WHAT THE INSTRUCTOR CAN LEARN BY RECORDING STUDENTS' DESKTOP SCREENS IN ONLINE PROGRAMMING CLASSES

Keiichi Takahashi

Kindai University, Iizuka, Fukuoka, Japan

ABSTRACT

In programming courses, instructors need to know the status of students during exercise sessions because they often make mistakes while creating programs. This study analyses desktop screens recorded by third-year students in a web programming course at Kindai University in Japan, categorizes the problems, and proposes content and methodological improvements for learning through online lectures. The results of the experiment revealed that (a) the students watched the lecture video at half the screen size, (b) it was difficult to rewind the video, and (c) even though students faced difficulties in completing the task, they decided not to ask questions and worked on the same problem for more than 30 minutes. While most previous studies have used questionnaire-based surveys, this study demonstrates that instructors viewing recorded desktop screens of students can identify issues that students may not be aware of on their own.

KEYWORDS

Video-Based Learning, Typing Error, Faculty Development

1. INTRODUCTION

Online lectures allow students to take courses at any time from the comfort of their own homes. In contrast, some students have pointed out issues, such as difficulty in asking questions to the lecturers and feeling isolated due to the lack of communication among students [1][2][3][4]. In addition, while online classes have been reported to increase student distraction, depressive symptoms, and stress, the students' performances in face-to-face and online classes are nearly identical [5].

When considering teaching methods in online classes, students indicated that instructors might not be taking full advantage of the capabilities of the online digital tools [6]. A study has shown that the problems with the delivery of online lectures can be organized into the following five categories: pace, clarity, quality, media diversity, and congruence. Measures for each of these issues can be addressed based on the existing research [7].

For online programming classes, it is helpful for students to provide their recorded videos to the instructors for review [8][9]. In addition, it has been shown that Zoom's breakout rooms can be used to enable students in a variety of situations to pair-program and cooperatively and actively engage in problem-solving the learning activities [10]. Online programming lectures require more consideration than classes in which the instructor provides one-way explanations because the students' status is unknown.

As described above, various studies have been conducted on online lectures. The discovery of issues and the measurement of results in those studies are based on questionnaires. The discovery of issues and the measurement of results in those studies are based on questionnaires. Although questionnaires can collect a large amount of data for statistical validation at a low cost, it is difficult to identify problems that students themselves do not recognize as problems. This study aims to create opportunities for instructors to identify issues with the materials they provide and consider ways to improve them by recording and analysing the desktop screens of students in actual online programming classes. This paper demonstrates that although analysing video files requires specialized skills and a lot of time, there are discoveries that can only be made by analysing the videos.

The research questions are as follows:

RQ 1. How do students view lecture videos?
RQ 2. What kind of programming mistakes do students make in online courses?
RQ 3. How do students get rid of bugs in online programming courses?

2. METHOD

2.1. The Surveyed Lecture

The surveyed lecture was the 10th lecture of the course on applied programming with Ruby (a programming language) for third-year students at Kindai University in Japan. The lecture consisted of two consecutive sessions of one lecture session and one exercise session, each lasting 90 minutes. The students learned the basics of Ruby in the first year. In addition, they were taught various applied programming in the first to the eighth lecture of the course. In the 10th lecture, the students mastered the basics of Ruby on Rails (Rails) programming. Rails is one of the web application development frameworks that uses Ruby. Similar software development frameworks are used in the development of web applications. The author specifically chose this lecture because of expecting that programming classes at universities will increasingly include classes on software development using such frameworks.

In this lecture session, students understood the basic functions and operations of Rails by building a simple web application to manage bookmarks. Since all of the operating procedures were described in the lecture materials, they could complete the assignment by entering the program according to the instructor's explanation during the lecture.

2.2. Operating Procedures

Table 2 shows the operating procedures extracted from the lecture materials. The operating procedures consisted of 21 steps. In Rails, developers are required to create web applications according to the Model-View-Controller (MVC) pattern. Therefore, the students were required to write a program in each module of MVC. In this lecture, the learners needed to modify six different files to create a program. The number of characters and lines of the program that they wrote in each file are presented in the table.

2.3. Programming Exercise Environment

The instructor allowed the students build a Rails programming environment on their computers, but it is challenging to build a development environment, especially using MS-Windows.

Therefore, we used Amazon's AWS Cloud9 to deal with this problem. It runs on a web browser regardless of the operating system, thereby making it easy for the students to use.

2.4. Participants and Survey Procedures

Four participants (two males and two females) were randomly selected from the participants of this course in 2021. First, they started the Zoom application at the beginning of the lecture and recorded the entire desktop screen of their computers using Zoom. Then, the students watched the lecture videos on Google Classroom and referred to the lecture materials as needed. They could ask for help from the instructor via the Zoom chat function if they had any issues. Finally, when the students had finished watching the lecture videos and completing the assignments, they stopped the recording and submitted the video files generated by Zoom to the instructor. The authors watched the video files and analysed the type of mistakes and the time to correct the errors.

3. RESULTS

3.1. Working Time Per Participant

The working times recorded in the video files received from the four participants are shown in Table 1. The length of the lecture video was 77 minutes. The working time ratio in the table shows the ratio of each participant's working time to the length of the lecture video. The lecture material was originally designed to be presented in a lecture room for over 90 minutes. As mentioned in Section 2.1, the lecture duration was 180 minutes. Average working time of the students was 110 minutes, indicating that they had sufficient time to complete the assignment during the lecture.

Table 1. Working time and its ratio recorded in the video files received from the participants

Participant	Working time	Working time ratio
1	$\frac{187}{77}$	2.42
2		0.99
3	81	1.04
4	96	1.24
Mean	110	1.42

Since the students who created programs while watching the lecture videos needed to stop and rewind them several times, the larger the working time ratio, the more time it took to input and debug the program. The underline in the working time indicates that the viewing of the lecture video was not completed during the lecture time. The reasons are described in Section 3.2.3.

3.2. Mistakes of Each Participant

The analysis results of the files recorded by the four participants are shown in Table 2. "Kind" is the type of mistake, and "Time" is the debugging time (in minutes) for correcting the mistakes.
Students' mistakes in the survey were categorized into the following three categories: "T" implies typing errors, which are mistakes in the text editor or terminal; "S" denotes forgetting to save the file, which implies that the student put the program in the text editor and then completed

the next operation without saving the file, resulting in a different result from the lecture video; and "D" represents the wrong directory, which is a mistake that results in a different result from the lecture video because the file was edited by the student or the directory where the command was executed differently from the operating procedure. As described in Section 3.1, Participant 1

could not finish watching the lecture video within the allotted time; the grey area in the table indicates this.

Table 2. Operations and learning status during the lecture for each participant

Operations	Operating filename	# Chars	# Lines	Participant-1		Participant-2		Participant-3		Participant-4	
				Kind	Time	Kind	Time	Kind	Time	Kind	Time
1	index.html.erb	47	1			T	5.8				
2	routes.rb	40	1								
3	bookmarks_controller.rb	37	3	D	-						
4	new.html.erb	200	7			T	1.8			T	33.6
5	routes.rb	40	1								
6	bookmarks_controller.rb	48	3							T	0.7
7	bookmarks_controller.rb	133	5								
8	index.html.erb	196	7			T	0.8				
9	routes.rb	47	1			T	6.0			T	6.3
10	bookmarks_controller.rb	41	3			T	6.0			T	4.1
11	bookmarks_controller.rb	86	4								
12	index.html.erb	44	1								
13	routes.rb	41	1								
14	bookmarks_controller.rb	52	1								
15	show.html.erb	136	9			T	3.3	T	0.4		
16	index.html.erb	49	1								
17	routes.rb	46	1							T	8.6
18	bookmarks_controller.rb	52	3								
19	edit.html.erb	199	7					S	1.5		
20	routes.rb	45	1								
21	bookmarks_controller.rb	149	5					T	4.7		
	Total	1364	53		-		16.1		6.6		19.7

Figure 1 shows the number of times the students made mistakes and the debugging time. About 88% of the mistakes and around 98% of the debugging time were caused due to typing errors. The number of occurrences of "forgot to save" and "wrong directory" was one each. Debugging time was less than one minute for each of them; thus, it did not affect the students' learning.

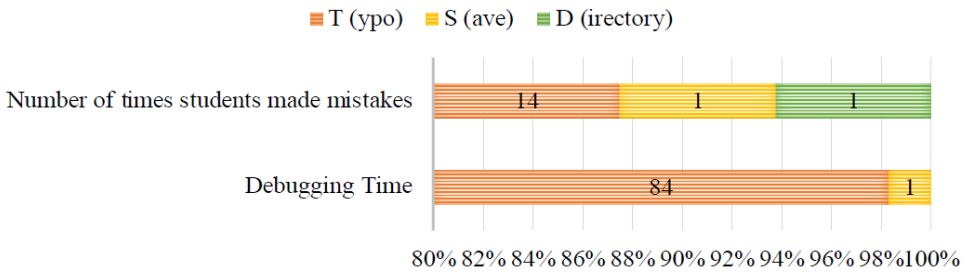

Figure 1. The number of times students made mistakes and debugging time for each type

3.2.1. Specific Examples of Typing Errors

In this section, typical examples of typos that frequently occurred are shown. The typos in this survey can be divided into three types.

Mistakes caused by Rails conventions. For example, when a student added the following to the file routes.rb, the student misspelled "bookmarks" as "bookmark" (participant 4, #17).

get 'bookmarks/:id/edit', to: 'bookmarks#edit'

In Rails, there is a convention that controller class names should basically be plural. Thus, noun plural identifiers appear frequently in Rails programs. Students who are unfamiliar with this convention tend to forget to enter the "s"; hence, the noun is incorrect.

Misspelling of homonyms. When the following program was added to routes.rb, some participants typed "destory" as "destroy" (participants 2 and 4, #9). Since the participants thought that they had typed it correctly, it was difficult for them to notice the spelling errors, and it took time to correct them.

delete 'bookmarks/:id', to: 'bookmarks#destroy'

Mistakes due to lack of understanding of Ruby syntax. In Rails development, Hash is often used as an argument for methods. There are several ways to describe Hash in Ruby, but the most concise notation is to add a colon after the key name. In the survey, a student forgot to add the colon after the url and action (participant 2, #4). If the students understand that the program they are typing is a Hash, they are unlikely to forget the colon.

<%= form_for @bookmark , url: {action: :create} do |f| %>

3.2.2. Specific Examples of Typing Errors

As shown in Table 2, participant 1 could not finish watching the lecture video. The reasons for this are as follows.

Immediately after executing the command to start the web server for testing, participant 1 accidentally closed the terminal window. The web server was left running, and the command to start a new web server could not be executed. The student continued to investigate the cause independently, and the operation procedure did not proceed.

4. DISCUSSION

4.1. RQ 1. How do Students View the Lecture Videos?

Since the lecture video was a single mp4 (MPEG Layer-4 Audio) file, it could be viewed at a higher playback speed. However, according to the recorded files of the participants, all of them watched the lecture video at a normal speed. This could be because they did not know when the operation explanation would be given during the lecture.

On the contrary, because the lecture video was a single file, the students often paused and input the program whenever the program related to the operation was shown in the lecture video. When the results differed from those in the lecture video, they rewound the video to check the

program. However, since the video file was 77 minutes long, it was fairly challenging to rewind it in a short time (Figure 2). They seemed so distracted by these operations that they were prone to typing errors.

Figure 2. Progress bar for student manipulation of video files on Google Classroom

As a workaround for this issue, we could prepare separate video files for each operation so that the students can concentrate on each task and input the program after watching the lecture video. In addition, a method of confirming that the program is correct should be added to the lecture materials so that they can proceed with the exercise with confidence.

4.2. RQ 2 What kind of Programming Mistakes do Students Make in online Courses?

Table 2 shows that there was no significant bias in the distribution of the error locations. It seems that the occurrence of the errors depends on the manner in which the participant approached the task rather than the operation procedure itself. For instance, participants 2 and 4 had more than five typos while participant 3 had two. Participant 2 seems to have preferred to write the program as fast as possible and let the computer find the mistakes and then fix them whereas Participant 4 seems to have preferred to create the programs steadily. It does not matter how many mistakes a student makes as long as they can correct the errors on their own.

As mentioned in Section 3.2.1, the most common error made by students in this survey was typing errors. Some of them are accidental while others are caused by preconceptions or ambiguous knowledge, such as “deveropment” and rails “db:mygreat.” Accidental errors are difficult to predict, but predicting systematic typing errors arising from an individual’s prior knowledge are even more challenging.

Moreover, in addition to forgetting the colon, as mentioned in Section 3.2.2, there were some cases of forgetting to enter periods and commas when entering programs, such as “<%= link_to book.title, book.url %>.” One of the causes of these errors seems to be related to the difficulty in viewing the lecture videos.

The instructor uses the entire desktop screen of the computer to provide explanations and records the screen to create a lecture video. Since the students need to simultaneously create programs while watching the video, the lecture video window and the AWS Cloud9 window are placed side by side on the desktop. Thus, it was difficult to distinguish periods and commas on the students’ screens because the lecture videos were displayed at a reduced size of less than half the screen.

This may be avoided by creating videos that are designed to be displayed in a reduced size on the students’ computer screens or by warning students in lecture materials about symbols that could result in typing errors.

4.3. RQ 3 How do Students Get Rid of Bugs in Online Programming Courses?

In this survey, 15 mistakes were made by the participants (Table 2). In one case, a participant asked the instructor for help, but in the other cases, the participants solved the problem by themselves. They had two strategies for solving the problems on their own. One was to rewind

the lecture video to confirm the procedure, and the other was to search the Internet for the error message displayed to find the solution.

Table 2 shows that the debugging time for the controller module (bookmarks_controller.rb) and the view module (*.erb) was about five minutes or less; hence, it was relatively quick to fix. This is probably because it was easy for the participants to notice the mistakes since there were often errors in the file that was modified just a few minutes prior in the operation procedure.

However, the errors in the routes.rb are difficult to fix because the error conditions vary widely depending on a slight difference in the mistake location. If the error cannot be resolved by the user, it is necessary to provide the correct solution file as soon as possible.
The operation that took the most time to modify was the fourth operation of participant 4 (33. 6 minutes). This procedure required seven lines of code to be entered into new.html.erb, the first line of which is shown below.

```
<%= form_for @bookmark , url: {action: :create} do |f| %>
```

Participant 4 forgot to type "=" in this program. If the "=" is inserted, HTML is the output, but if the "=" is missing, HTML is not the output, and no error message is displayed. It is difficult for the students to detect this error by debugging; thus, adding a warning to the lecture materials could reduce the number of errors.

In contrast, as mentioned in Section 3.2, some mistakes are difficult for the students to solve by themselves if they have little development experience. In this case, informing the students in advance that they should ask for help if they cannot solve a problem in 15 minutes may reduce the excessive trial and error by the students [11].

5. CONCLUSIONS

This study presents an experiment to analyse the recordings of students' desktop screens in an online programming class to confirm whether they learn as expected from the instructor's point of view and identify the learning problems.

The experiment results showed that approximately 90% of the errors were due to typing errors. These errors were caused by accidental mistakes or errors resulting from the person's prior knowledge. As a countermeasure, examples of typing errors could be added to the lecture materials in which the students often make mistakes. In addition, some errors were extremely difficult for the students to solve independently. Declaring a question rule to students before the lecture begins and advising them to consult the instructor if they cannot solve the problem within 15 minutes is expected to reduce the excessive struggle.

The results were observed while the students were working on their assignments, and it would be challenging for the learners to recognize these problems and respond quantitatively to the questionnaires. In addition, since there were only four participants in this experiment, other students' problems are unknown. However, considering that many of the issues shown in this study can be discovered by analysing the videos of only four students, the analysis in this study is considered effective as a method for examining the problems of several different online classes from the instructor's point of view. In the future, the findings of this study could be used to improve the lecture materials and lecture videos.

REFERENCES

[1] Mukhtar, K., Javed, K., Arooj, M., & Sethi, A., (2020) "Advantages, Limitations and Recommendations for online learning during COVID-19 pandemic era", *Pakistan Journal of Medical Sciences*, Vol.36 (COVID19-S4).

[2] Dutton, J., M. Dutton, & J. Perry, (2019) "How Do Online Students Differ from Lecture Students?", *Online Learning*, Vol.6, No.1.

[3] Takács, J. M. & M. Pogatsnik, (2021) "Online Learning from the Students' Perspective", *IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp.27–32.

[4] Firmansyah, R., Putri, D. M., Wicaksono, M. G. S., Putri, S. F., Widianto, A. A., & Palil, M. R., (2021) "Educational Transformation: An Evaluation of Online Learning Due To COVID-19", *International Journal of Emerging Technologies in Learning (IJET)*, Vol.16, No. 7, pp.61–76.

[5] R. M. G. García-Castelán, A. González-Nucamendi, V. Robledo-Rella, L. Neri, & J. Noguez, (2021) "Face-to-face vs. Online learning in Engineering Courses", *2021 IEEE Frontiers in Education Conference (FIE)*, pp.1–8.

[6] Teresa Fernandez-Bringas & Olga Bardales-Mendoza, (2020) "Overview of the Teaching and Learning on Virtual Education at University on a Worldwide Health Emergency (Covid19)", *The 4th International Conference on Education and E-Learning (ICEEL 2020)*, pp.80–85.

[7] Lange, C. & Costley, J, (2020) "Improving online video lectures: learning challenges created by media", *International Journal of Educational Technology in Higher Education*, Vol.17, No.16.

[8] Hassan, A. M., A. Dallal & M. A. S. Zaghloul, (2021) "Students' Perspectives on Online Lecture Delivery Methods for Programming Courses: A Survey-based Study during COVID-19", *2021 IEEE Frontiers in Education Conference (FIE)*, pp.1–5.

[9] Mcgowan, A. & P. Hanna, (2015) "How Video Lecture Capture Affects Student Engagement in a Higher Education Computer Programming Course: A Study of Attendance, Video Viewing Behaviors and Student Attitude", *eChallenges e-2015 Conference*, pp.1–8.

[10] Li, L., Xu, L. D., He, Y. M., He, W., Pribesh, S., Watson, S. M., & Major, D. A, (2021) "Facilitating online learning via zoom breakout room technology: A case of pair programming involving students with learning disabilities", *Communications of the Association for Information Systems*, Vol.48, No.12, pp.88–100.

[11] 15 min rule (Google AI team), https://twitter.com/math_rachel/status/764931533383749632, last accessed 2023-09-15.