Analysis of Fill-in-the-blank Questions Provided by Programming Education Support Tool pgtracer

Miyuki Murata
Faculty of Liberal Studies
National Institute of Technology, Kumamoto College
Yatsushiro, Japan
m-murata@kumamoto-nct.ac.jp

Naoko Kato
Faculty of Liberal Studies
National Institute of Technology, Ariake College
Omuta, Japan
naoko@ariake-nct.ac.jp

Youhei Nakayama
Graduate School of Information Science
Saga University
Saga, Japan
nakayama-youhei@is.saga-u.ac.jp

Tetsuro Kakeshita
Graduate School of Information Science
Saga University
Saga, Japan
kake@is.saga-u.ac.jp

***Abstract*—We have developed the programming education support tool pgtracer which provides fill-in-the-blank questions containing a C++ program and a trace table. In our previous studies, we assigned homework utilizing pgtracer to the students. We then investigated learning activities of the students and educational effect utilizing pgtracer. In this paper, we analyze the study log and the answer log collected by pgtracer. We analyze student activities and incorrect answers to find the tendency and frequent mistakes of the students. We expect that these results help to improve programming education through feedback to the class and the teacher.**

Keywords—Learning Analytics (LA); computer programming education; e-learning; Moodle; fill-in-the-blank question

I. INTRODUCTION

Computer programming is essential at national institute of technology and university majored in science and engineering. However, we often find students with low programming skill in an actual programming class. It is useful for a student to develop as many programs as possible in order to acquire practical programming skill. However, the students cannot practice enough in an actual class due to a limitation of teaching staff and time.

In order to cope with this problem, we are developing a programming education support tool pgtracer [1]. Pgtracer is developed as a Moodle plug-in so that a student can learn computer programming at any time and place. Pgtracer automatically evaluates student answer as soon as a student submit their answers. The automatic scoring function of pgtracer reduces the teacher's workload and can provide a quick response to the students. Pgtracer also provides data analysis functions from various viewpoints [2]. The functions also support teachers to recognize the learning activity and achievement of each student and the entire class.

We provided various programming homework using pgtracer to the student at our programming courses [3, 4]. The courses are provided for the first and second year students of the Institute of Technology, Kumamoto College. We have found the following facts through our experience. (1) The students who continuously worked on their homework understood computer programming better than those who prepared for the examination in a short time. (2) The understanding of the program becomes better for the students understanding the trace table better. (3) Student's programming skill affects their answering process to lead a correct answer.

In this paper, we analyze the incorrect answers and the answering processes of the students. We also discuss the tendency of the processes and incorrect answers.

There are various research contributions willing to support computer programming education. We shall introduce representative related researches in Section II and explain the difference from our research. Section III introduces pgtracer functions. We next explain the outline of the experiment in Section IV. In Section V, we shall analyze the collected data by pgtracer. The result and future work are discussed in the last section.

II. RELATED WORKS

Fu et al. analyze the error log of the programming and access log of the educational contents [5]. In order to support the students who study using on-line learning, Hering et al. and Gotthardt et al. analyze learning activities using SAS Business Analytics, and constructs a learning environment which provides valid educational content such as text and video [6, 7]. Malliarakis proposes a framework to guide the introduction of learning analytics mechanism using the game based learning analytics [8]. Ishiwada extracts frequent edit patterns of the students utilizing sequential pattern mining [9]. It automatically estimates the learning progress by analyzing the result and the learning activity. Igaki proposes the individual guidance support system called C3PV [10]. It stores the coding process and visualizes the process.

This research is supported by JSPS KAKENHI under grant number 17K01036.

978-1-7281-2627-2/19/$31.00 ©2019 IEEE
DOI 10.1109/IIAI-AAI.2019.00059

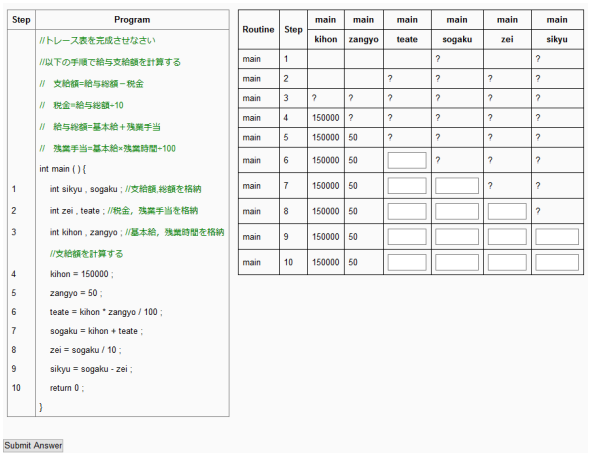

Fig. 1 Fill-in-the-Blank Question (2)-2 of pgtracer.

Pgtracer automatically collects learning log data such as student answer, score, required time to fill each blank, right answer ratio and required time to solve each question. Pgtracer also provides the data analysis functions to understand the learning activity of a student and an entire class, and to detect weak points of the students. The blanks can be defined at various program elements (e.g. single token, sequence of tokens, expression, and statement) and trace table (e.g. variable value, step number, and variable name). Thus the fill-in-the-blank questions provided by pgtracer has more flexibility and can express a wider level of difficulty than other existing programming education support tools.

III. Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions

pgtracer provides a fill-in-the-blank question composed by program and trace table (Fig. 1). A trace table represents the value of the variable and output. A fill-in-the-blank question is defined by four XML files, each of which represents a program, a trace table, a mask for the program or a mask for the trace table. A mask file defines blanks within a program or a trace table.

pgtracer provides functions to create a fill-in-the-blank question, functions to provide and evaluate a question, functions to collect and analyze a log.

1) Functions to Create a Fill-in-the-Blank Question [1]

A fill-in-the-blank question is composed of four XML files, representing either of a program, a trace table, masks for the program and the trace table. pgtracer provides the functions which automatically create these XML files. A token, consecutive sequence of tokens, expression, and statement are the candidate of blanks within a program. Variable value, step number, and variable name are the candidate of blanks within a trace table. We have clarified that the difficulty level of a question can be controlled by the place of a blank [1, 2]. The questions composed of the four types of XML files are stored in the question database.

2) Functions for Presenting a Fill-in-the-Blank Question and Automatic Evaluation of Student Answer [1]

TABLE 1
Information of Question (2)-2

Description	Calculate and print a salary allowance.
Educational Objective	Can trace assignment statements.
Difficulty Level	Easy
Place of Blank (# of blanks)	Within a trace table (5)

This function presents the list of fill-in-the-blank questions stored in the question database and presents the selected question by the student. The definition of the question contains various options. The options include question mode (self-learning mode or examination mode), show/hide of the correct answer after automatic scoring, show/hide of the analysis function to the students, the coloring of the corresponding step when a student selects a blank within a trace table. In the case of self-learning mode, pgtracer evaluates the answer by comparing with the correct answer just after filling of each blank. Here, if there are multiple correct answers, pgtracer cannot evaluate the student answer immediately. After the answer is submitted to pgtracer, pgtracer compiles and executes the filled program, and then compares the result with the correct trace table. Therefore tracer can evaluate the answer correctly even if a question has multiple right answers.

3) Functions to Collect Answer and Analyze Log [2]

Pgtracer collects a student answer, correct answer, required time just after filling to each blank. Pgtracer provides the learning history function, the analysis function of each student, the detailed analysis function of each problem, the detailed analysis function of each blank, the detail analysis function of the learning process of the select student.

Utilizing these functions, the students can check their own learning activity and the average and distribution of whole users. And a teacher can recognize the right answer ratio and required time each student or each question using a table or a graph.

IV. Experiment Plan

The experiment was performed for three classes of 123 students at the first academic year of Institute of Technology, Kumamoto College from October 2018 to February 2019. The students are learning computer programming at a common course named “Fundamentals of Programming I”. And they are not majored in computer science.

We assigned two or three fill-in-the-blank questions to the students as homework for each week. These questions are selected from the lecture contents of the corresponding week. The deadline for homework is the next lecture. There is no penalty when a student did not finish the homework.

We carried out two quizzes using pgtracer in the fifth and twelfth lectures at two classes (88 students). The quizzes contain three questions assigned as homework. The quizzes are executed in the examination mode and do not show the correct answer after the students submit their answers. We allowed the students to repeatedly solve each question.

TABLE 2
RIGHT ANSWER RATIO, AVERAGE SCORES AND MEDIAN OF REQUIRED TIME OF THE QUESTIONS

	Question	Right Answer ratio (%)	Average score (%)	Required Time (sec)	Question	Right Answer ratio (%)	Average score (%)	Required Time (sec)
Homework	(2)-1	94.7	95.0	190.7	(10)-1	87.1	93.2	111.8
	(2)-2	93.9	93.0	245.3	(10)-2	85.6	74.7	167.6
	(3)-1	92.4	89.7	297.8	(11)-1	84.8	73.1	877.5
	(3)-2	90.2	42.2	509.2	(11)-2	81.8	83.2	211.1
	(4)-1	89.4	80.1	295.9	(11)-3	81.1	73.5	297.8
	(4)-2	88.6	87.2	236.0	(12)-1	81.8	81.6	389.2
	(4)-3	87.1	51.0	327.9	(12)-2	81.1	80.6	255.7
	(5)-1	87.1	77.4	227.0	(12)-3	80.3	78.3	304.2
	(5)-2	84.8	70.5	364.4	(13)-1	79.5	97.0	146.7
	(5)-3	84.8	70.0	338.9	(13)-2	78.0	70.4	306.8
	(6)-1	84.1	94.6	123.3	(13)-3	78.8	64.2	325.8
	(6)-2	85.6	88.1	139.5	(14)-1	76.5	91.7	209.1
	(6)-3	84.8	86.1	110.8	(14)-2	74.2	97.5	133.5
	(7)-1	85.6	95.8	134.8	(14)-3	73.5	78.6	313.3
	(7)-2	85.6	76.3	289.0	(15)-1	64.4	69.7	281.4
	(9)-1	90.9	90.7	184.9	(15)-2	62.9	64.0	410.0
	(9)-2	88.6	89.6	288.8				
	(9)-3	88.6	81.7	230.1				
Quiz	(3)-1	100.0	79.7	135.1	(9)-1	100.0	87.1	118.5
	(4)-1	98.9	50.6	206.0	(10)-2	100.0	54.8	110.3
	(4)-3	93.2	25.6	159.9	(11)-2	100.0	73.9	94.1

We prepared the fill-in-the-blank questions according to the following policy.

- Define blanks at a value of a variable, a part of a statement and execution step to confirm that the students understand the lecture.
- Clarify the educational objective of each question.
- Try to control student's workload by preparing 2-3 questions per lecture and 3-5 blanks per question to facilitate continued use of pgtracer.

We prepared 34 questions through the 13 lectures. All of the questions set the same option. The questions are set in the self-learning mode. This is because, through our experience in the past, we found that evaluation of student answer just after the student fills a blank increase the student's intrinsic motivation. The analysis function is shown, and the coloring of the corresponding step when a student selects a blank within a trace table. The collect answer shows after the student submitted. For example, Table 1 represents information of question (2)-2 (Fig. 1) in the second week.

V. ANALYSIS OF STUDENT LOG

A. Analysis of the Problems

Table 2 represents the answer ratio, the average score and the required time at the first attempt. Most of the questions have an average score higher than 70%. Thus the questions are relatively easy for the students. We analyze the questions whose average score is less than 70%. Questions (3)-2, (4)-3, (13)-3, (15)-1 and (15)-2 are provided as a homework, and Questions (4)-1, (4)-3, (10)-2 are provided as quizzes. Except for Question (4)-3, all questions which have blanks within a program seems to be difficult for the student.

There is a negative correlation of -0.59 between the average score and the average of the required time. Thus we consider that the student who marks a higher score can answer in a short time.

B. Analysis of Incorrect Answer

Table 3 represents the blanks whose right answer ratio is less than 70%. The blanks contain in the question are provided as homework. All blanks are defined within a program. No. 1, 4, 6, 7, 8 are defined within an iterative statement. The blank No. 1 with the lowest score is defined within a nested for statement. Therefore, we found that a blank contained within the iterative statement is difficult for the students.

For the cases of No. 2 and 3, we consider that the students could not describe printf and getchar standard function calls. As for No. 5, it was the first time to fill in the variable definition part, so that students might be confused about what to answer.

Our research group considers that analyzing the wrong answer can measure the search domain of the student, that is, the degree of understanding of the program. Under this assumption, we classify the wrong answers into the following four cases.

(1) No answer

TABLE 3
BLANKS WHOSE RIGHT ANSWER RATIO AT FIRST TRIAL LESS THAN 70%

No	Question	Place of Blank Right answer is in [ ]	Right Answer Ratio (%)	Average # of Trials	Typical Wrong Answers. The number in () means the number of answers. If it is blank that mean one answer.
1	(15)-1	if(j< [num - i]){ within a for statement	37.3	5.3	No answer(25), num(5), i(2), 0, 4, 6, i++, num+i
2	(3)-2	printf([" クラス %c\n"]);	54.6	7.7	No answer(8), "%c" (2), " クラス%c"(2), クラス%c , クラス E, " クラス%s" 他
3	(5)-3	ch=[getchar];	60.0	4.0	No answer(6), 65(6), "Kumamoto", getcher, str, int, %d, etc..
4	(13)-3	}while([cnt_pos < 5]); within a do-while statement	60.0	4.7	cnt_pos>5(6), No answer(5), cnt_pos==5(2), cnt_pos<6, "%d int, cnt_neg==5, cnt_pos, cut_pos>5, etc..
5	(4)-3	[int hour;] within a variable definition statement	65.2	3.6	No answer(12), hour(5), 1(4), 1.5 9/hour, double hour; int hour, scanf("%d","hour"), int, inthoure;, inthour, scanf("%lf",&mail)
6	(11)-3	[cntPos ++]; within a if statemen	65.7	3.7	No answer(8), cntPos=num/num(2), and=ans+i, cntNeg=4, cntPos+1, cntPos+=cntPos, cntPos+i, cntPos= etc..
7	(15)-1	printf([“\n”]); with in a nested for statement	66.1	3.5	No answer(8), ""(3), "*"(2), "%d",i, "|n", *, ,&num, ans=5
8	(12)-3	pow = [pow * 2]; within a for satement	68.5	4.0	No answer(11), n^2 (3), 2, 2*n^, 22, 32, 4, i*j, j*i, n*j, , pow*j

(2) An answer that is presumed that the student does not understand the instruction, i.e. what to describe.

(3) An answer that is presumed that the student understands the instruction but does not understand how to describe the instruction.

(4) Right answer

We consider the case (1), no answer means that the student does not understand the instruction, the student has no idea lead to the correct answer. It also means that the student does not answer because of just looking to check the question.

In the program of the fill-in-the-blanks problem, we have explained the idea of the algorithm or the concrete algorithm using comment within the program. The case (2) is the case where it was judged based on the comment that the specific process to be performed by drilling the program could not be reproduced. It can be further classified according to whether the comment instruction is specific or not.

On the other hand, the case (3) is a case where it was judged that although concrete processing was understood based on the comment, it was not understood how to express it in C language. The case (3) is further classified according to the type of error, and the distribution and the reason why the solution is reached are examined. Table 4 represents the answer classifications for No. 1 blank in Table 3. By reflecting this result at the time of problem development, we consider that it is possible to create a question that reinforces the weak points of the students.

TABLE 4
CATEGORY OF INCORRECT ANSWERS OF THE BLANK NO. 1 IN TABEL 3

Answer	Category	# of Students
No answer	(1)	25
Constant such as 0, 4, 6	(2)	3
num	(3)	5
i	(3)	8
i++	(3)	12
num+i	(3)	1

We analyze the blank No.1 whose right answer ratio is the lowest in Table 3 in detail. Fig. 2 represents the distribution between the number of steps needed to answer and the average of the examinations for the students who marked a 100% score at question (15)-2. Here, the blank in question (15)-1 are defined within a program, and the number of blanks in the question is five. As shown in Fig. 2, most students answered with less than 20 steps, but the correlation coefficient is -0.31 between the number of steps required and the average of the examinations, which has a weak negative correlation.

Table 5 represents incorrect student answers entered during the answering process. We can observe from this table that the students reach the correct answer after trials and errors of various incorrect answers.

We consider the answering process for the two answers that had a large number of steps, such as 53 and 35. The students input the expression using variables num and i first, but thereafter, with a constant interval from 0 to 10 at short intervals. After that, the students thought for a while and entered an expression using variables num and i, and finally got to the correct answer. The difference in the number of

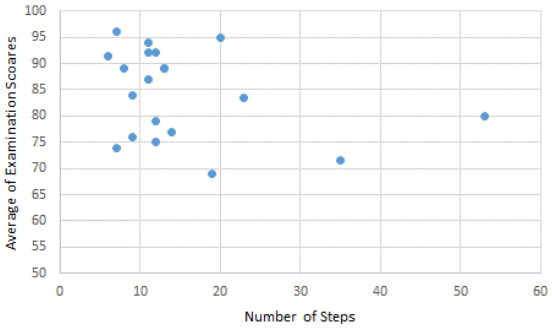

Fig. 2 Distribution between the Average Exam Score and Number of Steps during Answering Process (Score = 100%)

steps in the two answers depends on the number of times you enter the constant.

TABLE 5
NUMBER OF STUDENTS WHO FILLED INCORRECT ANSWER DURING ANSWERING ABOUT NO.1 IN TABLE 3

Incorrect Answer	# of Students	Total # of Students
Expression using i and num (i+num, i*num, etc.)	2	4
Expression using num (num, num-1, etc.)	14	32
Expression using i (i, i++, i-j, etc.)	8	16
Other expression (j, *5, 5-j etc.)	3	4
Constant（0, 2, 4, 5 etc.）	12	42
int	1	1
Invalid string (j]\)	1	1

From the above, we can observe that the students use num or i, the students try to input constant before they notice that num and i are used in combination. Also, after noticing that both num and i are used, it can be seen that the correct answer (num-i) can be easily derived.

C. *Required Time to Fill the First Blank*

The required time to fill the first blank is defined as the required time to fill the first blank that the student answered after the student started to answer a question. We can presume that the required time filling the first blank contains the time not only to answer the blank but also to understand the entire program. There is a weaker correlation between the required time filling the first blank and the average of examination because their correlation coefficient is -0.06.

Table 6 shows the average of the required time to fill the first blank of three groups categorized by the average score of the examinations. We find that the group with the highest score required a longer time than the group with the middle score. We consider that the students in the highest score group understand the entire program before the students start filling the blanks. We also find the lowest score group required the longest time. We consider that the students in the lowest score group needed a long time to understand the entire program and intention of the question.

TABLE 6
AVERAGE EXAMINATION SCORE AND REQUIRED TIME TO FILL THE FIRST BLANK

Average Exam Score	Required Time to Fill the First Blank	# of Students
High (More than 80)	58.6 sec	70
Middle (between 80 and 60)	49.6 sec	52
Low (Less than 60)	120.7 sec	6

Table 7 represents the average of the required time to fill the first blank of each place of blanks. In order to fill the blank within a trace table, the student needs to understand the entire program. Thus we presume that the student a required longer time to fill the blank within a trace table than the blank within a program. In addition, when there are blanks within both a program and a trace table, it takes the longest time. Thus we presume that it is difficult for students to understand the program in such cases.

TABLE 7
PLACE OF BLANKS AND REQUIRED TIME TO FILL THE FIRST BLANK

Place of Blank	Average Required Time to Fill the First Blank	# of Students
Program	44.6 sec	22
Trace table	47.5 sec	15
Program and Trace Table	50.5 sec	3

D. *Analysis of Answering Process*

Here we analyze the answering process of an individual students. If a student updates a blank during the answering process, we utilize the first-filled blanks. In the case that the blanks are defined only within a program and only within a trace table, we found that the student started filling the top blank. On the other hand, in the case that the blanks are defined within both of a program and a trace table, the first-filled blank was classified into either a program or a trace table, as shown in Table 8. Table 9 represents the place of the first answers and the average exam scores. We could not recognize the difference depending on the student preference.

TABLE 8
PLACE OF THE FIRST-FILLING BLANK (QUESTION HAS BLANKS WITHIN PROGRAM AND TRACE TABLE)

Question	Topic	Program	Trace Table
(6)-3	if statement	66	24
(7)-1	else-if statement	24	74
(9)-3	if statement	40	56

TABLE 9
PLACE OF THE FIRST-FILLED BLANK AND AVERAGE EXAMINATION SCORE

Preferred Place of the First-Filled Blank	Average Exam Score (%)	# of Students
Prefer program	77.8	50
Prefer trace table	81.8	48
Same preference	77.1	27

Figs. 3 and 4 show the student's answering process in question (15)-2. The horizontal axis represents the elapsed time and each color represents the filling of a blank in (15)-2. The question contains the explanation of the blanks, i.e. correct answer so that the two students achieve a 100% score of the question. Student 1 in Fig. 3 has an average score of 95.0% for examination, while Student 2 in Fig. 4 has a 75.0% score so that Student 1 has higher programming achievement. The order of both answers is the same, but we can observe that the time taken for the first blank answer is longer for student 1. It is considered that student 1 understood the problem before starting the first filling and then started to answer.

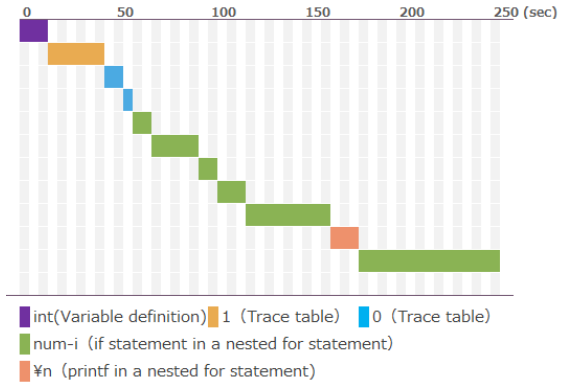

Fig. 3 Answering Process (Score=100%, Average Exam Score = 95.0%)

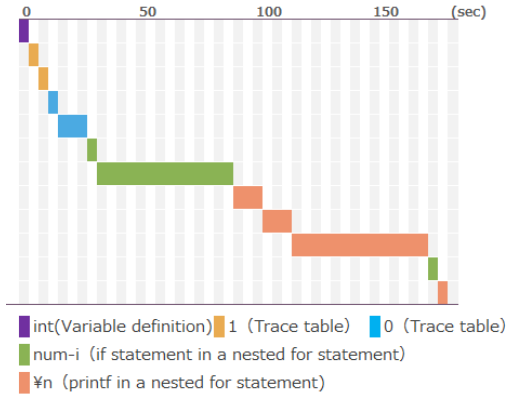

Fig. 4 Answering Process (Score=100%, Average Exam Score=75.0%)

E. Learning Activity of Students

The answer rate tends to decrease gradually. While the response rates of the two classes are 93.6% and 91.6% respectively, one class is significantly lower than the other classes at 59.0%.

In two classes with high answer rates, we announced to the students about homework and to give time to answer the homework using pgtracer. In classes with low answer rates, we did not do this, so we presume that student motivation decrease in the latter case.

VI. Conclusion and Future Work

In this paper, we assigned homework using pgtracer in a programming subject that was started in the first year of Institute of Technology, Kumamoto College, and analyzed the logs collected by pgtracer. As a result, we found that more difficult for the students to fill in the program than to fill in the trace table for the students, that the accuracy rate of the blanks in the program related to the iterative process is low, the students basically fill the blanks in the order from the top. It was found that the student having the high achievement of programming took more time to fill the first blank.

In this paper, we analyzed using the data collected in Programming Basic I class in the second half of 2018, but we would like to perform similar analysis using the data collected in the first half of 2018. In addition, we would like to consider not only the differences between the program and the trace table, but also the classification of places of blanks in more detail, and device the estimating method of understanding for each student. When conducting a detailed analysis, we expect that the R language can be used to find clusters of students with similar understandings, and also to find groups of fill-in-the-blank questions with similar student understandings.

References

[1] T. Kakeshita, R. Yanagita, K. Ohta, "Development and evaluation of programming education support tool pgtracer utilizing fill-in-the-blank question", *Journal of Information Processing: Computer and Education*, Vol. 2, No. 2, pp. 20-36, Oct. 2016. (in Japanese)

[2] T. Kakeshita, K. Ohta, "Student log analysis functions for web-based programming education support tool pgtracer", 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS2015), Brussels, Bergium, pp. 120-128, Dec. 2015.

[3] T. Kakeshita, M. Murata, "Application of Programming Education Support Tool pgtracer for Homework Assignment", International Journal of Learning Technologies and Learning Environments, Vol. 1, No. 1, pp. 40-61, 2018.

[4] M. Murata, T. Kakeshita, "Analysis method of student achievement level utilizing web-based programming education support tool pgtracer", 5th International Conference on Learning Technologies and Learning Environment (LTLE 2016), Kumamoto, Japan , pp. 316-321, July 2016.

[5] Fu, X. Shimada, A. Ogata, A. Taniguti,Y. Suehiro,D. "Real-time learning analytics for C programming language courses ", ACM International Conference Proceeding Series, pp. 280-288, 2017.

[6] Hering, W. Huppertz, H. Kramer, B. J. et al. "On benefits of interactive online learning in higher distance education: Case study in the context of programming education", eLmL - International Conference on Mobile, Hybrid, and On-line Learning 2014, pp. 57-62, 2014.

[7] Gotthardt, K. Kramer, B. J. Magenheim, j. Neugebauer, J. "On benefits of interactive online learning in higher distance education: Repeating a learning analytics project in the context of programming education", International Journal on Advances in Life Sciences Vol, 6, Issue 3-4, pp. 350-363. 2014.

[8] Malliarakis, C. Satratzimi, M. Xinogalos, S. "Integrating learning analytics in an educational MMORPG for computer programming", Proceedings - IEEE 14th International Conference on Advanced Learning Technologies, ICALT 2014, pp. 233-237. 2014.

[9] K. Ishiwada, Y. Morimoto, S. Nakamura at et al., "Development of a Method for Analyzing Source Code Editing Processes to Estimate Students' Learning Situations", IEICE Technical Report 116(438), pp.75-80. 2017(in Japanese).

[10] H. Igaki, S. Sato, A. Inoue at et al. , "Programming Process Visualization fo Supporing Studens in Programming Exercize", Transactions of Information Processing Society of Japan 54(1), 330-339, 2013. (in Japanese).