Identification of Factors Affecting Cognitive Load in Programming Learning with Decision Tree

So Asai[1*], Dinh Thi Dong Phuong[2], Hiromitsu Shimakawa[1]
[1] Ritsumeikan University, Kusatsu, Shiga, Japan.
[2] Paracel Technology Solutions Co., Ltd, Da Nang, Vietnam.

* Corresponding author. email: asonaso116@gmail.com
Manuscript submitted January 30, 2019; accepted March 8, 2019.
doi: 10.17706/jcp.14.11.624-633

Abstract: We propose a method to identify cognitive load and its factors affecting learning efficiency in programming learning from learning behavior by learners. In general, since concepts of programming is difficult, some learners suffer inappropriate cognitive load to understand them. Although teachers must keep the cognitive load of such learners appropriate, it is difficult for them to find learners under inappropriate cognitive load from a large number of learners. In order to find learners with inappropriate cognitive load, decision trees generated from learning behavior when learners solve fill-in-the-blank tests identifies cognitive load. An experiment shows that the decision trees generated by our method identify factors of cognitive load for IL and GL. This result clarifies learning factors affecting cognitive load of learners and lets the teachers can take measures to their cognitive load appropriate.

Key words: Decision tree analysis, e-learning, machine learning, programming learning.

1. Introduction

In educational institutes of information technologies, many learners have not experienced programming. It is difficult for them to understand unfamiliar concepts in programming, which makes them fail to acquire programming skills.

One of the severe causes of understanding failure is that inappropriate cognitive load [1] is imposed on them. In programming learning, the cognitive load is an important factor, because it is closely related to understanding and schematization of programming skills of learners. Teachers are required to keep learners away from inappropriate cognitive load, providing an environment which promotes learners to engage in programming learning. However, it is difficult for teachers to find learners suffering from inappropriate cognitive load among all learners.

This study proposes a method to estimate factors causing the cognitive load from learner's behavior in programming learning. The method generates decision trees with the learning behavior, to identify their cognitive load. It clarifies states of the cognitive load of each learner, along with its factors. The teachers can adjust their cognitive load with minimal efforts.

2. Cognitive Load in Programming Learning

2.1. Cause of Understanding Failure

In programming, learners must utilize many concepts to write appropriate code. Learners must organize

various kinds of knowledge of programming in their mind. Principal factors which prevent learners from achieving programming skills are difficulties in both of the concepts themselves and ways of organizing them [2]. Learners fail to acquire either of them do not know what to write for assignments or examinations. Though learners know the concepts, if they lack in ability to organize them, the learners cannot imagine actual behavior of program code they have written. Such learners have many possibilities to fail in programming learning because they avoid challenging higher-level assignments which require the organization. The teachers should find those learners in early stages to prevent them from failing to learn programming.

2.2. Cognitive Load Affecting Learning

Working memory is a pseudo memory which corresponds to the capacity for what to be learned. It is consumed at thinking or holding something in learning. The amount of working memory varies with each learner. Every learner has his/her capacity of learning. When learners repeat to study identical elements in working memory, the items organize as a schema. This study refers to it as schematization. Once elements become a schema, they are organized together with their usage. No cognitive load imposes on the learners for their use.

The cognitive load affects understanding failure caused by the difficulty of programming concepts [3], [4]. The cognitive load indicates how much working memory is assigned to a task on an unknown item to be learned. In the cognitive load theory, usage of the working memory is classified into the following three types [5], [6].

Intrinsic Load (IL)

IL is due to the inherent difficulty of the assignment against the abilities of learners. This load gets high when learners feel extreme difficulty for the assignment because their working memory is small for the assignment.

Extraneous Load (EL)

EL is caused by surrounding learning environments. EL is brought by the poor quality of teaching materials and lectures used by learners. Teachers need to design the materials and the lectures so that they reduce this load.

Germane Load (GL)

GL is related to schematization of contents to be learned. Imposition of this load means that learners are organizing the learning content and its usage into their knowledge for themselves. Learners are encouraged to experience this load [7].

If it is an ideal learning situation that learners continue to acquire new knowledge, it is desirable that cognitive load is high in GL, in which knowledge is being schematized, while it should be low in IL and EL. This study refers to it as an appropriate state of the cognitive load. Teachers must endeavor to keep the cognitive load of learners appropriate. However, every learner feels different difficulty for each of the given learning content. Effects of lesson design also vary with learners in programming lectures and exercises conducted as mass classes. It is difficult to estimate the cognitive load for each learner. Even more, it is nearly impossible to judge it from the appearance of learners on the spot during the class.

2.3. Estimation of Cognitive Load

Many researchers are engaged in works to measure the cognitive load [6], [8], [9]. Several methods are proposed to measure the cognitive load. Their usefulness is confirmed in various fields.

Morrison *et al*. [10] measured the cognitive load of learners in the programming classes, without sensors. He applied the method Leppink *et al.* [11] established for statistics classes to programming learning. The cognitive load is evaluated through 10 questions. Learners answer each question by 11 scales. These

question items are correlated to figure out either of IL, EL, or GL. However, since learners answer the questions once in the class, the cognitive load is assessed as one value throughout the whole learning process in the class. This method does not clarify factors of the cognitive load of learning, because it needs fine course assessment of the cognitive load in the progress of the time.

Yousoof *et al.* [12] proposed a method to measure and reduce cognitive load from its accumulation by dividing visual information and character information in the programming class. This method mainly focuses on EL. It does not adequately consider IL or GL, which is caused by learners. With considering three types of cognitive load, it is necessary to clarify the factors of essential understanding failure based on learning behavior.

Fridman *et al.* [13] measured the cognitive load during driving vehicle without any wearing sensors. Applying the deep learning to video data of eye movements, this study judges the cognitive load in real time. This study does not distinguish the three types of the cognitive load. A requirement in programming learning is to discriminate IL, EL, and GL. Since IL and EL decrease learning efficiency, they should be low. On the contrary, high GL is preferable, because it implies the learner schematizes learning items.

It is not ideal to attach physical sensors to learners, to obtain cognitive load in programming learning. Since there are many learners, it brings the huge cost to attach sensors to all of them. In addition to that, sensors may have inappropriate influence on learning efficiency.

A contribution of our study is to identify factors of cognitive load, estimating the three types of cognitive load without specific sensors.

3. Estimation of Factors Affecting Cognitive Load

3.1. Method Overview

The study aims to estimate cognitive load from behaviors of programming learning, to reveal factors that cause the cognitive load. We propose a method to estimate the factors when learners study programming with procedural languages such as C. Fig. 1 shows the method overview. For the estimation, learning behavior is collected when past learners answer fill-in-the-blank tests.

Letting the learners specify their cognitive load [10], the method generates decision trees with the learning behaviors and the cognitive load. These are combines as datasets.

The decision trees associate characteristics of learning behavior with factors of the cognitive load of the past learners. The cognitive load of present learners is figured out, with the decision trees applied to their learning behaviors. The method helps teachers to confirm whether sufficient learning effects have brought to the current learners. The teachers can also address learners with insufficient learning effects based on factors of their cognitive load.

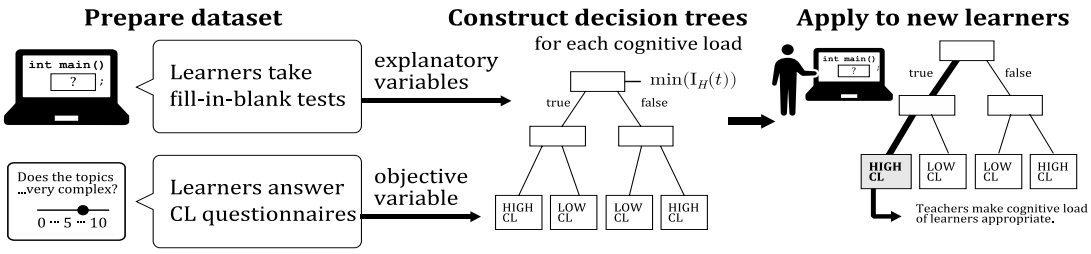

Fig. 1. Method to estimate cognitive load and its factors.

3.2. Collecting Learning Behavior

We focus on learning behavior when learners answer fill-in-the-blank test. Fill-in-the-blank test is frequently used to measure learner's understanding in programming classes [14]. The scores of

fill-in-the-blank tests reveal the understanding of learners because learners can answer code fragments only in blanks, considering the coincidence of disclosed code [15]. In fill-in-the-blank tests, it is less likely to speculate answers than in multiple-choice tests [16]. Fill-in-the-blank tests are also useful for direct/indirect measurement of the cognitive load because the cognitive load is imposed when learners seek correct answers.

Generally, in fill-in-the-blank tests for programming, test forms are distributed to learners. A teacher collects the test forms to score them to notify the learners of the results later. Since the results are not notified on the spot, learners proceed to next learning stage, without making their learning effects sure.

The method provides an automatic scoring system [17]. It is a web application, with which learners can score their answers interactively. The system scores an answer that a learner gives for each blank on demand. It notifies the correctness of the answer immediately. Learners using the system are allowed to reconsider their answers many times until their answers become correct within the time limit.

In our method, one test corresponds to program code, parts of which are blanked out. More than one tests are provided for learners. As learning behavior, the method collects 3 data items: the consuming time, change histories of answers, and the number of scoring demands. More concretely, the system records the elapsed time from the time point a learner starts a specific test, answers the learner submits for each blank, and the number of scoring demands transmitted to the system, respectively. We focus on 4 types of specific behaviors as the followings:

Scoring requests

When learners come up various answers for a test, they can check whether each of their answers is correct. Suppose learners whose programming ability is short to solve specific tests. It is hard for them to prepare items various enough to solve the tests in their working memory. It prevents them from coming up with all answers. Therefore, we assume that such learners demand to score rarely.

Meanwhile, learners who are high in EL do not seem to frequently demand to score, because they do not fully understand the application usage and statements of the assignments. The system enables learners to request scoring many times. The feature of the system aims at letting them seek correct answers, which increases the cognitive load inside them.

Time until first input

This behavior is defined as the elapsed time from the designation of a specific blank with the cursor to start of writing down a code fragment to fill it. Considering appropriate procedures, learners must convert them into the right programs. It enables them to find correct answers. If the task is difficult for learners, they are assumed to take much time to fill up blanks. At that time, their IL seems to get high. The method adopts the average of the elapsed time as one of the explanatory variables of learner's IL.

Page transitions

In fill-in-the-blank tests, learners engage in multiple assignments in one session to measure their understanding of various programming skills. The learners can choose assignments to solve and switch them halfway. When a learner either finishes answering for all blanks of an assignment or gives up, the learner moves to other assignments. The more learners feel high degrees of difficulty for assignments, the more they transit assignments.

Time transitions for correct answer rate

Learners try to figure out the correct answers for each test, demanding the scoring to the system. Let the correct answer rate as the ratio of blanks they fill with correct answers against all the blanks. As a whole, learners increase their correct answer rate. Learners who have enough understanding of the tests can answer all of them quickly. The learners get the correct answer rate corresponding to the full mark or one close to it.

On the other hand, learners who lack understanding find correct answers using a long time or fail to find them at all. The time transition of the correct answer rate is anticipatedly useful to discriminate the cognitive load. For representation of the anticipation in an integrated way, the method quantifies the time transition by (1).

$$T = \frac{1}{2}\sum_{k=1}^{n}(p_k + p_{k-1})(t_k - t_{k-1}) + p_n(t_l - t_n), \quad (1)$$

where n is the number of scoring demands of the learner, p_k is the correct answer rate at the kth scoring, t_k is the elapsed time at the kth scoring from t_0, and t_l is the deadline for answering. $k = 0$ is the state at the start time of answering, where $t_0 = 0$ and $p_0 = 0$.

The quantification enables us to evaluate the anticipation, as the accumulation of the correct answer rate of a learner over the elapsed time, as Fig. 2 shows. In the case that the correct answer rate becomes high early, T gets large as shown in Fig. 2(a). Conversely, when a learner is low in the correct answer rate, using a long answering time, T is small as shown in Fig. 2(b), (c).

The automatic scoring system for fill-in-the-blank tests can collect these behaviors to calculate explanatory variables of factors of the cognitive load.

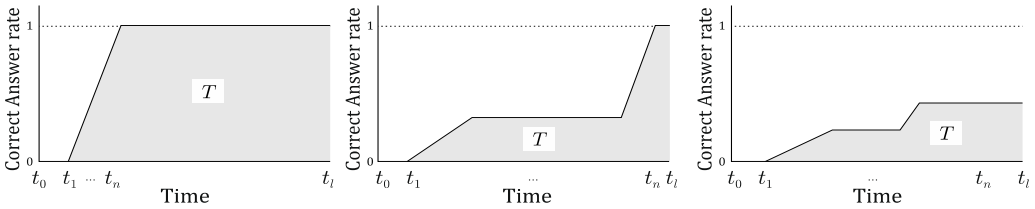

Fig. 2. Examples of time transitions for correct answer rate.

3.3. Investigating Cognitive Load

Table 1. Questions of Cognitive Load Questionnaire

No.	Question sentence	Target CL
1	The topics covered in the activity of solving fill-in-blank problems were very complex.	IL
2	The activity of solving fill-in-blank problems covered program code that I perceived as very complex.	IL
3	The activity of solving fill-in-blank problems covered concepts and definitions that I perceived as very complex.	IL
4	The instructions and/or explanations of the website during the activity were very unclear.	EL
5	The instructions and/or explanations of the website were, in terms of learning, very ineffective.	EL
6	The instructions and/or explanations of the website were full of unclear language.	EL
7	The activity of solving fill-in-blank problems really enhanced my understanding of the topics covered.	GL
8	The activity of solving fill-in-blank problems really enhanced my knowledge and understanding of computing/programming.	GL
9	The activity of solving fill-in-blank problems really enhanced my understanding of the program code covered.	GL
10	The activity of solving fill-in-blank problems really enhanced my understanding of the concepts and definitions.	GL

The method uses the cognitive load measurement questionnaire [10] to identify three types of the cognitive load of learners. To investigate the cognitive load, learners answer the questionnaire which consists of 10 questions. The questions are classified into 3, 3, and 4 items corresponding to IL, EL, and GL, respectively. In the method, qualifiers are added to each statement of the cognitive load measurement questionnaire, to clarify the target of each question. Table 1 shows the list of the questions. Learners evaluate each of the questions with 11 scales. In the case of a strong agreement for the question, learners mark a larger number. When the marked number for a question is large, the learner is judged to have a high degree of the cognitive load.

3.4. Generating Decision Trees

The method generates decision trees to associate characteristics of learning behavior with factors of the cognitive load evaluated by the questionnaire. Explanatory variables and objective variables of the decision trees are the learning behavior and factors of the cognitive load, respectively. Decision trees are generated for each of the three types of the cognitive load because the purpose of our study is to clarify factors for each of the 3 types of the cognitive load. At each node of a decision tree, a feature of the learning behavior is determined so as to bisect learners. The impurity in each node of the decision tree should be small to minimize the information gain. The impurity in each node is represented by entropy $I_H(t)$ which is calculated with (2).

$$I_H(\mathrm{t}) = -\sum_{i=1}^{2} P(i \mid t) log_2 P(i \mid t), \tag{2}$$

where $P(i \mid t)$ is the ratio of the number of child node i branched from node t. The difference of the entropy after the branch from the one before the branch is calculated to maximize the information gain. The information gain $G(t)$ at the node t is obtained from (3).

$$G(t) = I_H(t) - \sum_{i=1}^{2} \frac{|t_i|}{|t|} I_H(t_i) \tag{3}$$

Nodes whose information gain is less than a threshold value are regarded as leaves, to prevent over-fitting of the decision trees. Learners whose cognitive load matches the branching condition are extracted in each node. Eventually, learners of specific learning behavior characteristics are given to each of leaf nodes. The characteristics are obtained as an objective variable. Nodes on a path from the root node to a leaf node corresponding to high cognitive load are factors of learning behavior which affect the cognitive load.

3.5. Identifying Cognitive Load of Learners

New learners solve fill-in-the-blank test with the automatic scoring system as past learners did. Their learning behavior is checked against the decision trees obtained from the past learners. The cognitive load of the new learners is characterized by the branching conditions which appears in the path from the root node to the leaf node. The detected degree of the cognitive load is notified of teachers. When IL or EL is high or GL is low, the teacher should follow up the learners to make their cognitive load inappropriate. Learners can understand programming more sufficiently while suppressing burdens of teachers. It contributes to preventing learners from failing programming learning. Teachers can focus on preparation of lectures to provide higher educational effects.

4. Experiment

4.1. Overview

An experiment was conducted to ascertain whether the method identifies factors of cognitive load. The

purposes of this experiment are the following:

- Collecting datasets of learning behavior of learners answering fill-in-the-blank test
- Verifying decision trees generated from the datasets

Subjects are Vietnamese college students who are learning programming in C and information technology. All of them are the second year of college and has already educated in C language programming for beginners. At the time of the experiment, their interests and abilities to C programming are diverse. The materials of the experiment were provided in English since most of them can read English enough. In the experiment, subjects solved five assignments of fill-in-the-blank test whose several code fragments are blanked out. For each assignment, concepts of two-dimensional array, pointer, structure, linked list, and sorting algorithm are used. All of the concepts universally difficult in programming learning. The assignments asking the concepts were chosen so that unbiased datasets can be obtained since the concepts make a difference in understanding of learners [18], [19].

The quality as an assignment for learning programming is guaranteed since these assignments are used in programming classes in Ritsumeikan University. The subjects use a website implemented automatic scoring for fill-in-the-blank test described in Section 3.2. The subjects access the experimental website with a browser that they usually use and login with user ID and password given in advance to solve assignments. The subjects can solve from any assignments and switch assignments to solve on the way within the time limit. Learning behavior of subjects is stored on the server by asynchronous communication with Web beacons immediately as soon as learners take a specific action. After the time limit has elapsed, the subjects finish solving the assignments and subsequently answer the questionnaire of cognitive load measurement.

4.2. Result

Datasets of learning behavior are obtained from 54 subjects. 3 subjects are excluded due to the insufficient number of learning behavior. Decision trees corresponding to IL, EL, and GL are generated from the datasets with the proposed method. The explanatory variables are given 4 kinds of behaviors described in Section 3.2, and the objective variables are given high or low of IL, EL and GL based on the answers of the cognitive load measurement questionnaire. In order to verify the accuracy of the decision tree, datasets are divided into data for decision tree generation and verification with K-fold cross-validation with $k = 6$. Fig. 3 shows the generated decision trees. The nodes at 5 or less depth in any decision trees are pruned. Table 2 shows the results of the accuracy of decision trees with the cross-validation. The variable importance of learning behavior is shown in Table 3, 4 and 5.

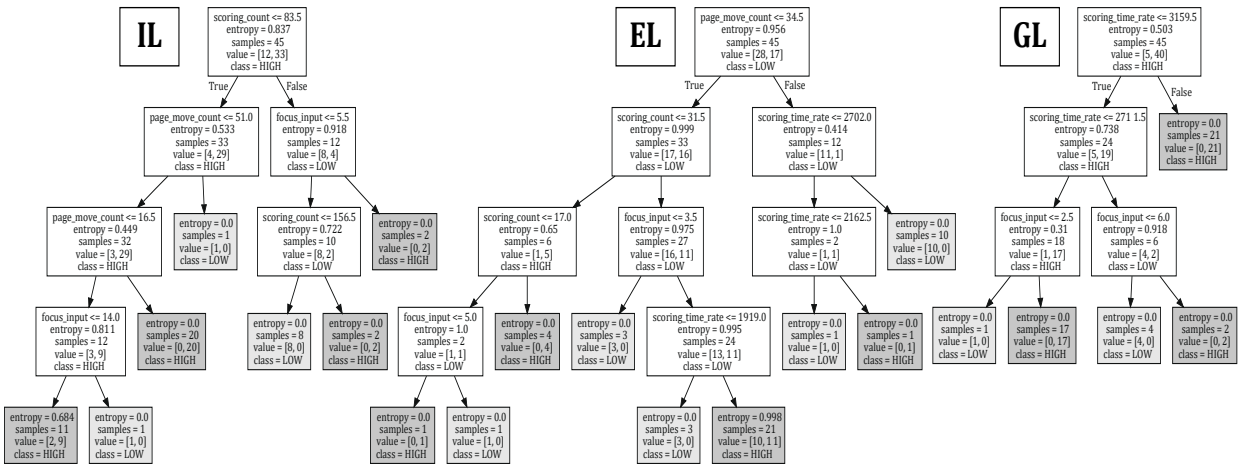

Fig. 3. Decision trees for IL, EL, and GL.

Table 2. Accuracies of Generated Decision Tree

CL	Train data (SD)	Test data (SD)
IL	0.941 (0.017)	0.741 (0.153)
EL	0.811 (0.061)	0.593 (0.083)
GL	0.970 (0.025)	0.889 (0.091)

Table 3. Variable Importance of Decision Tree for IL

Learning behavior	Importance
Scoring requests	0.362
Time transitions for correct answer rate	0.249
Page transitions	0.196
Time until first input	0.193

Table 4. Variable Importance of Decision Tree for EL

Learning behavior	Importance
Scoring requests	0.263
Time transitions for correct answer rate	0.347
Page transitions	0.194
Time until first input	0.197

Table 5. Variable Importance of Decision Tree for GL

Learning behavior	Importance
Scoring requests	0.000
Time transitions for correct answer rate	0.671
Page transitions	0.000
Time until first input	0.329

5. Discussion

This section discusses the effectiveness of the decision trees and influence of the important variables on cognitive load of learners and measures teachers should take.

5.1. Intrinsic Load

The most important variables among the decision tree for IL are the number of scoring request, and the second is time transitions for correct answer rate. In the decision tree for IL shown in Fig. 4, when both of these variables are larger, IL is evaluated as high. Let us focus on leaves of the decision tree that are determined to have a high IL shown in Fig. 4. The leaf of high IL classified with the largest number of subjects is a case that they have few scoring requests and gets a low score of time transitions for correct answer rate. This means that the assignment has a high degree of difficulty that subjects who are under high IL cannot find the correct answer. The node of the second largest number of the subjects having high IL is a case that they have few scoring requests, get a moderate score of time transitions for correct answer rate and take short time until starting to write answers into blanks. In this case, the subjects have been solving while considering the correct answers for the blanks. It is implied that the subjects classified into either of the cases assign working memory inappropriately to IL for engaging in solving the assignments for a long time. Thus, teachers alleviate IL of learners by giving assignments of which difficulty is lower.

5.2. Germane Load

The decision tree for GL consists of only two explanatory variables of time transitions for correct answer rate and time until first input. When the score of time transitions for correct answer rate and time is large, the most subjects having high GL can be identified. That means learners who gain a high correct answer rate early is identified as high GL.

On the other hand, even when the score of time transitions for correct answer rate is low, the subjects are judged as having high GL. These subjects take a long time from focusing the blank to starting input some code fragment. We consider that subjects have combined code fragments before and after the blanks to systematize programming skills.

High GL means that it is in the process to systematize knowledge of programming. Therefore, learners who are evaluated to have a high GL obtain a sufficient learning effect. Teachers do not need to reinstruct

such learners and can let them proceed to more advanced next learning stages.

5.3. Extraneous Load

The entropy of the multiple nodes classified with a large number of the subjects was not sufficiently minimized. This result implies that the four variable of learning behavior cannot evaluate EL of learners. In order to make an evaluation of EL possible, new explanatory variables of learning behavior should be given decision tree analysis. In the decision tree for EL, the most important explanatory variables of learning behavior are the number of scoring request and page transitions. Both behaviors are related to using the automatic scoring system for fill-in-the-blank test. Therefore, we consider that finding another learning behavior related to that improve the accuracy of the decision tree.

It is difficult to find learners who are under the inappropriate cognitive load without this method. The decision trees generated by our method clarify IL and GL early. That allows teachers to take measure to make learners with inappropriate cognitive load appropriate with a light burden.

6. Conclusion

We propose a method to estimate the cognitive load and its factors, and discuss the usefulness of the decision tree generated by datasets obtained in the experiment. There are three types of cognitive load: IL, EL, and GL. The decision tree based on four variables representing features of the learning behavior is proven to be effective to identify IL and GL. We consider that the accuracy of estimating EL may improve, adding another explanatory variable. Teachers can find learners who have inappropriate cognitive load early. Each test or blank is assumed to give different cognitive load. In the future, we classify tests or blanks in viewpoints effective to identify the cognitive load.

References

[1] Okamoto, M., & Kita, H. (2014). A study of novices missteps in shakyo-style learning of computer programming. *Memoirs of the Center for Educational Research and Training, Shiga University 22*, 49-53.

[2] Mohamed, S., Hamilton, M., & D'Souza, D. (2011). Understanding novice programmer difficulties via guided learning. *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, *ITiCSE '11* (pp. 213-217), New York, NY, USA. ACM.

[3] Schnotz, W., & Kürschner, C. (2007). A reconsideration of cognitive load theory. *Educational Psychology Review*, *19(4),* 469-508.

[4] Sweller, J., Ayres, P., & Kalyuga, S. (2011). *Cognitive Load Theory*. Springer.

[5] DeLeeuw, K. E., & Mayer, R. E. (2008). A comparison of three measures of cognitive load: Evidence for separable measures of intrinsic, extraneous, and germane load. *Journal of Educational Psychology*, *100,* 223-234.

[6] Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, *22(2),* 123-138.

[7] Sweller, J., Merrienboer, J., & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, *10(3),* 251-296.

[8] Haapalainen, E., Kim, S., Forlizzi, J. F., & Dey, A. K. (2010). Psychophysiological measures for assessing cognitive load. *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, *UbiComp '10* (pp. 301-310). New York, NY, USA. ACM.

[9] Garner, S. (2004). The CLOZE procedure and the learning of programming. *Proceedings of International Conference on Learning*. Granada, Spain.

[10] Morrison, B. B., Dorn, B., & Guzdial, M. (2014). Measuring cognitive load in introductory cs: Adaptation

of an instrument. *Proceedings of the Tenth Annual Conference on International Computing Education Research*, *ICER '14* (pp. 131-138).

[11] Leppink, J., Paas, F., Van der Vleuten, C. P. M., Van Gog, T., & Van Merriënboer, J. J. G. (2013). Development of an instrument for measuring different types of cognitive load. *Behavior Research Methods*, *45(4),* 1058-1072.

[12] Yousoof, M., Sapiyan, M., & Kamaluddin, K. (2007). Measuring cognitive load — A solution to ease learning of programming. *World Academy of Science, Engineering and Technology International Journal of Computer and Systems Engineering*, *1(2),* 32-35.

[13] Fridman, L., Reimer, B., Mehler, B., & Freeman, W. T. (2018). Cognitive load estimation in the wild. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, *CHI '18* (pp. 1-9). New York, NY, USA. ACM.

[14] Chang, K., Chiao, B., Chen, S., & Hsiao, R. (2000). A programming learning system for beginners — A completion strategy approach. *IEEE Transactions on Education*, *43(2),* 211-220.

[15] Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, *ITiCSE '05* (pp. 14-18). New York, NY, USA. ACM.

[16] Medawela, R. S. H. B., Ratnayake, D. R. D. L., Abeyasinghe, W. A. M. U. L., Jayasinghe, R. D., & Marambe, K. N. (2018). Effectiveness of "fill in the blanks" over multiple choice questions in assessing final year dental undergraduates. *Educación Médica, 19(2),* 72-76.

[17] Asai, S., & Shimakawa, H. (2017). Automatic scoring system of fill-in-the-blank tests to measure programming skills. *Proceedings of the 6th the International Conference on Information Technology and Its Applications* (pp. 23-29).

[18] Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, *ITiCSE '05* (pp. 14-18). New York, NY, USA. ACM.

[19] Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming — Views of students and tutors. *Education and Information Technologies*, *7(1),* 55-66.

So Asai was born in Aichi, Japan in 1993. He received the bachelor of engineering in information science and technology from Ritsumeikan University, Shiga, Japan in 2017. Since 2017, he has been a student of Graduate School of Information Science and Engineering at Ritsumeikan University. He is currently interested in education engineering and data science.