult for a learner who doesn't have an experience that she/he forget to think a boundary of $x < 0$ for a condition $0 \leq x \leq 3$ to make test cases for the boundary.
4	(No answer)

TABLE IV Scores without the system.

	S1	S2	S3	S4
Q1	0	3	2	2
Q2	0	1	4	1

TABLE V Scores with the system.

	S1	S2	S3	S4
Q1	3	3	4	4
Q2	4	4	4	4

B. Results

TABLE II shows a result of the Q1 to Q5 while TABLE III shows a result of the Q6 by each participant. TABLE IV shows scores of the programs. The scores are the correctness of the programs on a 5-point scale from 0 to 4. The score 0 means that the program is incomplete or has errors. The score 1 means that the program is completed and works. The score 2 means that the program is completed and complies with requirements of the question. The score 3 means that the program was tested with test cases described in the question. The score 4 means that the program was tested with necessary test cases to determine correctness of the program even if they are not described in the question.

VI. Discussion

According to the answer for Q2 and Q5 shown in TABLE II, the participants felt that the support of automatic test was effective. In addition, as the answer for Q4, the participants were able to understand the test-first approach. From these results, it is suggested that the proposed system is effective for learners to learn the test-first approach.

A participant answered 1 to Q3. We consider a reason of this is that the hints shown by the system was not informative enough for the participant. It is a future work that we find kinds of effective hints which are adopted for each learner and types of questions.

For Q1, most of the participants answered 'negative' that means making test cases is not difficult. As a reason for this, we consider that the questions were easy for the participants who the 4th undergraduate students were. In addition, we observed that the participants who answered 1 or 2 for Q1 organized the questions by diagraming requirements of the questions. It is suggested that diagraming is effective to reduce the difficulty of making test cases.

According to Table 3, No. 1 responded that it is not difficult to come up with input values. No. 3 responded that without experience of thinking about before and after boundaries, it would be difficult to create a test case for boundary values. It turns out that it is not difficult to come up with output values. It is important to provide support in think of input values.

According to TABLE IV and V, the scores with the system are higher than the scores without the system. In addition, the participants were able to make almost all necessary test cases with the system. Thus, the participants were able to make their programs which satisfy the requirements by the questions.

Without the system, the participants did not actively try to test their programs. Especially, the participant, S4, was the fastest to finish the exercises. However, S4 did not test her/his programs at all. Therefore, S4's program for Q2 did not work according to the requirements.

The reason why the scores with the system were high is that the participants determined the necessary test cases before making programs, and they made their programs based on the test cases. In addition, they could easily compare actual outputs by their programs and expected outputs described in the test cases. It increased their chances of self-reviewing of the programs and realizing their mistakes. We conclude that this contributed to the improvement in the quality of their programs.

VII. Conclusion And Future Work

In this study, we proposed a method and a system to support novice learners in programming learning to make their program and test cases. We conducted an experiment to verify effectiveness of our system. According to the result, it is suggested that (1) using the test-first approach with our system is not difficult, (2) supports by the system is basically useful, and (3) test-first approach gives learners chances to consider many test cases.

As a future work, we plan to improve the system by keeping a history of test cases. In addition, we are going to implement a function to organize questions of exercise. After that, we conduct an experiment with a lot of first-year undergraduate students in a basic programming course.

Acknowledgment

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in Aid for Young Scientists, 2020-2022 (20K19941, Takafumi Tanaka)

References

[1] Kent Beck, Test-Driven Development By Example, Addison-Wesley Professional, 2002.

[2] N. Funabiki, Y. Matsushima, T. Nakanishi, and K. Watanabe, "A Java Programming Learning Assistant System Using Test-Driven Development Method," IAENG International Journal of Computer Science, vol.40, no.1, pp.38-46, Mar 2013 .

[3] Y. Hachisu, S. Kobayashi, A. Yoshida, and K. Agusa, "Test Case Evaluation System for Programming Exercises," JSSST Journal of Computer Software, vol.34, no.4, pp.54-60, Nov. 2017 (in Japanese).

[4] K. Katagiri and S. Sakai, "Learning Support for Programming Beginner with Functions to Orgnize Inputs and Expected Results based on Specifications," Proceedings of the 81th National Convention of IPSJ, p p.769-770, Jan. 2020.