Proceedings of the Federated Conference on Computer Science and Information Systems pp. 609–618
DOI: 10.15439/2020F44
ISSN 2300-5963 ACSIS, Vol. 21

Comprehension analysis considering programming thinking ability using code puzzle

Hiroki Ito
Ritsumeikan University
Graduate School of Information Science and Engineering
Email: hirokiito6900@de.is.ritsumei.ac.jp

Hiromitsu Shimakawa
Ritsumeikan University
College of Information Science and Engineering
Email: simakawa@cs.ritsumei.ac.jp

Fumiko Harada
Connect Dot Ltd.
Email: harada@de.is.ritsumei.ac.jp

***Abstract*—In programming education, the instructor tries to find out the learners who needs help by grasping the learners' development of understanding using tests that require knowledge. However, in reality, not many learners will acquire the skill of writing source codes. This kind of current situation implies that programming ability of learners cannot be measured by tests that require knowledge. This paper focuses on not only the knowledge items required for programming but also the programming thinking (computational thinking), which is the ability to combine the constituent elements of the program. In this paper, we propose a method to estimate the learner's understanding from the learner's process to solve the code puzzles that require programming thinking as well as knowledge. We developed the interface to realize the proposed method. The experimental result with the interface showed that the proposed method could estimate with the accuracy of 80% or more.**

I. Introduction

IN programming education for beginners, there are many learners who cannot create correct sources in spite of passing written tests that ask their knowledge about grammar and how to express algorithms. The programming ability cannot be measured only by verifying only learner's knowledge. This is because programming skill also requires the ability to construct program elements logically with a perspective.[1][2] Programming thinking is the ability to assemble the components of a program with a perspective. The method of measuring programming thinking ability from the requirements to be satisfied has not been established yet. In the actual situation, the only way to verify the true programming ability of a learner, which consists of both knowledge and programming thinking ability, is for the instructor to stand next to the learner and watch the answer. However, in a large-class lecture, it takes too much time to check the understanding of all students in this way. Most of the current educational settings use measurement methods that are biased in terms of knowledge because it is easy to grasp the understanding situation. As the result, many learners will not be able to understand the intention of the task and to acquire the ability to realize it. From such the situation, there is a demand for a method that can easily estimate the learners' understanding situation by considering programming thinking ability.

II. Programming education support

A. Current programming education support

This paper discusses an understanding analysis focusing on programming thinking ability. Programming thinking ability means "what kind of combination of movements is necessary to realize a series of activities intended by oneself, and how to combine symbols corresponding to each movement. And the ability to logically consider how to improve the combination of symbols to get closer to the intended activity. "[3]

Although programming education for beginners is conducted in educational institutions such as universities and newcomer education at companies, many people cannot actually program even if they can pass a written test that asks knowledge. This indicates that programming cannot be achieved by knowledge alone, and is thought to be due to the lack of programming thinking as mentioned above as pointed out by the Ministry of Education, Culture, Sports, Science and Technology in Japan[3].

This paper defines the learning item achievement level as the degree of acquirement of the knowledge given in a lecture, materials, and so on. It is a contrasting skill of the programming thinking ability. Most current programming education support focuses on the learning item achievement level. Therefore, there is an urgent need to establish a learning support method that considers programming thinking.

B. Programming learning by a code puzzle

This research uses a code puzzle as a learning interface. A code puzzle is a rearrangement problem where the learner rearranges code fragments such as source code and pseudo code and assembles them to perform appropriate processing. The code puzzle is inspired by Parson's Programming Puzzle proposed by Parson *et al.*[4] Parson *et al.* said that, for beginners, code puzzles are more effective and better at nurturing logical thinking than full-coding . Moreover, code puzzles present the logic flow unlike the blank filling problem. Code puzzles simplifies to acquire a feature of the learner's learning behavior used for estimation of comprehension from the actions of selecting, moving, and rearranging blocks.

C. Schema for programming tasks

Schema is a term in cognitive psychology. Schema in cognitive psychology refers to the relationship between thought patterns and knowledge for problem solving in human long-term memory.[5]

When a person solves a problem of programming, he or she reads the problem and sets a path for problem solving. At this time, if he/she has no necessary knowledge and thought pattern, it is not possible for him/her to set the path for problem solving. In other words, we can consider that humans combine knowledge to solve problems. In the process of combining knowledge, humans form patterns of knowledge and thinking as a schema. The pattern of knowledge or thinking here is exactly the programming thinking. In this research, we consider to estimate the degree of schema construction from the combination of knowledge and logic based on the schema theory.

D. Related works

The method proposed by Jadud *et al.*[6] is an early study that identifies learners who need guidance using compilation errors. However, this focuses only on the error, and it is not possible to measure why the error occurred and how much the learner understood.

Mysore et al[7] proposed a Web system Porta that can identify the part where the learner is struggling. The comprehension factor is inherently intricately intertwined. However, Porta estimates the comprehension level only by focusing on the fact that the learner takes time. Therefore, the ability and growth of learners are not taken into consideration.

Guo *et al.*[8] proposed an interface that supports one-to-many programming learning in real time and its implementation Codeopticon. However, Codeopticon depends on the quality of the instructor and forces a heavy burden on the instructor.

Asai *et al.*[9] identified the cognitive load on learners and the factors that caused the cognitive load by the blank filling problem. However, the blank-filling problem prevents the flow of logic. It cannot be said that it considers programming thinking.

Many of these existing researches focus only on the aspect of knowledge and do not estimate the degree of understanding based on logic(programming thinking).

As a study focusing on behavior, Ihantola *et al.*[10] estimated the difficulty level of a task using a decision tree from the answering process such as answering time and keystroke of a programming task. They suggested that the answer process brings significant difference in learner's comprehension. However, they did not mention programming thinking ability and cannot estimate factors of misunderstanding.

As one of the traditional programming education formats, there is Parson's programming puzzle proposed by Parsons *et al.* This is introduced as a tool that is easy for beginners to work on. Parson *et al.* asserted that code puzzles can identify specific points and errors that the learner has stumbled. They argued that, since the code puzzle answer is a well model answer, learners can relive good programming practice. However, although this tool focuses on the acquisition of programming thinking ability, it does not estimate the degree of comprehension.

III. Educational support using behavior when answering code puzzles

A. Educational support focusing on programming thinking

The purpose of this study is to provide novel education support method focusing on not only the learning item achievement level as shown in ordinary education support but also the ability to assemble those learning items and to realize the intended program (called Programming thinking). The learning item achievement level in this paper is defined as the level of knowledge necessary to solve programming problems. On the other hand, programming thinking ability is defined as the ability to combine the knowledge to design deliverables that match the programming task. The learning item achievement can be measured easily by paper-based or Web-based tests. However, programming thinking ability cannot be measured without observing the programming process of the learner, which is the learner's behavior. This is because programming thinking occurs in the process of creating a program. A learner who can perform programming thinking will answer a programming problem with prospecting and constructing the logic flow toward the answer. Therefore, it is not possible to judge whether the learner has acquired programming thinking or not unless the learner's behavior in the answering process is investigated.

It has been difficult to measure programming thinking ability unless the instructor watches the learner's answering process to the programming task. Code puzzles are a better tool for learners to focus on combining knowledge. The method proposed in this research estimates such programming thinking ability by analyzing the process in which a learner solves a code puzzle. The outline of the educational support method aimed at in this research is shown in Figure 1. At first, the learners solve the code puzzle. The interface used for the code puzzle is the original application that runs on the WEB. This application collects the operation histories of the learners. These operation histories will differ depending on the learner. Based on this hypothesis, the understanding levels of the learner are classified by machine learning based on the collected operation histories. A machine learning model that can interpret the reason for classification is adopted. The reason feedbacks to the learner and instructor.

B. Collecting learner behavior using code puzzles

The method proposed in this study uses the tools shown in Figure 2 and 3 to collect learner's characteristic behaviors to measure the understanding. The test subjects are Japanese, so the content is displayed in Japanese. In this research, the notation of the program is based on PAD proposed by Futamura *et al.*[11]

In the proposed method, an exemplary program or source code that satisfies all the requirements given in the task is

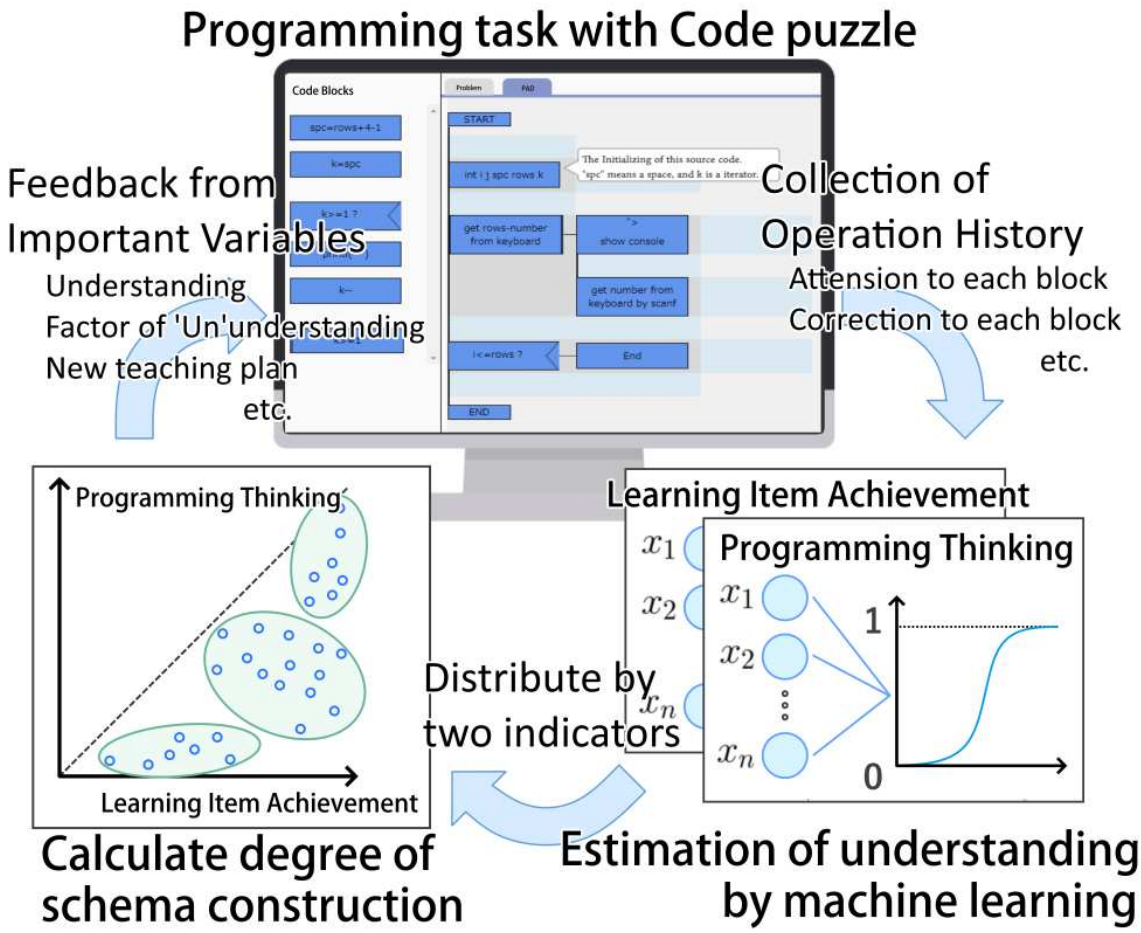

Fig. 1. Schematic diagram of the proposed method

divided into code fragments or pseudo code with functional cohesion. A code fragment generated by the division is defined as a block in this paper.

As shown in Figure 2 and 3, blocks for assembling the program are displayed for a task. The learner assembles the program using the blocks so that all the given requirements are satisfied. This can be regarded as a code puzzle that considers the arrangement of blocks so that the constraints are satisfied.

The proposed method examines the learners' thinking processes during solving the code puzzle in order to judge whether the learners have acquired programming thinking or not. When the learners answer with a perspective of how to combine the blocks toward satisfying the given requirements, they will place the blocks in the correct position without hesitation. The learners without such a perspective will wonder where to place the blocks. The learners who incorrectly interpreted the meaning of the problem sentence and/or given requirements will quickly move the blocks to wrong positions. In this way, it is assumed that the learners' thinking processes during answering appears in the behavior of moving blocks. From this idea, the proposed method analyzes the behavior of each learner collected by the tool during the learner is placing blocks.

The learner can switch between the task sentence screen(2) and the drawing screen(3) from the tab at the top of the tool. The learner interprets the problem from the task sentence as shown in Figure 2, selects suitable blocks from the blocks displayed on the left side of the tool as shown in Figure 3, and assembles them by drag and drop. Some of the blocks contain blanks. Blocks with blanks increase the degree of freedom in expressing procedures and are intended to cause the learner to get lost. Additionally, if the learner hovers the mouse over a block, a description of that block is displayed. The learner finishes the answer by pressing the submit button when they think they has answered it correctly.

C. Explanatory variables collected by the tool

The tool analyzes the relationship between the learners' behaviors during answering and their programming ability through machine learning models. In this research, the behavior of the learner is the explanatory variables and the understanding of the learner is the response variables. Before applying this method, we consider what the explanatory variables correspond to the learner's thinking so that the learner's

Fig. 2. Question sentence screen

incomprehensible factors can be investigated. The explanatory variables used in the proposed method are roughly divided into three.

- Attention to each block
  - First time of touching each block
  - Time spent watching at each block description by hover
  - The number of times each block has been dragged and dropped
  - Drag and drop frequency
  - Frequency of drag and drop in each quarter of the answering time
  - Dispersion of time watching the description
  - Dispersion of drag and drop number
- Correctness in placing blocks
  - Correctness or incorrectness of input to each blank
  - Number of times each blank correction failed
  - How close the whole is to the model answer (editing distance)
  - Number of times overall blank correction failed
- Time spent
  - Total task time
  - Ratio of drawing screen display time for task sentence screen display time
  - Number of times to switch between the task sentence screen and the drawing screen

The attention level and correctness of each block reveal the block that confuses the learner. The overall correctness reveals how accurately the blocks are combined. In addition, the time and ratio spent on drawing using PAD indicate how much the learner got confused during drawing.

D. Estimating the understanding for each learner

In this method, the understanding is estimated based on the behavior of the learner collected by the tool. It is estimated using the random forest method and logistic regression as machine learning methods. Since the two methods can calculate the importance of variables, it is possible to consider classification factors. Although the random forest method is more accurate, the probability of classification can be known by logistic regression, and this classification probability can be grasped as an understanding level of 0.0 to 1.0.

In the proposed method, the understanding is estimated based on the behavior of the learner collected by the tool. It is estimated using the random forest method and logistic

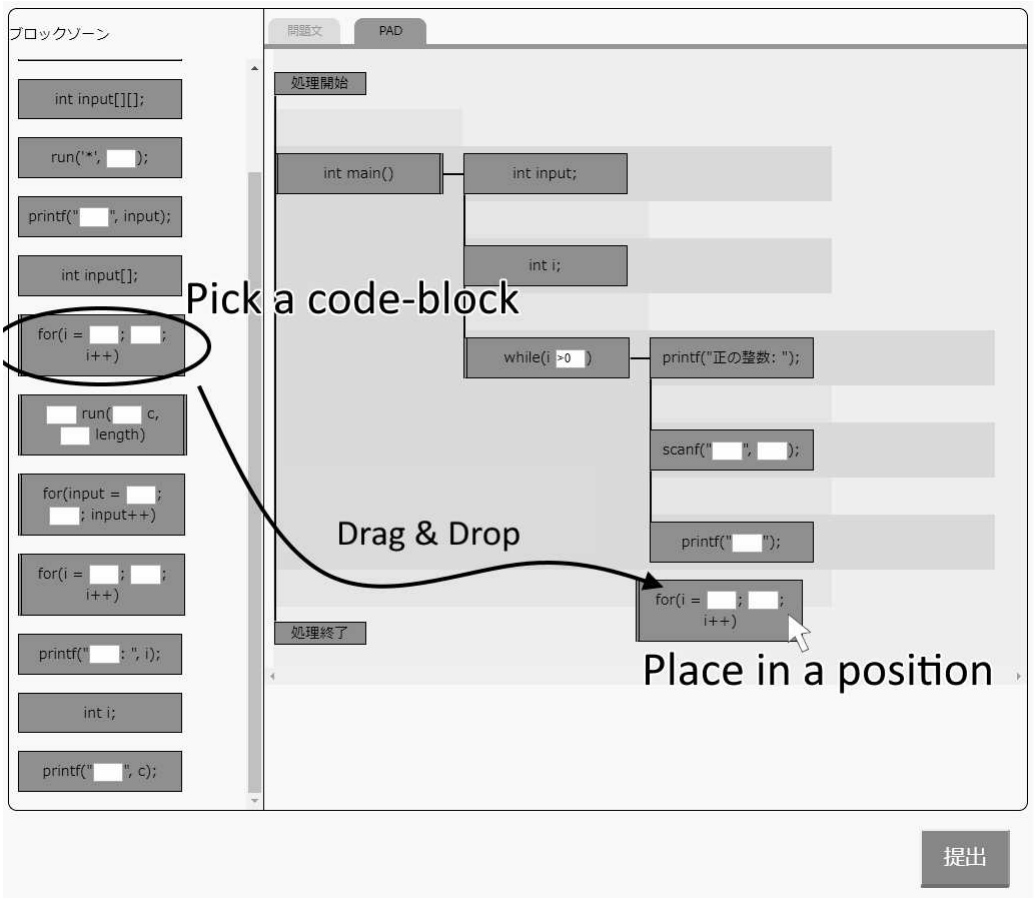

Fig. 3. Drawing screen

regression as machine learning methods. Since the two methods can calculate the importance of variables, it is possible to consider classification factors. Although the random forest method is more accurate, the probability of classification can be known by logistic regression and this classification probability can be grasped as an understanding level of 0.0 to 1.0. The understanding is judged by two measures: the learning item achievement level and programming thinking ability. Therefore, the proposed method uses machine learning to create two models of learning item achievement and programming thinking. This is based on the knowledge that the real ability of programming cannot be realized unless the learning items achievement and the programming thinking are compatible.[12]

According to the schema theory[5], a person combines knowledge to solve tasks. Therefore, the programming thinking, which is the power to combine knowledge, is considered to be based on knowledge.

Therefore, the degree of schema construction is defined as follows by using the learning item achievement k and the programming thinking ability l as an index that integrates the learning item achievement and the programming thinking ability. k and l are defined by values between 0 and 1.

$$
\begin{aligned}
d &= \left\{ \begin{matrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{matrix} \right\} \bullet \left\{ \begin{matrix} k \\ l \end{matrix} \right\} \\
e &= (f(k) - l)^2 = (k - l)^2 \\
s &= d - e
\end{aligned}
$$

d is the dot product with the unit vector $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ and vector $(k.l)$. In other words, d takes the maximum value if both of k and l are 1.0. d become larger as k approaches to l. From the geometrical perspective, as d becomes larger the direction vector $(k.l)$ approaches to that of the line with the slope 45 degrees, whose direction vector is $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. e is the squared error from the line $f(x) = x$. When calculating s, which represents the balance between learning item achievement and programming thinking ability, e indicates imbalance and functions as a penalty.

E. Factors of understanding for each learner

In order to estimate the understanding of a learner, it is possible to obtain the classification result of whether or not the learner understands the task from the machine learning model. When using logistic regression and random forest, it is possible to see which explanatory variables and how much

influenced the classification by referring to the regression coefficient and variable importance, respectively. From these indexes, the factors that a learner classified as not being understood are found. The position on the code to which the factor corresponds, that is, the misunderstanding part, becomes clear. In this study, this factor is called the misunderstanding factor. In addition, these indicators provide clues as to where learners tend to stumble and which parts a learner does not know.

F. Feedback to learners and teachers

The proposed method provides two types of feedback:

- Degree of Schema construction including programming thinking
- Misunderstanding factors and Misunderstanding points that take programming thinking into consideration

The schema construction can visualize understanding of a large number of learners with unified numerical values. Based on the schema construction, the learner can grasp how much he/she understand with compared to others. Furthermore, the instructor can detect learners who need guidance at early stage.

In addition, when analysis is performed over a period of time, it is possible to calculate the degree of growth and to find stumbling based on changes in the degree of schema construction.

The misunderstanding factors and misunderstanding points can be considered from the coefficient of logistic regression and the importance variables of the random forest. The misunderstanding factors can be grasp by comparing the learner's behavior based on the variables of high importance. This analysis tells the learner what part they do not understand or how they tends to answer. In addition, the instructor can be informed of what part learners confused and what learners do not understand. Therefore, it is possible to create a new teaching plan for individuals and the whole.

IV. Experiment

A. Objective and method

The purpose of this experiment is to clarify the understanding considering the learner's programming thinking ability from the learner's operation history using code puzzles. The subjects are 17 university students, including first-year undergraduate students who started learning programming and first-year graduate students who are accustomed to programming. Each of the subjects solved a given programming task. The time to solve the task took the minimum of 7 minutes and the maximum of 40 minutes. The degrees of attainment were various between the subjects. The experiment was conducted with the time constraint of 1 hour per person. In the first 10 minutes, we gave a brief tutorial on how to use the tool. In the next 40 minutes, each subject worked on a programming task using the code puzzle. In the last 10 minutes, for the labeling described later, we conducted a questionnaire asking about the cognitive load of this task and usual experience. While solving the task, the instructor did not give any hints and just watched to measure programming thinking ability.

B. Actual understanding and cognitive load measurement

Before performing analysis by machine learning, the instructor labeled the objective variables used in supervised learning. The label used for training is a binary value (0/1) indicating whether or not the subject understood. For subject, a 0 or 1 label was assigned to each subject for each of the learning item achievement and programming thinking ability.

1) The problem content was very complicated.
2) The knowledge used for filling in the blanks and functions was very complicated.
3) The concepts and ideas in the task were very complicated.
4) It was very unclear how to use tools and PAD notation.
5) It was difficult to understand how to use tools and PAD notation.
6) Tools and PAD notation are very inefficient from a learning perspective.
7) The task has improved my understanding of programming.
8) The task improved the understanding of programming process
9) The task has increased my understanding of programming concepts and definitions.
10) The task has improved my knowledge of programming.
11) It took a lot of mental effort because the task was complicated.
12) Due to the explanation in the task and the usage of the tool, I took a lot of mental effort.
13) It took a lot of effort to improve knowledge and understanding in the task.
14) Relative programming score
15) Relative programming experience

Fig. 4. Questionnaire for measuring cognitive load

The questionnaire answer in the experiment was used for labeling. Figure 4 shows the list of questionnaires used in this method. The questionnaire includes the items to investigate the cognitive load [13] felt by the subject while solving the task. This questionnaire is a question group based on a 10-point Likert scale. The subject answers the question sentence subjectively. Generally, if the learner does not misunderstand a task, a high cognitive load means a low understanding of the task. However, in reality, there were many contradictory in the answer to the questionnaire and the performance of the task. In other words, there were many subjects who did not perform the task well despite the small cognitive load. This suggests that some subjects misunderstood the content of task. From this, it can be said that it is difficult to evaluate understanding, especially programming thinking ability, unless the instructor watches the learner's behavior of the process

of solving the task. Therefore, in the labeling, with referring to the questionnaire, the instructor give the labeling from the viewpoint of both the learning item achievement and the programming thinking based on the behavior during task.

C. Estimation of understanding and distribution of learners

Based on the operation history data collected from the 17 subjects and the labels given by the instructor, the learning item achievement and programming thinking ability were classified. Both the random forest and logistic regression are used for classification. With the random forest, high classification accuracy and variable importance can be obtained. On the other hand, with the logistic regression, variable importance based on regression coefficient and understanding by values from 0.0 to 1.0 can be obtained. For the analysis, we used Scikit-learn in Python to find the optimum parameters by the grid search and ensured the generalization performance by using the Leave-one-out cross-validation.

TABLE I
PERFORMANCE OF LEARNING ITEM ACHIEVEMENT CLASSIFICATION MODELS

Logistic regression	
Accuracy	0.82
Precision	0.78
Recall	0.88
F-score	0.82

Random forest	
Accuracy	0.82
Precision	0.86
Recall	0.75
F-score	0.80

TABLE II
PERFORMANCE OF PROGRAMMING THINKING CLASSIFICATION MODEL

Logistic regression	
Accuracy	0.82
Precision	0.79
Recall	1.00
F-score	0.88

Random forest	
Accuracy	0.94
Precision	0.92
Recall	1.00
F-score	0.96

Tables I and II show the accuracy rates of the model created by this experiment. The accuracy rates of both models exceeds 80% and it can be said that the models have a high degree of accuracy.

Additionally, the distribution of the subjects' understanding is displayed on the two-dimensional coordinates using the two axes of the classification probabilities of learning item achievement and programming thinking ability predicted by the logistic regression. The distribution map is shown in Fig. 5. The label of the distribution map is the degree of schema construction of the corresponding subject. In the distribution map, the subjects close to the diagonal line of 45 ° show that the degree of learning items achievement and the programming thinking ability are similar and that the two abilities are well balanced. Among them, the subjects located in the upper right shows that both abilities are high. On the other hand, learners distant from the 45 ° line to the lower side have a high learning item achievement but have a low programming thinking ability, which indicates a poor balance. If the schema construction degree is used, those who have high knowledge are evaluated only to some extent and those who have a balance of both values are highly evaluated.

V. CONSIDERATION

A. Correlation between schema build and grade

The validity of the schema construction level is confirmed by comparing the calculated schema construction level with the results of the actual class performance.. Table III shows the calculated schema construction level of the subjects and the total scores of the 32 fill-in-the-blank tasks that the subjects have took in the actual classes. The correlation coefficient among them is also shown. The subjects were those who were able to obtain actual class performance data among the 17 subjects. In addition, we excluded those that could not be tested due to poor physical condition and excluded weeks of tasks that were not directly related to programming.

TABLE III
CORRELATION WITH THE CALCULATED SCHEMA CONSTRUCTION LEVEL AND A LIST OF GRADES

Subject	Schema construction level	Score
A	1.922	754
B	1.887	760
C	1.885	736
D	1.510	713
E	1.502	769
F	1.016	723
G	1.001	775
H	0.430	542
I	0.215	674
	Correlation	0.67

The correlation coefficient of 0.67 cannot be said to be extremely high. However, it correlates to some extent with the credible data of scores. This shows the validity of the degree of schema construction. The reason why a high correlation does not appear is that the data of the performance of the fill-in-the-blank tasks does not consider programming thinking ability. On the other hand, the proposed method easily quantifies the real programming ability considering programming thinking ability.

B. Consideration of Understanding Factor and Learner Behavior by Important Variables

We compare the classification models of learning item achievement and programming thinking ability and consider the differences. Table IV shows the important variables of the models created in this experiment.

The subjects with low learning item achievements moved many variable blocks and loop blocks by focusing on the variables related to "touch" and "hover". From this, it is said that they did not understand how to use variables and could not answer the basic parts such as loop statements. It is considered that this is because the subject with low learning item achievement could not think the meaning of the variable name. In addition, the learning item achievement is lower when the iterator variable “i” is declared earlier and the touch frequency up to 1/4 hour is higher. This suggests that the learners who worked on the task sooner after provision of the task got confused. It is considered that they have repeatedly switched between the drawing screen and the task display

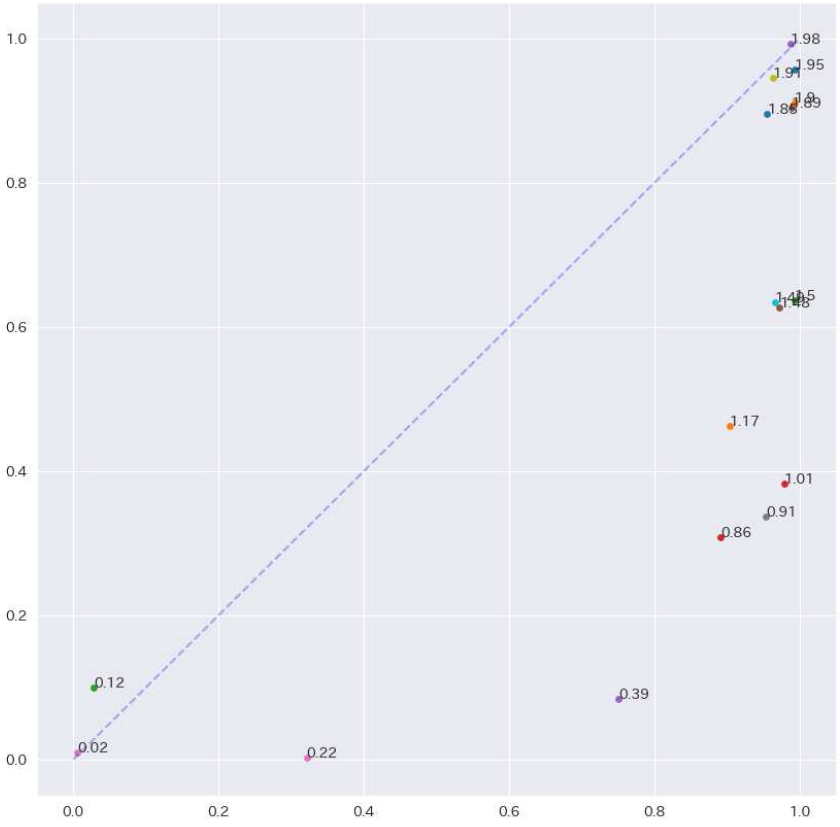

Fig. 5. Distribution chart based on learning item achievement and programming thinking ability

screen. On the other hand, the subjects with high programming thinking ability took less time to adopt the "input block". From this, it is considered they immediately grasped the meaning from the name of the variable and adopt it. Moreover, since they read the explanation of the variable especially "loop block" carefully, it can be said that they tend to think carefully how many times to loop.

In the learning item achievement level, the "one-character printing block", which is the deepest part of the loop, is important. From this, it can be seen that the subjects with low learning item achievement could not reach to think this module. On the other hand, in programming thinking, the time spent for watching the explanation of single-character printing is important. This suggests that the subjects with high programming thinking have reached the deepest part of the loop and considered it.

In addition, the variables related to the touch frequency is important in the learning item achievement. This indicates that the subjects with low learning item achievement tended to assemble modules without thinking. In programming thinking ability, the variable of touch frequency is important. In other words, it was found that the subjects without programming thinking touched many modules that were not necessary for the task. They have been at a loss. In addition, there are many important variables that indicate "looking at the module description". This shows that the subjects with high programming thinking ability firmly identified the modules to be used at first and established the course before tackling the task.

The important variables for the programming thinking ability can indicate similarity between the model and the subject's answers. This shows that an ideal answer cannot be achieved only with the learning item attainment level. It is because programming thinking is indispensable for solving code puzzles.[4]. In addition, the variable of "the time of looking at the explanation of 1-character printing" is important. This results implies that the subjects who were confused about the matters such as "1-character printing" may be greatly evaluated negatively. Moreover, since "designation of arguments and

TABLE IV
TOP 10 IMPORTANT VARIABLES BY LOGISTIC REGRESSION AND RANDOM FOREST

touch: Number of drag and drop
hover: Time spent looking at the code block description
first: How fast the code block was first adopted

[1]List of important variables for classifying learning item achievement

	Regression coefficient	Explanatory variable	What block
1	0.64	touch14	1 character printing
2	0.43	Blank 9 Similarity	Number of printing
3	0.42	touch19	Line break
4	0.35	hover18	For loop
5	0.34	Blank 1 Similarity	Num of loop specifying
6	-0.35	touch18	For loop
7	-0.37	Blank 3 Similarity decrease	Variable specifying
8	-0.40	Tab switching	
9	-0.42	touch12	Variable defining
10	-0.55	first12	Variable defining

	Variable importance	Explanatory variable	What block
1	0.15	touch14	1 character printing
2	0.14	first17	For loop(Trap)
3	0.10	Touch frequency ~ 1/4 period	
4	0.09	Touch frequency ~ 3/4 period	
5	0.09	hover17	For loop(Trap)
6	0.08	touch19	Line break
7	0.07	first13	For loop
8	0.06	hover20	line number printing
9	0.05	Touch frequency	
10	0.04	first8	console printing

[2]List of important variables for classifying programming thinking ability

	Regression coefficient	Explanatory variable	What block
1	0.73	Model answer similarity	
2	0.45	Blank 11 Similarity	Argument specifying
3	0.39	Blank 5 Similarity	Specifier specifying
4	0.39	first10	Printing(Trap)
5	0.35	first5	Variable defining
6	0.33	Blank 6 Similarity	Line break
7	0.32	Blank 12 Similarity	Condition specifying
8	-0.37	Blank 9 Similarity decrease	Number of printing
9	-0.47	Touch frequency distribution	
10	-0.72	hover14	1 character printing

	Variable importance	Explanatory variable	What block
1	0.18	first5	Variable defining
2	0.17	touch13	For loop
3	0.12	hover14	1 character printing
4	0.12	hover12	Variable defining
5	0.12	first17	For loop(Trap)
6	0.10	touch11	Variable defining(Trap)
7	0.06	hover15	Function defining
8	0.05	Percentage of Drawing time	
9	0.05	touch16	Character input
10	0.03	Blank 8 Similarity decrease	Number of printing

conditions" is important, it can be said that a learner with high programming thinking ability grasps the flow of data and chooses appropriate variables.

The findings about the subjects with low understanding obtained in this experiment are shown below.

Regarding the low learning item achievement, it is difficult to interpret the usage of variables and loops block and there is a tendency to get lost in basic matters. , They also tend to tackle the task immediately without interpreting the task deeply.

Regarding the low programming thinking ability, it is greatly evaluated negatively when the subjects cannot approach the model answer and eventually stumbles on a basic matter. They do not look at the explanations deeply, touch many blocks, and do not set the course for answers. In other words, they cannot think deeply. Furthermore, they cannot assemble the structure firmly. They do not understand the data flow and cannot specify arguments or conditions. From these facts, it is considered that the learning item achievement is the basic ability and the programming thinking ability is the comprehensive ability including the learning item achievement.

C. Usefulness in educational settings

The results of this experiment show that it is possible to estimate the comprehension level considering the programming thinking ability from the learner's operation history when answering the code puzzle. At present, there are many one-to-many forms in educational settings such as universities or companies. Though they sometimes hire multiple assistants to support learners, the current situation is that the number of assistant is insufficient. Therefore, it is difficult for the instructor to grasp the understanding level of all learners.

The schema construction estimated by the proposed method can measure the understanding of many learners at once if a model is learned by labeling dozens of samples. In particular, the classification probability output by the logistic regression is a value from 0.0 to 1.0. Therefore, by setting a threshold appropriately, we can discriminate between learners who can be able to solve the task and learners who need guidance.

In the field of education, it is common to learn various elements with multiple tasks. The schema construction calculated for multiple tasks over a period can be visualized by studying the transitions of learning progress and growth. Moreover, by referring to the important variables, the instructor can consider where the learner stumbled. For example, in the task of this experiment, it can be seen that many learners did not understand what the meaning of the variable has from its name because the importance of the block for the variable was high. Since the importance of variables on blocks regarding the number of loops was high, it can be seen that many learners could not fully consider the processing flow.

VI. ISSUES AND FUTURE

The problem of the proposed method is the ambiguity of blocks used in code puzzles and the cost of labeling. At first, creating blocks used in code puzzle is a large cost for creating a task in an actual educational setting. As a countermeasure, it is possible to implement an algorithm for adding similar blocks or randomly select blocks from other tasks' blocks and them.

In addition, labeling is not a small cost in the educational setting. However, this is unavoidable as long as supervised learning is used. Blikstein *et al.* [14] suggested that the understanding of learners can be measured by using the clustering method with programming structure. Therefore, in our research, there is a possibility that the understanding and the factor of misunderstanding can be estimated by labeling using such the clustering method. The model created in one year can be used in the following years as long as the tasks are the same and the level of learners is the same. In general, at one university, the levels of students are almost the same for several years. Furthermore, the distribution of the programming ability of students as described in this paper can be visualized based on the learning item achievement and programming thinking ability calculated by the model. Furthermore, there is a program that visualizes the students' behavior. With visualization tools, it is not difficult to label for multiple people. This also allows refinement of the model.

VII. Conclusion

This paper proposed a method to estimate the learner's programming thinking ability as well as learning item achievement by analyzing his/her answering process of the code puzzles.

In order to measure programming ability precisely, it is necessary to provide the learners an environment like a code puzzles where he/she can focus on combining programming elements and to extract his/her answering process and behavior. The estimation is performed by learning the random forest and logistic regression models, where the objective variables are programming thinking or learning item achievement levels, respectively, and the explanatory variables are learner's actions in solving code puzzles.

As the result of the experiment, it was found that the proposed method was able to estimate the understanding with the accuracies of more than 80In addition, considering the difference between the learning item achievement and the programing thinking ability from the difference of the variable importance, it was confirmed that the programming thinking ability is based on the learning item achievement and that the programming ability cannot be measured only by the learning item achievement. Furthermore, based on the schema theory[5], we defined the schema construction level from two labels, learning item achievement and programming thinking ability. We compared it with the performance of the fill-in-the-blank problems in actual classes. These results suggest that just the performance of the fill-in-the-blank problems does not indicate the programming ability.

The results of this experiment show that the learner's programming ability can be measured more accurately by considering the learner's logical constructive ability in the code puzzle rearrangement problem. The accurate measurement of the learner's programming ability contributes to developing the learner's true programming ability, which cannot measured by only the score of written tests. In addition, the importance of each variable in the behavior analysis leads to the identification of learner's misunderstanding factors and the improvement of class contents.

In the future, we will develop a new method to deal with more complicated programming problems and to simplify making tasks to be applied to the proposed method.

References

[1] J. T. S. P. o. C. S. B.-S. C. B. Kenneth L. Whipkey, "Identifying predictors of programming skill," *ACM SIGCSE Bulletin*, vol. 16, no. 4, pp. 36–42, 1984.

[2] L. J. M. U. of Cincinnati, "Identifying potential to acquire programming skill," *Communications of the ACM*, vol. 23, no. 1, pp. 14–17, 1980.

[3] M. of education, "Elementary programming education guide (second edition)," https://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_02_1.pdf.

[4] D. Parsons and P. Haden, "Parson's programming puzzles: a fun and effective learning tool for first programming courses," in *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 2006, pp. 157–163.

[5] W. Schnotz and C. Kürschner, "A reconsideration of cognitive load theory," *Educational psychology review*, vol. 19, no. 4, pp. 469–508, 2007.

[6] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 73–84.

[7] A. Mysore and P. J. Guo, "Porta: Profiling software tutorials using operating-system-wide activity tracing," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 201–212.

[8] P. J. Guo, "Codeopticon: Real-time, one-to-many human tutoring for computer programming," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, 2015, pp. 599–608.

[9] H. S. So Asai, Dinh Thi Dong Phuong, "Identification of factors affecting cognitive load in programming learning with decision tree," vol. 14, no. 11, 2019, pp. 624–633.

[10] P. Ihantola, J. Sorva, and A. Vihavainen, "Automatically detectable indicators of programming assignment difficulty," in *Proceedings of the 15th Annual Conference on Information technology education*, 2014, pp. 33–38.

[11] Y. Futamura, T. Kawai, Y. Horikoshi, M. Tsutsumi *et al.*, "Program design and creation with pad (problem analysis diagram)," *IPSJ Transactions*, vol. 21, no. 4, pp. 259–267, 1980.

[12] J. W. Coffey, "Relationship between design and programming skills in an advanced computer programming class," *Journal of Computing Sciences in Colleges*, vol. 30, no. 5, pp. 39–45, 2015.

[13] B. B. Morrison, B. Dorn, and M. Guzdial, "Measuring cognitive load in introductory cs: adaptation of an instrument," in *Proceedings of the tenth annual conference on International computing education research*, 2014, pp. 131–138.

[14] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller, "Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming," *Journal of the Learning Sciences*, vol. 23, no. 4, pp. 561–599, 2014.