Development of a System using Interactive Video for Novice Programmers

Tatsuyuki Takano, Kanto Gakuin University, Japan
Kentaro Matsui, Tokyo Denki University, Japan
Osamu Miyakawa, Tokyo Denki University, Japan
Takashi Kohama, Tokyo Denki University, Japan

The Asian Conference on Society, Education & Technology 2015
Official Conference Proceedings

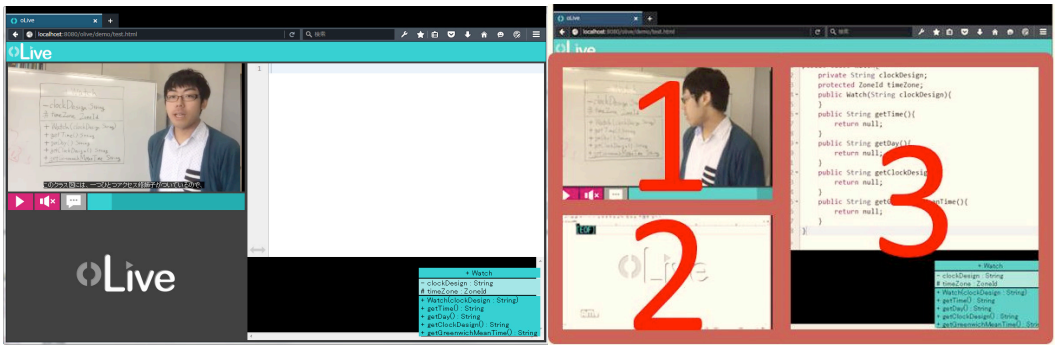
Abstract
In e-learning in programming education, there is a method used of creating a program using a lecture video and textbook, and then submitting a file. However, when errors occur in the files submitted by students, it is often unclear when the errors occurred. In such cases, it is desirable to teach the timing of error occurrence. With regard to interactive video, the video contents change according to the televiewer's situation or selection. The system that we have created presents a video and view design with the domain of an editor for programming education. A student creates a program according to the directions given on the video. Then, in cases where the student's program contains errors, even at the middle stage of program creation, the video changes and the error is pointed out with statements such as "no semicolon is found" or "the method name contains a spelling error," etc. This article reports on the timing details that change the analysis method of student coding and the videos used in the system.

Keywords: programming education, software engineering, e-learning

Introduction

In e-learning in programming education, there is a method used of creating a program according to a lecture video and textbook, and then submitting a file. However, when the file that a student has submitted contains an error, it is often unclear when that error occurred. Therefore, it is desirable to teach the timing of error occurrence. With regard to interactive video, the contents of a video change according to a televiewer's situation or selection (Dongsong, Zhou, Briggs, & Nunamaker, 2005). Students create a source code in accordance with instructions given on a lecture video; the purpose of this research is to develop a system in which the video urges error correction, if it detects an error during the creation of a program.

The appearance of the system



(a) (b)

Figure 1: Main screen

The main screen is shown in Fig. 1 (a). This screen is displayed after login and the lecture video begins immediately. Fig. 1 (b) shows the composition of the main screen, which consists of the following three parts.

1. Lecture video (main video)
2. Sub video
3. Text editor

A student inputs a source code into a text editor according to directions given in the lecture video. The teacher's personal computer screen, etc., is used to assist the lecture video, and a sub video flows through the teacher's screen and synchronizes with the lecture video. Each video chapter is divided into steps based on the input method procedure specified by the source code in the lecture video. A program block is made into a subject about the method of input procedure, and the target programming language is Java (Tatsuyuki, Takashi, & Osamu, 2014).

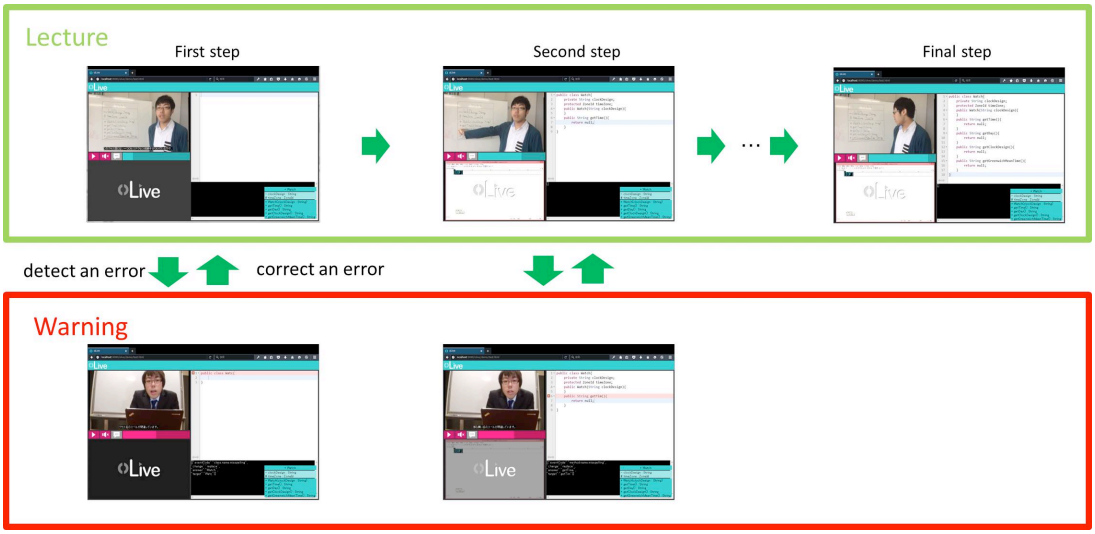


Figure 2: Learning process using the system

The process of study using this system is shown in Fig. 2. A lecture video consists of two or more chapters and each corresponds with one study step. The student inputs the source code into the text editor for every step. The student does not proceed with the lecture video until the step has been completed. However, if a student has created the source code before the lecture video is complete, that step is skipped and the next chapter ensues. Furthermore, when the source code that the student has created contains a spelling or other error, the video changes to one that urges error correction. The error correction video consists of chapters that correspond to various error patterns, and the chapter that corresponds to the student's error is displayed. Once the student has corrected the error in the source code, the lecture video resumes. The student completes the source code by means of these processes.

Algorithms

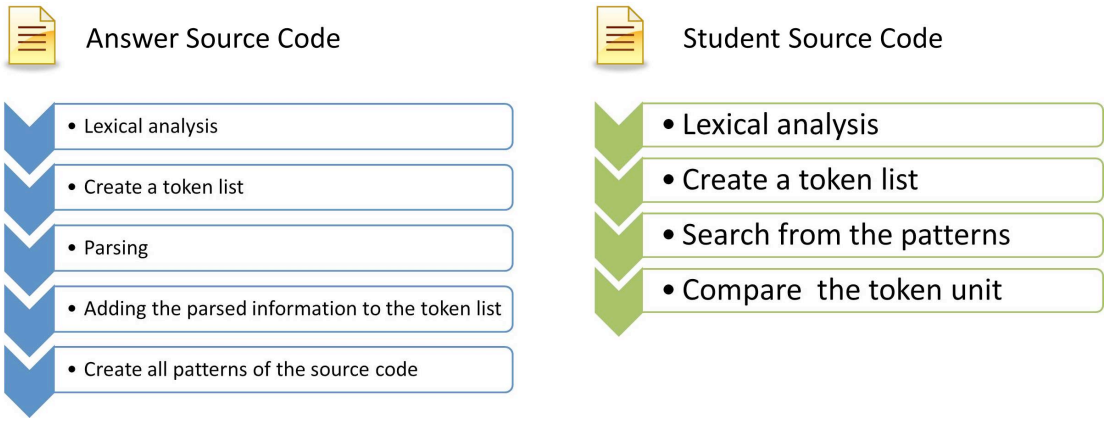


Figure 3: Outline of algorithms

The outline of the algorithm of error detection is shown in Fig. 3. The algorithm is divided into analysis of the source code of an answer, and analysis of the student's source code. First, a lexical analysis and syntax analysis are conducted for the source code of an answer. Then, the pattern of the source code corresponding to each step of the lecture is created. A search is made to analyze correspondences between the student's source code and the pattern list. The pattern is compared per lexical token and in this way errors are detected.

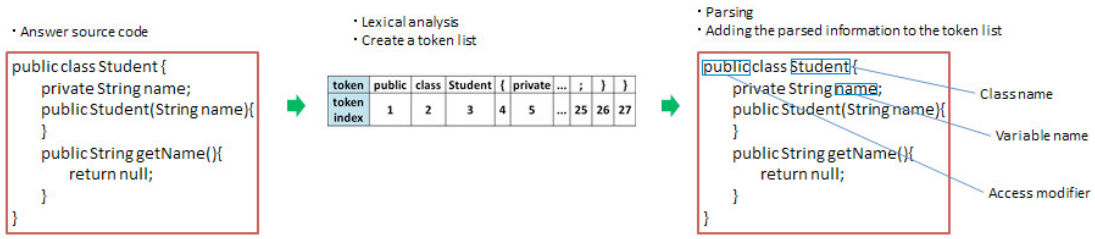


Figure 4: Analyze answer source code

Fig. 4 shows the process up to the analysis of the source code of an answer. A token sequence is created for the source code of an answer through lexical analysis. Moreover, syntax analysis is conducted from the token sequence, and information, including an access ornamentation child, a class name, a variable identifier, etc., is connected to a token sequence from the result.

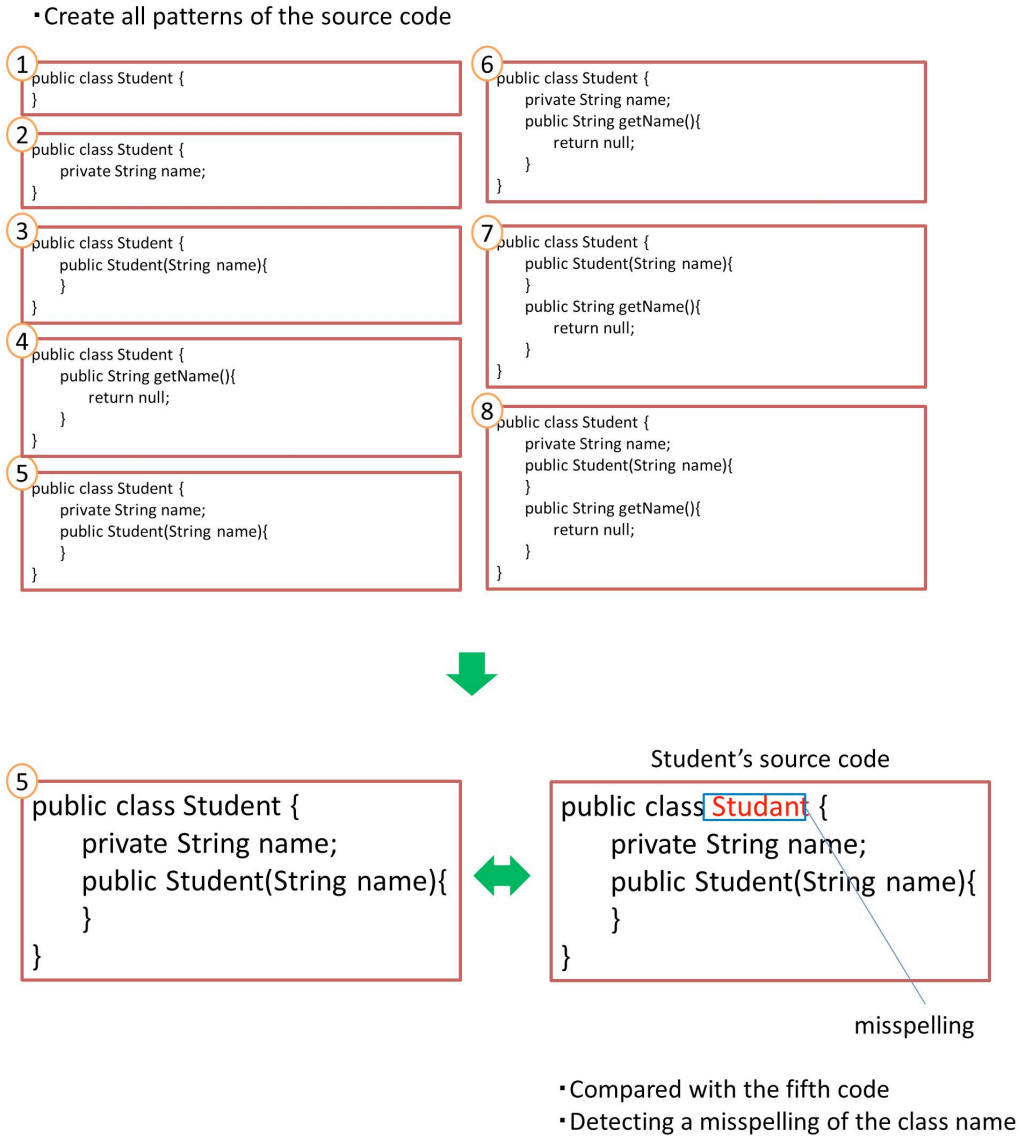


Figure 5: Creating patterns and detecting errors

The process of detecting an error from creation of a pattern list is shown in Fig. 5. The source code of the analyzed answer is classified into units, such as field and method, and creates the list of patterns that cover each step of a lecture video. Next, the lexical pattern of a student's source code is analyzed and a pattern search is conducted to identify the most similar code from the pattern list. A comparison is made of the pattern and lexical tokens, and errors are pinpointed. By analyzing the source code of an answer, the type of error can be identified, such as that the class name is misspelled. When there is 75% of similarity based on the distance between the character strings by Levenshtein distance, it is judged as a spelling error. Moreover, in the case of an anagram where the character order has been changed, this is also judged a spelling error. Sixty-nine types of error are covered in the error video. A student's source code may be in a state where syntax analysis is impossible, if, for example, there is no curly brace. However, even in such cases, it is a feature of this technique that it can respond by conducting a lexical analysis alone while referring to the information in the syntax analysis from the source code of an answer.

System Integration

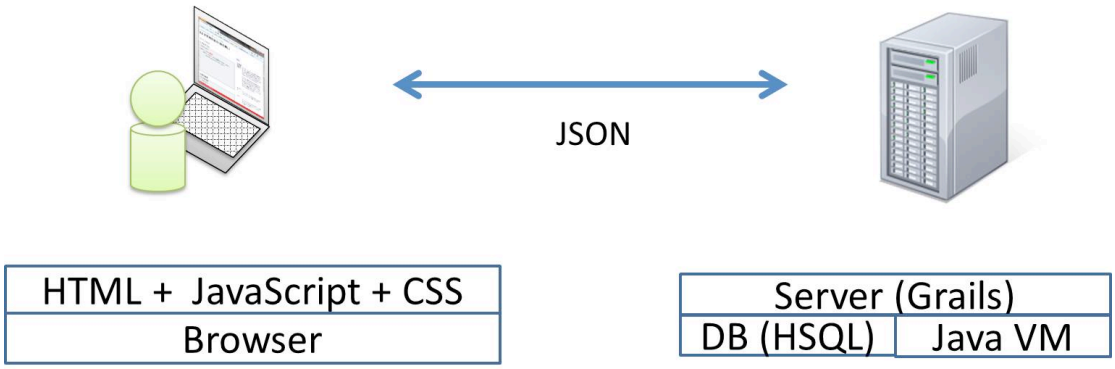


Figure 6: System integration

The system is outlined in Figure 6. The client's screen consists of HTML, JavaScript, and CSS. Further, the inputted data is transmitted to the server side as JSON data. The server was mainly developed using Grails.

Conclusion

In this research, when a student created a source code in accordance with a lecture video and an error was detected in the student's code, the system switched to a video that urges error correction. The error correction video supports 69 main types of student error. However, since the error correction video could reproduce only one error at a time, the system could not perform correspondences in cases where two or more errors occurred. Therefore, the system has now been improved so that it can correspond to two or more errors. Moreover, the system is ready for test employment in a real lecture setting, to acquire student data, and to be evaluated.

References

Dongsong, Z., Zhou, L., Briggs, R. O., Nunamaker Jr., J. F. (2006). Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. *Information & Management*, *43*(1), 15–27.

Tatsuyuki, T., Takashi, K., & Osamu, M. (2014). Development of a system to analyze students' keystroke sequence in programming education. Proceedings from *The Asian Conference on Society, Education & Technology 2015*, 185–192.