Utilizing Programming Education Support Tool pgtracer in an Actual Programming Course

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

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

***Abstract*— We developed a programming education support tool pgtracer. Pgtracer provides fill-in-the-blank questions to the students and collects student log to analyze student's learning process and understanding level. In this paper, we report our experience to utilize pgtracer at an actual programming course as homework assignment. We develop fill-in-the-blank questions corresponding to the course syllabus at each week. Student activities on pgtracer are analyzed to develop questions for the succeeding weeks. We also provide data to the instructor about the activities and achievement of the students for better collaboration between lecture and homework. We received positive feedbacks from both of the teacher interview and student survey about the usefulness of pgtracer as programming education support tool.**

Keywords—Learning Analytics (LA); computer programing 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. Recently, Japanese government announced that computer programming education begins at elementary school from 2020 in order to develop society members for the 21st century by fully utilizing computer.

However we often find students with low programming skill at an actual class. Many of them do not understand basic programming concepts such as loop, function and pointer. Individual support for such students is quite important.

We developed a programming education support tool pgtracer [1,2] to support teachers and students for the above purpose. Pgtracer utilizes fill-in-the-blank questions composed of program and trace table. The students fills the blanks. Then pgtracer automatically executes the filled program and compares the answer with the right answer. Pgtracer is developed as a Moodle plug-in so that the student can learn computer programming at any time and place as long as the internet connection and personal computer is provided.

Pgtracer utilizes fill-in-the-blank question as illustrated in Fig. 1. This means that students need not develop a computer program from the scratch. Pgtracer collects student log of filling the blanks so that teacher can analyze the collected log to check the activity and achievement of the individual student as well as those of the entire class. This is a typical application of the learning analytics.

Step	Program
	//キーボードから10個の整数を入力，その合計と平均を表示する．
	//平均は小数点第2位まで表示する
	# include < stdio.h >
	int main ( ) {
1	int i ; //繰り返しの制御用変数
2	int num , sum = ; //入力した数，合計を格納
3	avg ; //平均を格納
	//整数の入力と合計への加算
4	(i = 0 ; i < 10 ; i ++) {
4.1	scanf ("%d" , &num) ;
4.2	sum num ;
	}
	//平均の計算
5	avg = ( ) sum / 10 ;
	//結果の表示
6	printf ("合計 = %d\n" , sum) ;
7	printf ("平均 = %.2f\n" , avg) ;
8	return 0 ;
	}

Routine	Step	main i	main num	main sum	main avg	Output
main	1	?				
main	2	?	?	0		
main	3	?	?	0	?	
main	4		?	0	?	
main	4.1		11	0	?	
main	4.2		11	11	?	
main	4		11	11	?	
main	4.1		22	11	?	
main	4.2		22	33	?	
main	4		22	33	?	
main	4.1		33	33	?	
main	4.2		33	66	?	
(...)						
main	4	9	99	495	?	
main	4.1	9	110	495	?	
main	4.2	9	110	605	?	
main	5		110	605	60.5	
main	6		110	605	60.5	合計 605
main	7		110	605	60.5	平均 60.50
main	8		110	605	60.5	

Fig. 1 A Fill-in-the-Blank Question of pgtracer.

In this paper, we report our experience to utilize pgtracer at an actual programming course at Kumamoto national institute of technology. Fill-in-the-blank questions are utilized as homework assignment. Homework is required at each course for the students to achieve certain educational outcomes defined in the course syllabus.

Fill-in-the-blank questions are developed based on the course syllabus at each week. We monitor the student activity using pgtracer and provide feedbacks to the instructor. Feedback is also provided to improve the questions for the succeeding weeks.

Student activity is analyzed using the number of students to solve each question. Student achievement is analyzed using distributions of score and required time of each question and blank. We also interviewed the teacher and conducted a survey questionnaire to collect comments and opinions from the student. We selected some students and performed an interview for the students.

This research is supported by JSPS KAKENHI under grant number 16K01022 and 17K01036.

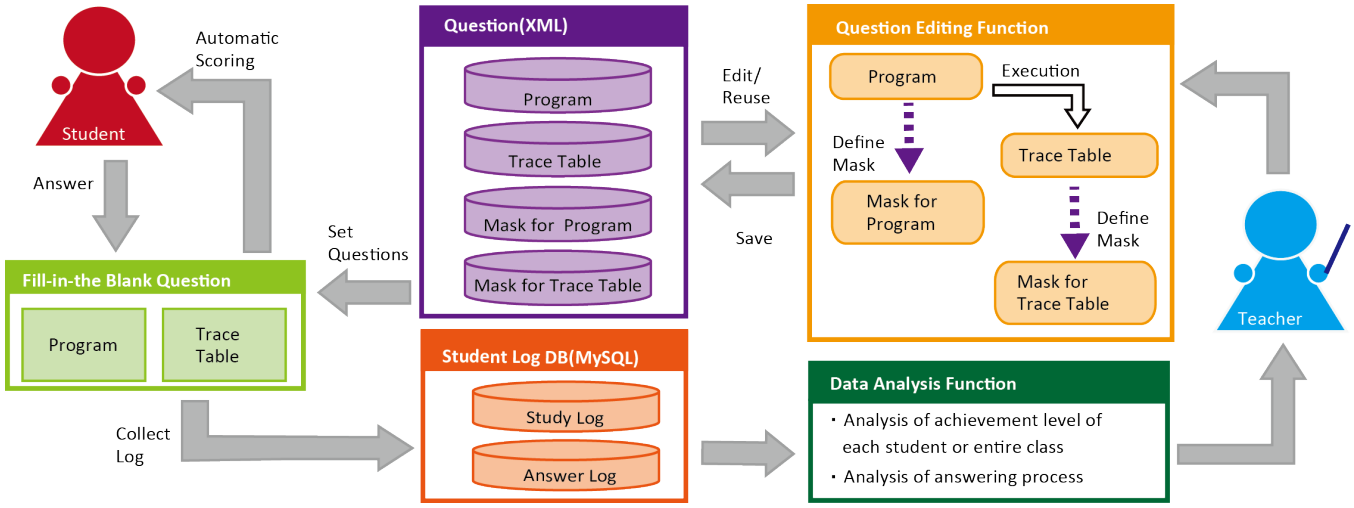

Fig. 2 Programing Education Process utilizing pgtracer.

This paper is organized as follows. The next section introduces the programming education model and workflow using pgtracer. We next explain the course planning in Section 3. Preparation policy of the fill-in-the-blank questions is explained in Section 4. In Section 5, we explain the analysis result of the student log. We report the analysis result to the instructor to improve the course and questions as explained in Section 6. We conducted a survey questionnaire and student interview as well as interview to the instructor. The results are provided and discussed in Section 7.

II. Programming Education Model using pgtracer

A fill-in-the-blank question of pgtracer is composed a pair of a C++ program and a trace table representing execution order of steps with the routine name, values of each variable and output of each step. A student fills the blanks such that the program and the trace table become consistent. Trace table is important for program comprehension and can help students when they get stuck during programming. It is important to visualize execution process of a program for a novice programmer. Thus we expect that a trace table is an effective means for programming education especially for beginners.

Fig. 2 represents the programming education process utilizing pgtracer. A fill-in-the-blank question is composed of a program, a trace table, a mask for the program and a mask for the trace table. They are described using XML. We separate a program and a mask for the program so that multiple masks with different difficulty levels can be defined for a single program. Trace table and the corresponding masks are separated for the same reason. Question DB contains valid combinations of the XML files.

When a student answers to a question, the system automatically evaluates the answer and feedbacks the score to the student. The student then can view the right answer. At the same time, pgtracer collects the log data of the answering process and the score.

The collected data is utilized to analyze the achievement level and the learning process of each student and the entire class. Pgtracer provides various analysis functions for the collected data. The instructor uses the analysis functions to improve the educational contents including fill-in-the-blank questions and the instruction to each student.

Pgtracer also provides functions to create and edit XML files representing a program, a trace table, a mask for program and a mask for trace table. Pgtracer automatically converted a program to the corresponding XML file. Then a teacher provides input data file to execute the program. Then pgtracer generates the XML file representing the corresponding trace table using the input data file. Next the teacher can create and edit program mask and trace table mask using pgtracer. The teacher can specify masks and hidden portion of the program and the trace table. Pgtracer then generates XML files representing the masks for program and trace table.

TABLE 1
Lecture Plan

First Semester		Second Semester	
Week	Contents	Week	Contents
1	Fundamentals of Computer	1	One Dimensional Array
2	Representation of Numeric Values	2	
3	Flowchart	3	Two-Dimensional Array
4	Constant, Variable, Assignment	4	
5	I/O (printf, scanf)	5	Pointer
6	Type and Operator	6	
7	Conditional Branch	7	Function
8	Mid-Term Examination	8	Mid-Term Examination
9	Conditional Branch	9	Function
10	for Statement	10	
11	while Statement	11	Variable Scope
12	do while statement	12	File
13	break, continue, switch	13	Struct
14	Exercise	14	
15	Examination	15	Examination

Thus a teacher can create and edit XML files without the knowledge of XML.

The teacher can define various options of the created questions. The option includes 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, coloring of the corresponding step when a student selects a blank within a trace table.

III. Course Planning

The experiment was performed for the 218 students at the second academic year of Kumamoto national institute of technology from October 2016 to February 2017. The students are majored in one of the following areas: mechanical engineering, electronics, civil engineering, architecture, bio technology and chemistry. They are learning computer programming using C language at the common course named "Fundamental of Computer Science". The course is composed of 30 weeks of 90 minutes classes each week. The course is provided by two teachers, one is giving lecture and exercise and the other, one of the author, is supporting the course by utilizing pgtracer for the department of biological and chemical systems engineering.

Table 1 represents course outline of "Fundamental of Computer Science". The experiment was performed during the second semester. We introduced pgtracer to the students at the end of June 2016 and registered the students to Moodle at the beginning of July 2016. We provided 9 questions including tutorial question and announced the student to utilize pgtracer.

IV. Preparing Fill-in-the-Blank Questions Using pgtracer

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

- Provide three questions for each of the 10 weeks in the second semester.
- Each question is usually based on the teaching contents of the corresponding week. We utilize the same or similar program taught at the corresponding week for the question. However we sometimes include questions of the previous weeks.
- The question is presented in the self-learning mode. Students can know whether their answers are correct or not just after their filling of each blank, since pgtracer instantly evaluates each blank just after a blank is filled in the self-learning mode.
- Although pgtracer supports program mask for an entire statement, we restrict program mask for a token or a part of an expression. This is because that the students are programming beginners.
- Trace table mask are defined for a set of consecutive cells of the same column. This is because that the value of a variable at the previous or next step provides hints to the students.

TABLE 2
Average Scores and Required Time of the Questions

Question	# of student	Average # of Trials	Question	# of student	Average # of Trials
(1)-1*	78	1.5	(6)-1	21	1.38
(1)-2*	73	1.63	(6)-2	19	1.21
(1)-3*	69	1.86	(6)-3	18	1.56
(2)-1	64	1.72	(7)-1	3	1.67
(2)-2	58	1.83	(7)-2	3	1.67
(2)-3	56	1.93	(7)-3	2	1.5
(3)-1	47	1.38	(8)-1	2	2
(3)-2	43	1.77	(8)-2	2	1.5
(3)-3	42	1.55	(8)-3	1	1
(4)-1	40	1.85	(9)-1	1	2
(4)-2	38	1.34	(9)-2	1	1
(4)-3	33	1.27	(9)-3	1	1
(5)-1	36	1.28	(10)-1	1	1
(5)-2	36	1.19	(10)-2	1	2
(5)-3*	36	1.17	(10)-3	1	1

* contains data of the test user.

- There are the cases that the same topic is taught for two weeks. Then more complex questions are assigned for the latter week which partially contain algorithm components.

We found during the experiment that the students tend to quit the exercise when the width of the trace table is too wide to be displayed on the computer screen. Considering this, we improve the questions such that the program and trace table can be displayed within the screen by adjusting the number of array element, dividing a long comment into two or more lines and by replacing long names of a variable or a function with shorter ones.

The questions are developed by the two authors. One creates the questions as a supporting teacher of the target course and recognizes the actual teaching contents and progress of the course. The other author reviews the created questions based on the validity of the place of the blanks, difficulty level and consistency between comments and source code.

We developed 30 questions for 10 weeks. The same option is used for the questions of the same week. The common options throughout the all questions are self-learning mode, show the analysis function and use coloring of the step corresponding to the current blank in the trace table. We basically show the correct answer after the students fill a blank. But we hide the correct answer to the students for the questions on weeks 4 and 9.

The total number of blanks within a question is between 6 and 14. The average number of blanks is 10. The average time to create three questions for a particular week is about 2 hours. We spent 1 hour to write the target programs and 1 hour to register and edit the program and trace table masks using pgtracer.

V. Analysis of Student Achievement

Table 2 represents the number of student answers and the average number of trials of each student. The question number (n) represents the corresponding week of the second semester. The largest number of students answer questions (1)-1 to (1)-3 prepared for week 1. However the number is gradually decreasing for the succeeding weeks. This represents the decrease of the student motivation.

The number of student answers suddenly drop between weeks 7 and 10. This is because that the contents of weeks 7 to 10 are not included in the mid-term examination. It can be clearly observed that the inclusion of the learning contents to the examination is quite important to keep the student's motivation at a high level.

Fig. 3 represents the score distribution of question (2)-3. The highest peak is between 90 and 100%. The second peak is between 0 and 10. This tendency is common among most of the questions. This implies that the question is not difficult.

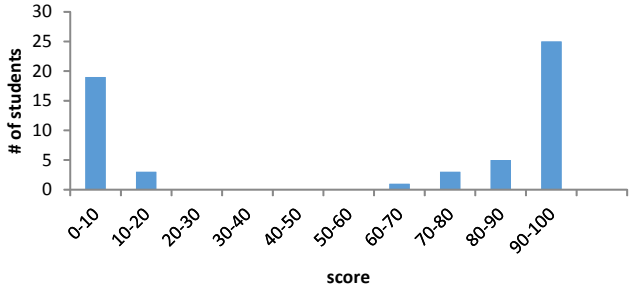

Fig. 3 Score Distribution of Question (2)-3

Fig. 4 represents the relationship between score and required time of the same question as Fig. 3. The required time of the answers with low score tend to be short. This implies that the students only view the question during preparation of the examination and did not actually solve the question.

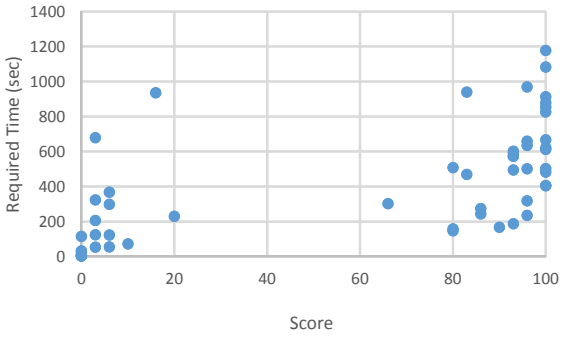

Fig. 4 Relationship between Score and Required Time of Question (2)-3

Fig. 5 illustrates the relationship between average score and average number of trials of the questions. The blue points represent the questions where the right answer is shown after the evaluation, while the orange points represent the questions where the right answer is not shown. The correlation coefficient between average score and average number of trials is -057. The number of students with 100% score increase for the questions with higher average score. Students quit the trial when they get the 100% score so that the average number of trial tends to be higher for the questions with low average score.

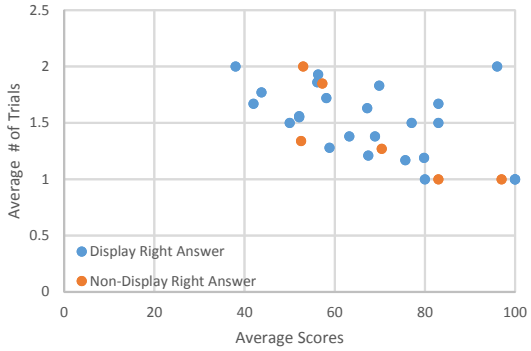

Fig. 5 Distribution of Average Scores and Average # of Trials

Although we expected that the number of trials is different between the two cases to show or hide the right answer, there is no significant difference. This implies that the students try to repeat the trials until they got the 100% score. However we observed a significant difference between the scores of the first trial and the maximum score. In the case to show the right answer, the difference is 20.2, while the difference is 6.7 for the case to hide the right answer.

Table 3 represents the difference between the examination scores of weeks 8 and 15 classified by the number of solved questions of the exercise. We can observe that the examination score of the week 15 is improved for the students who solved more questions during the experiment. This is an evidence that pgtracer is useful to improve programming skill of the students.

TABLE 3
Relationship between # of Solved Questions and Examination Score

# of Solved Questions	Average Difference of Exam Score
18	+2.47
1 ～ 9	+0.89
0	−0.18

VI. Feedback to the Instructor

We provide 3 fill-in-the-blank questions each week during the experiment and monitor the students' behavior. At first we provide the questions as a self-learning contents, which means that the pgtracer score does not affect the evaluation of the subject. Then the number of students was 37 and the total number of trials was 154.

We then notifies the above situation to the instructor and the instructor announced to the students that the pgtracer questions from (1)-1 to (6)-3 are utilized in the mid-term examination. Then the number of students and the total number of trials increase to 78 and 803 respectively.

We are expecting to utilize these experience in order to effectively control the student's behavior in the future classes.

VII. Overall Evaluation

A. *Analysis of Questionaire to the Students*

We conducted a questionnaire to the students after the experiment. Table 4 represents the questions to the students. The number of answers was 118 (92.2% of the enrolled students).

Consider questions (1) and (2). The students take 2 to 3 subjects other than the current subject (Fundamental of Computer Science) which assign homework to the students. Table 5 shows such subjects for each department of the students. The bubble chart illustrated in Fig. 6 represents the relationship between the spent time for the current subject and other subjects per week. The numbers within the bubbles represents the number of students corresponding to the answer represented by the x-y coordinate of the chart.

We can observe that 53 students (44.9%) spend less than 30 minutes per week for the programming exercise of the current subject. However there are 4 students (3.39%) who require more than 3 hours. Required time to solve the homework greatly differ depending on the students.

For question (3), 32.2% of the students solve more than 9 questions. Fig. 7 illustrates the reason of not solving more than 9 questions collected from the remaining 67.8% of the students.

The most popular reason is that the students did not have enough time. Typical students are taking other subjects listed in Table 5 as well as other school activity such as club activity and student council activity. Also considering the reason that pgtracer score do not affect evaluation, we need to provide some kind of incentives to motivate students to work on pgtracer exercise.

Considering the reason from 15 students that the student did not understand how to use pgtracer, more explanation and instruction are required at the initial stage to introduce pgtracer. We received the comments in question (8) that they did not understand the concept of trace table. Although we explained and demonstrated how to understand trace table, most of the students are not familiar with the concept of trace table within the conventional computer programming education. Some students did not understand that the trace table represents the execution order of the steps within a program and the value of the variables at each step.

On the other hand, 10 students answered that they already understand computer programming. More difficult problems should also be provided to motivate such type of students having sufficient programming fundamentals.

Next we shall consider questions (4) and (5). As illustrated in Fig. 8, 60% of the students answered that the pgtracer questions are easy or fair. This means that the provided questions were not too difficult and are at the appropriate level as a self-learning exercise.

We also find that 75% of the students answered that pgtracer is useful to learn computer programming as illustrated in Fig. 9.

We received some student comments collected from question (8).

TABLE 4
Questions to the Students

(1)	Average time to answer the questions per week
(2)	Average time to work on other exercise per week
(3)	Did you solve more than 9 questions from question (1)-1 to (6)-3 ?
(4)	If the answer to question (3) is yes, how was the difficulty level of the questions?
(5)	If the answer to question (3) is yes, was the exercise useful to learn computer programming?
(6)	If the answer to question (3) is no, select reason from the provided list.
(7)	If you select “others” in question (6), please specify the reason.
(8)	Provide comment and/or suggestion to improve pgtracer if you have some.

TABLE 5
Other Subjects with Student Assignment

Department	Subject Name
Mechanical and Intelligent Systems Engineering	• Introduction to Micro Computer Programming • Engineering Drawing II • Manufacturing Practice II
Architecture and Civil Engineering	• Introduction to Micro Computer Programming • Surveying and Surveying Practice II • Drawing and Design I
Biological and Chemical Systems Engineering	• Introduction to Micro Computer Programming • Experiment for Bioengineering

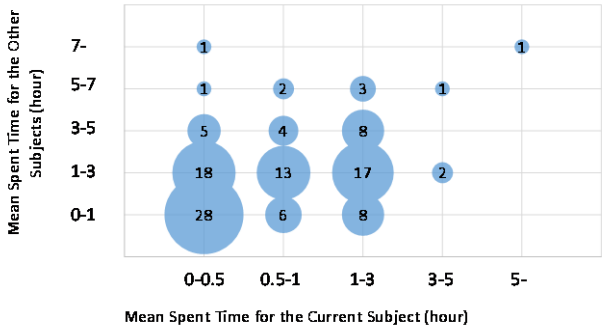

Fig. 6 Relationship between the Spent Time per Week for the Current Subject and Other Subjects

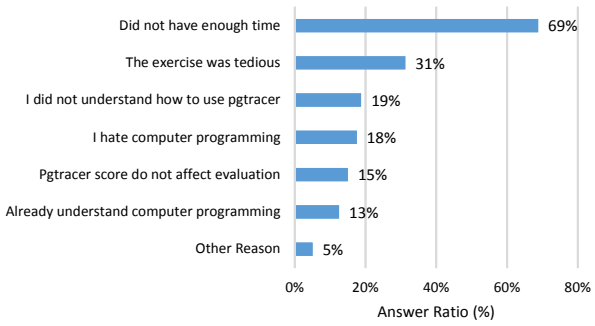

Fig. 7 Reasons of not Solving More than 9 Questions

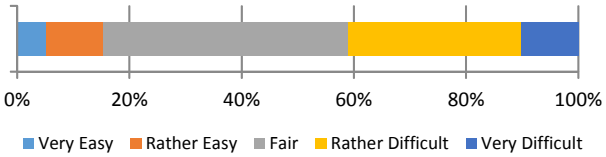

Fig. 8 Difficulty Level of the Questions

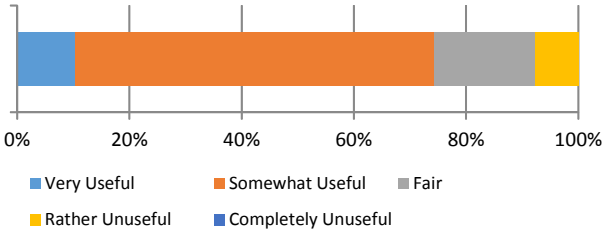

Fig. 9 Is pgtracer useful to learn computer programming?

The current pgtracer cannot show the entire columns of the trace table when the trace table is too wide. Although the entire columns can be displayed by shrinking the contents, the shrunk contents can be unreadable for the students. As a result, we sometimes find unfilled blanks which are not displayed on a web browser. We have notified this to the students but some of the students missed the notice. It is important to keep the width of the trace table small such that the entire columns of the trace table are displayed without shrinking the contents. This requires a careful selection of the displayed/hidden columns and the order of the displayed columns.

Some students provide comment that pgtracer should be utilized within the lecture. More collaboration is required between the lecture and homework to facilitate self-learning.

B. Student Interview

We also interviewed the students who solved more questions than average. They said that fill-in-the-blank question is easy to solve than creating a program from the scratch. The self-learning mode quickly provides correctness judgement after filling a single blank. Such function is evaluated favorably so that we can conclude that the self-learning mode is an effective means to facilitate students' motivation. The students said that they can clearly understand basic grammar and behavior of the program. Such positive effect can be expected to the programming education using pgtracer.

Some students respond that they understand how to develop correct algorithm by reading comments associated to the program described according to our programming guidelines. Reading a good program is an effective means to learn computer programming. Pgtracer can facilitate careful reading of such programs by providing fill-in-the-blank question of the trace table.

Some of the students said that they solved the problems through discussion with their friends. We can expect to facilitate group learning or LTD (learning through discussion) using pgtracer.

We have sent e-mails to the students each time we provided new problems. Some students said that such messages are useful for them to continue motivation of learning.

C. Instructor Interview

We also interviewed the primary teacher of the target class and obtained the following comments.

- Since pgtracer allows to define various types of blanks depending on the understanding level of the students, I can create wide range of problems which can cover both programming beginners and experienced students.
- Since pgtracer shows skeleton of the program so that it is suitable to teach good programming skill.
- Pgtracer is suitable to teach execution flow of a program.
- Difficulty level of the exercise is appropriate since basic problem and relatively difficult problem are both provided.
- Most of the students using pgtracer achieved the highest level score at the final examination. Some of these students achieved 10% more score compared with the first semester.
- It is recommended to improve pgtracer to visualize behavior of the program. Then the understanding level of the students will be improved.

These comments are essentially the same as our expectation to develop pgtracer.

VIII. Conclusion and Future Work

In this paper, we conducted an self-learning experiment using pgtracer at an actual programming course. We observed certain skill up of the students. Both of the students and the primary teacher of the course told that pgtracer is useful for programming education. Pgtracer questions can be stored to the DB for future reuse. Main obstacle to use pgtracer is the unfamiliarity to the trace table from the viewpoint of the students.

As a future work, we are planning to investigate effective means to teach trace table to the students. Appropriate incentive mechanism will also be investigated through PDCA cycle illustrated in Fig. 1.

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] 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.