International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 05 - Issue 02 || February 2019 || PP. 43-51

Viewing Understanding Faults in Programming Learning from Cognitive Load

So Asai[1], DinhThi Dong Phuong[2], Fumiko Harada[3], Hiromitsu Shimakawa[1]
[1]*Graduate School of Information Science and Engineering, Ritsumeikan University, Japan*
[2]*Paracel Technology Solutions Co., Ltd, Vietnam*
[3]*Connect Dot Ltd, Japan*

Abstract: We propose a method to identify factors of cognitive load causing understanding failures in learners engaging in programming learning, through the analysis of their learning behavior. It is important to make them acquire programming skills with clarifying causes of understanding failures of learners. In general, since the concepts of programming is difficult for learners. The difficulty is primarily affected by three types of cognitive load: intrinsic load (IL), extraneous load (EL), and germane load (GL). Since many learners belong to one class, it is difficult for teachers to distinguish learners with appropriate cognitive load from ones with inappropriate cognitive load. In order to find learners with inappropriate cognitive load, our method generates decision trees classifying learners with their learning behavior when they solve fill-in-the-blank tests. The decision trees are used to identify factors of each type of cognitive load. An experiment shows the decision trees our method generated enable to view the factors affecting inappropriate IL and GL. This result clarifies the learning factors affecting inappropriate cognitive load of learners and the teachers can address to make their cognitive load appropriate.
Keywords: Data mining, decision tree analysis, e-learning, programming learning.

1. INTRODUCTION

There are students who cannot master programming, even though they enter universities specializing in information and communication technologies. In the modern society where the information and communication technologies are indispensable, it is important to make them acquire programming skills, clarifying causes of their understanding failures.

Cognitive load [1] plays an important role to account for the understanding of learners in programming. Understanding failures come from inappropriate cognitive load. In programming learning, cognitive load is an important factor to consider, and closely related to understanding and schematization of programming skills of learners. It is not bad for learners to have cognitive load in learning programming abilities. Rather, proper cognitive load should be welcomed. Instructors should prepare an environment where appropriate cognitive loads are imposed on learners. However, since many learners belong to one class, it is difficult for teachers to distinguish learners with appropriate cognitive load from ones with inappropriate cognitive load.

The method proposed in this study identifies factors of cognitive load causing understanding failures in learners engaging in programming learning, through the analysis of their learning behavior. The method generates decision trees classifying learners using their learning behavior when they solve fill-in-the-blank tests. The decision trees are used to identify factors of their cognitive load. The identified factors enable teachers to grasp states of their understanding. The teachers can take measures to convert inappropriate cognitive load of learners into appropriate one with minimal efforts.

2. COGNITIVE LOAD IN PROGRAMMING LEARNING

2.1. Cause of understanding faults for programming

Requirements for programming skills are both understanding the concepts and writing appropriate programs. Learners must be able to organize various knowledge of programming themselves. Principal factors that learners cannot achieve them are difficulty of the concepts and ways of thinking programming [2]. Learners do not know what they write to assignments or examinations. In other case, learners who have enough knowledge of programming cannot consider concrete processes of programs they have written actually. Such learners may possibly fail to learn programming and avoid challenging to programming. Their teachers find them early to prevent them failing to learn programming.

2.2. Cognitive load to affect learning

Working memory is a pseudo memory that is defined as process capacity and is consumed when thinking or holding something. The capacities of working memory are different according to each individual.

When learners study identical elements repeatedly in working memory, they are organized as a schema. The elements that have once become a schema are systematized together with their usage so that no cognitive load on use.

Cognitive load affects understanding failure caused by difficulty of programming concepts [3, 4]. In cognitive load theory, usage of working memory is classified with three types of the followings [5, 6].

Intrinsic Load (IL)

IL is due to abilities of learners and inherent difficulty of the assignment. When the capacity of working memory is small and learners feel a tremendous degree of difficulty for the assignment, this load is high.

Extraneous Load (EL)

EL is affected by surrounding learning environment. Since EL depends on the quality of teaching materials and lectures used by learners. teachers need to design the materials and the lectures to reduce this load.

Germane Load (GL)

GL is a cognitive load related to schematization of learning contents. Being imposed this load means that learners organize the learning content by themselves as knowledge. Learners are encouraged to have this load.

If it is the ideal learning situation that learners continue to acquire new knowledge, it is desirable that cognitive load is low in IL and EL, and GL is high in which knowledge can be systematized. This study refers to it as an appropriate state of cognitive load. Teachers must endeavor to keep cognitive load state of learners appropriate. However, difficulty degree of the learning content and effect of lesson design differ amount learners in programming classes and exercises conducted as mass lectures. Therefore, it is difficult to estimate cognitive load for each learner. Even more judging visually on the spot during the class is nearly impossible.

2.3. Measurement of cognitive load

Many researchers are engaged in study to measure cognitive load [7, 8]. Several methods are proposed to measure cognitive load, and these methods that certain usefulness is shown in various fields. Morrison et al. [9] measured cognitive load of learners in the classes. This paper applies a method that established by Leppink et al. [10] to programming learning. Cognitive load is evaluated by 10 questions obtaining IL, EL and GL. Learners answer by 11 scales for the questions. These question items are correlated to measure IL, EL, and GL. However, the object of the evaluation of the cognitive load is the whole learning. This method does not clarify factors of the cognitive loading of learning.

Yousoof et al. [11] proposed a method to measure and reduce cognitive load from its accumulation by dividing visual information and character information in programming class. This method mainly focuses on EL. It does not fully consider IL or GL, which is caused by learners. Considering three types of cognitive load, it is necessary to clarify factors of essential understanding failure based on learning behavior.

Fridman et al. [12] measure cognitive load during driving vehicle without wearing any sensors. This study judges cognitive load by applying deep learning to movies of eye movements in real time from datasets of cognitive load due to eye movements. This study does not distinguish between cognitive load. A requirement of cognitive load in programming learning is to discriminate IL, EL and GL since IL and EL, which decrease learning efficiency, should be low and GL, which schematizes learning, should be high.

It is not ideal to attach physical sensors to obtain cognitive load to learners in programming learning. Since there are many learners, attaching sensors to all of them makes huge cost. Also, there is possibility that sensors may have inappropriate influence on learning efficiency. A contribution of our study is to identify factors of cognitive load by estimating three types of cognitive load without specific sensors.

3. ESTIMATING COGNITIVE LOAD FACTOR FROM LEARNING BEHAVIOR

3.1. Cognitive load factor estimation method

This study targets learners who study procedural programming language such as C language. We estimate factors of the cognitive load from their learning behavior. Each factor brings different effects on the learning behavior. We propose a method to estimate the factors of cognitive load, analyzing features of the learning behavior. Fig. 1 shows the overview of the method. The method collects datasets of learning behavior when past learners answer fill-in-the-blank tests in order to estimate cognitive load and its factors. The cognitive load of past learners is acquired by the existing method [9]. Based on learning behavior and cognitive load of past learners against fill-in-the-blank tests, this method constructs a decision tree. This decision tree relates learning behavior of past learners to factors of their cognitive load. The decision tree provided with learning behavior of new learners indicates a factor of their cognitive load. The method promotes teachers to investigate

whether appropriate cognitive load is imposed on the learner in programming learning It points out learners who have inappropriate cognitive load. It also shows the factors of the inappropriate cognitive load on those learners. Depending on the factors, the teachers can take appropriate measures to those learners.

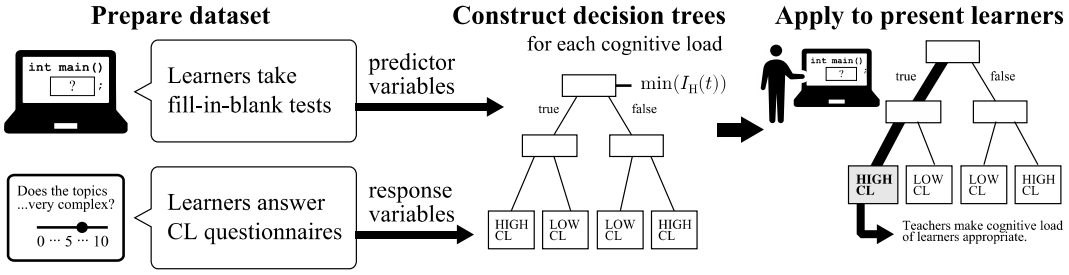


Figure1: Method overview

3.2. Collecting learning behavior and cognitive load

The method collects learning behavior by learners to estimate factors of cognitive load. We focus on behaviors of learning programming when learners answer fill-in-the-blank test.

Fill-in-the-blank test is frequently conducted in order to measure learning effects of programming classes [13]. Used for measuring learning effects, fill-in-blank test can also be used for measuring cognitive load directly and indirectly. The fill-in-the-blank test discriminate understanding level of learners by correctness, because its parts to be answered are limited. Furthermore, fill-in-the-blank test is less likely to be speculated to answers than multiple choice problems [14]. Generally in learning with fill-in-the-blank test, learners write answer on a distributed paper that shows some blanks. A teacher collects the papers to score them and notifies the learners of the result later. Not being notified of the result frequently, learners proceed to next learning stage with uncertain learning effects.

Fill-in-blank test in the method are scored with the automatic scoring system, which is a web application implemented online [15]. The system scores answers for each blank that learners give automatically and notifies correctness for the answers immediately. Learners using the system are allowed to reconsider their answers many times until their answers become correct within the time limit. The method collects consuming time, histories of answer changes, number of scoring with the system. Concretely, the system records number of scoring request, answers learners submitted and elapsed time from starting test as learning behavior which is taken by learners at answering. We focus on 4 types of specific behaviors as the followings.

Scoring request

Learners can check whether each of their answers for fill-in-the-blank test is correct many times. Not being able to process a lot of tasks, learners cannot come up with various answers. Thus, we consider that such learners request scoring rarely. Meanwhile, not fully understanding the application usage and statements of the assignments, learners with high EL do not frequently request scoring. Features to request scoring many times is aimed at letting learners think and learn the correct answer to find cognitive load.

Time until initial fulfillment

This is time from leaving the cursor in the blank to starting to fulfill code fragments. Learners must consider an appropriate processing and convert it into a program to find the correct answer. Learners with high IL that difficult to apply to programs take a lot of time to do. The average of this time is given to the method as score.

Page transitions

In learning with fill-in-blank test, multiple assignments are given learners in the same session in order to measure understanding of learners for various subjects. Hence the learners can choose the order of solving assignments, and also switch them halfway. Finishing answering for all blanks of the assignment or giving up, learners move to other assignment. Learners who feels a high degree of difficulty for the assignments transition more assignments.

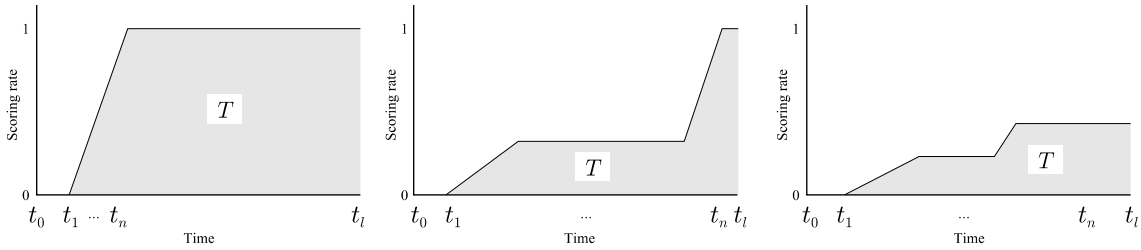
Time transition of correct answer rate

Learners conceive correct answers for each blank with attempting scoring. The correct rate of learners gradually increases. Learners who has understanding for the assignments can answer all of them quickly and get full or close correct marks. Other learners who has lack of understanding for the assignments gets correct

answers over time or cannot answer correctly at all. Thus, it is anticipated that the time transition of the correct answer rate affects for discriminating cognitive load. In order to evaluate centrally, the method quantifies this behavior by the following formula:

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

where n is the number of scoring request of the learner, p_k is a correct answer rate at the kth scoring, t_k is the elapsed time at kth scoring from t_0 and t_l is time deadline of answering. k = 0 is the state at the start time of answering, defined as $t_0 = 0$ and $p_0 = 0$. The score makes simultaneous evaluation of the correct answer rate of a learner as the product and the time reaching it possible like Fig. 2. In the case that correct answer rate becomes high early, the score is to be larger as shown in Fig. 2 (a). Conversely, low correct answer rate and long answering time make the score small as shown in Fig. 2(b)(c). The automatic scoring system for fill-in-the-blank test can collect or calculate these behaviors.



(a) learner answers quickly (b) learner answers over time (c) learner answers poorly

Figure 2: Examples of time transition of the correct answer rate

On the other hand, method use a cognitive load measurement questionnaire [5] in order to identify three types of cognitive load of learners. Learners answer the questionnaire which consists of 10 questions to be measured cognitive load. The questions are classified into 3, 3 and 4 items which are ask about IL, EL, and GL, respectively.

In the method, qualifiers that clarify the question content and its target are added to each statement of the cognitive load measurement questionnaire. Learners evaluate each the question in 11 step scale. In the case of strong agreement for the question, learners mark large number. When the scores of the question item are large, the learner is judged to have high degree of cognitive load.

3.3. Generating decision trees identifying cognitive load

The method generates decision trees to identify between learning behavior and cognitive load evaluated by the questionnaire. Explanatory variables and objective variables of the decision trees are the learning behavior and the cognitive load. Decision trees are generated for each of three types of cognitive load since the purpose of our study is to clarify factors for each cognitive load. The features to compose the decision trees are determined so as to bisects the learners based on the learning behavior on fill-in-the-blank test. The decision tree consists of nodes based on the features. Impurity in each node of the decision tree is small and information gain is minimized. Impurity in each node is represented with entropy $I_H(t)$which by the following formula:

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

where $P(i|t)$ is the ratio of the number of child node i branched from node t. Also, the difference of entropy before and after branch is used to maximize the information gain. The information gain $G(t)$ at the node$t$ is obtained by the following formula:

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

In order to prevent over-fitting of the decision trees, nodes whose information gain is less than the threshold value is set as leaves. Cognitive load of learners who match the branching condition of learning behavior are given to each of the leaf node as objective variable. Nodes on the path to high cognitive load are factors of learning behavior which affect cognitive load.

3.4. Identifying cognitive load of learners

Present learners similarly solve fill-in-the-blank test with the automatic scoring system. Learning behavior that they have acted is checked against the decision trees. Cognitive load of the learner is the leaf node

that is reached. Detected high or low cognitive load is notified to his teachers. When IL or EL is high or GL is low, the teacher follows up him to make his cognitive load inappropriate. Learners can understand programming more sufficiently. and will not fail learning to programming while burdens of teachers can become minimum. Teachers can focus on preparing lectures that can provide higher educational effect.

4. EXPERIMENT

4.1. Conditions in experiment

We conducted an experiment to confirm that the proposed method can identify factors of cognitive load of learners. The experiment aims the following:

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

Subjects are college students of College of Information and Technology, Danang University, Vietnam. They are learning programming in C. All of the students are second grade of the college. They have already finished beginners programming course using C language. At the time of the experiment, there are various students in interests and abilities to C programming. Since almost of the students can read English sentences well, materials of the experiment were provided in English. Subjects solved five assignments which mainly related to a two-dimensional array, a pointer, a structure, a linked list, and sorting algorithm. In general, it is difficult to master them in C language programming. There would be a difference in understanding of learners who study them [16,17]. The assignments were chosen to obtain unbiased data. The several code fragments in the assignments are blanked out as fill-in-the-blank tests. Each assignment was selected from the ones used in an actual programming exercise class in Ritsumeikan University. The quality of the assignments is assured by more than one teachers.

The subjects solved the assignments on a website implementing the automatic scoring system for fill-in-the-blank tests explained in Section 3.2. The subjects make access Web sites with browsers they are familiar with. Since all subjects logged in to the Web system with separate user accounts, it is possible to record the individual history of code fragments every subject filled in each blank. The subjects can answer as many times as they want within the time limit. Moreover, they can solve from any assignment. They also can change assignments into others any time. Learning behavior of subjects is stored on the server using asynchronous communication with Web beacons immediately as soon as they take predetermined actions such as pressing buttons to request scoring. After the time limit has elapsed, we forced the subjects terminate solving the assignments. Subsequently, they answer the questionnaire to measure the cognitive load.

4.2. Result

In the experiment, we collected data set of answers and learning behaviors from 54 subjects. 3 out of the 57 subjects are excluded due to insufficient number of answering. The proposed method generated decision trees corresponding to IL, EL, GL from the data set. In the generation, we used four types of learning behavior as explanatory variables explained in Chapter 3, while results of the questionnaire on IL, EL, GL as objective variables, with 2 level values of high or low. In order to verify the accuracy of the decision trees with 6-fold cross validation, datasets are divided into 2 groups: for training and for test. Table I shows the results of the accuracy of decision trees with the cross validation. Table II, III and IV represent the average of the variable importance of each kind of learning behavior.

Table I: Accuracies of generated decision tree

Cognitive Load	***Training 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 II: Variable importance of decision tree for IL

Variable of learning behavior	***Importance***
Scoring request	0.362
Time transition of correct answer rate	0.249
Page transitions	0.196
Time until initial fulfillment	0.193

Table III: Variable importance of decision tree for EL

Variable of learning behavior	***Importance***
Scoring request	0.263
Time transition of correct answer rate	0.347
Page transitions	0.194

Table IV: Variable importance of decision tree for GL

Variable of learning behavior	***Importance***
Scoring request	0.000
Time transition of correct answer rate	0.671
Page transitions	0.000

5. DISCUSSION

Fig. 3, 4, and 5 show decision trees generated for IL, GL, and EL, where edges deeper than 5 are pruned. The section confirms the effectiveness of the decision trees for each kind of cognitive load of learners. It also discusses the influence of the important variables on each kind of them. Eventually, it proposes measures teachers should take on learners from viewpoints of the cognitive load.

5.1. Intrinsic load

In the decision tree for IL, the most important variable is the number of scoring request. The second is the time transition of the correct answer rate. The decision tree for IL shown in Fig. 3 judges that learners have high IL if the 2 variables shows high values in them. Let us focus on leaves where learners are judged to have high IL in Fig. 3.

The more learners with high IL a leaf divides, the more powerful it is. The most powerful leaf corresponds to the case where the number of scoring times is small and the value of the time transition of the correct answer rate is small. It means that the assignment was so difficult that the subjects cannot find the correct answer because of high IL. The second most powerful leaf indicates the case where they issue few scoring requests, have moderate values of the time transition of the correct answer rate, and take a short time until initial fulfillment into blanks.

The subjects in this case solve assignments, considering the correct answers for the blanks well. In both cases, they engage in solving the assignments for a long time. It implies that the subjects assign inappropriately large part of working memory to solving the assignment. When learners suffer from the difficulty of specific assignments too long time, high IL occurs in those learners. The assignments might take off chances of schematization of programming skills from the learners. Teachers should alleviate IL of learners, modifying the assignments so that their difficulty should get lower.

5.2. Germane load

The decision tree for GL consists of only two explanatory variables: the time transition of the correct answer rate and the time until the initial fulfillment. The most powerful leaf in decision trees for GL distinguishes subjects whose value of the time transitions of the correct answer rate is large. It implies learners who early achieves high correct answer rate are identified as those of high GL.

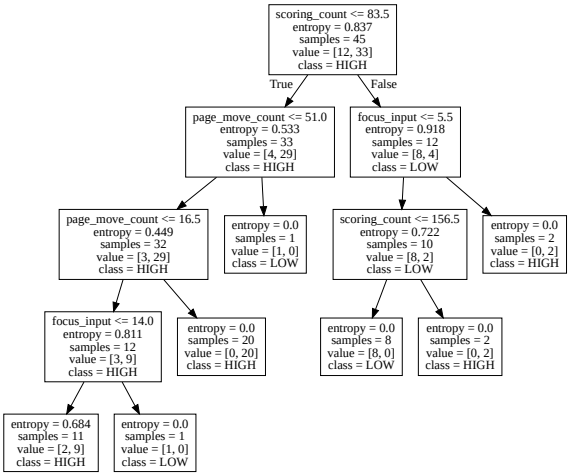


Figure 3: Decision tree for IL

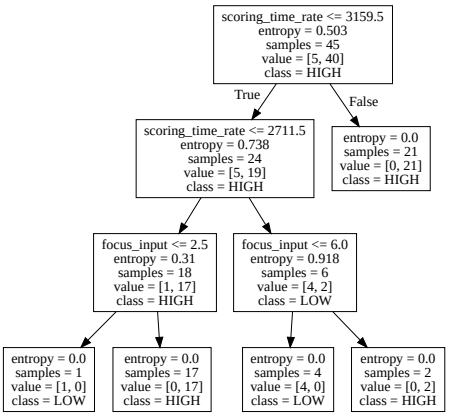


Figure4: Decision tree for GL

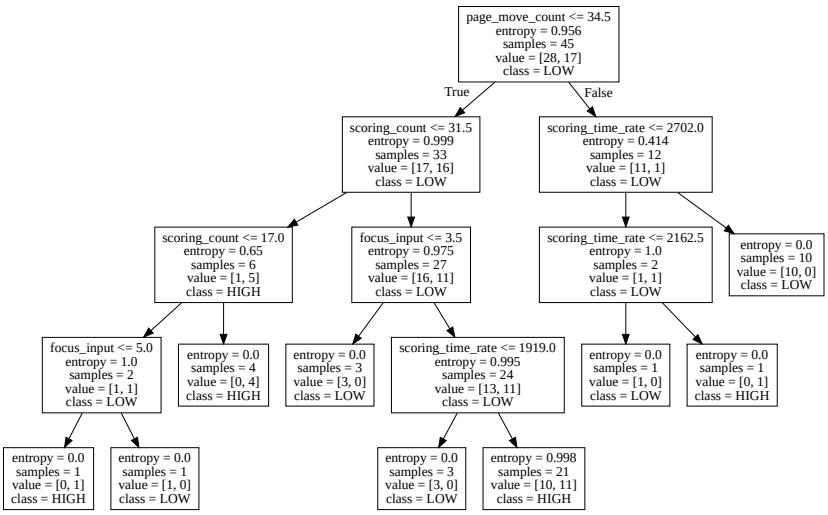


Figure5 Decision tree for EL

On the other hand, some subjects are regarded as ones of high GL, even when the score of the time transition of the correct answer rate is low. These subjects take a long time from focusing the blank to starting fulfillment of some code fragment. In fill-in-the-blank tests, learners have to understand the code fragments around the blank to find a suitable one for the blank. For subjects who took time to fill in, it is considered that a large GL occurred to incorporate the pattern of the code fragments around the blank into their own knowledge system. High GL means that learners are in the process to systematize knowledge of programming. Learners evaluated to have a high GL are expected to obtain sufficient learning effects. Teachers do not have to instruct such learners. They should lead them to more advanced assignments in the next learning stages.

5.3. Extraneous load

The entropy was not minimized sufficiently at multiple nodes where most learners were classified in Fig. 4. It means that the four variable of learning behavior cannot evaluate EL of learners. In order to make it possible to evaluate EL, we should incorporate new explanatory variables to generate decision trees.

In the decision tree for EL, the top 2 important explanatory variables are the number of scoring request and the page transitions. Both behaviors are related to usage of the automatic scoring system for fill-in-the-blank tests. The important variables indicate that high EL occurs because of the use of the automatic scoring system. The subjects might be confused to use the system, because they are not familiar with the system. We consider that another kind of learning behavior related to the use of the automatic scoring system could improve the accuracy the decision tree.

The decision trees generated the proposed method generates clarify the cognitive load of IL and GL early. The method is indispensable to find learners suffering from inappropriate cognitive load. It allows teachers to accommodate inappropriate cognitive load of learners into appropriate one with low burden.

6. CONCLUSION

This paper proposes a method to estimate the factors of the cognitive load of learners engaging in programming learning. An experiment reveals the effectiveness of the decision tree.

The decision trees which take four variables of learning behavior identify IL and GL. The accuracy to estimate EL is expected to be improved by addition of other explanatory variables. The learning behavior used in the decision trees can be easily obtained with Web programming technologies. Since the method clarifies the cognitive load of learners from the learning behavior, teachers can find learners who have inappropriate cognitive load as soon as they suffer from it.

In the future, we focus on individual assignments or blanks in order to refine the ability to estimate factors of cognitive load.

REFERENCES

[1] J. Sweller, P. Ayres, and S. Kalyuga. *Cognitive load theory*. (Springer,2011).

[2] S. Mohamed Shuhidan, M. Hamilton, and D.D'Souza. Understanding novice programme rdifficulties viaguided learning. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 213–217, New York, NY, USA, 2011.

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

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

[5] K. E. De Leeuw and R. E. Mayer. 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, 2008.

[6] W. Schnotz and C. Kürschner. Are consideration of cognitive load theory. *Educational Psychology Review*, 19(4):469–508, 2007.

[7] E. Haapalainen, S. Kim, J. F. Forlizzi, and A. K. Dey. Psycho-physiological measures for assessing cognitive load.In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp'10, pages 301–310, New York, NY, USA, 2010.

[8] F. Paas, J. Tuovinen, H. Tabbers, and P. van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist*, 38(1):63–71, 12003.

[9] B. B. Morrison, B. Dorn, and M. Guzdial. Measuring cognitive load in introductory CS: adaptation of an instrument. In *ICER '14 Proceedings of the tenth annual conference on International computing education research*, pages 131 –138, 2014.

[10] J. Leppink, F. Paas, C.P.M. Vander Vleuten, T. VanGog, and J.J.G. VanMerriënboer. Development of an instrument for measuring different types of cognitive load. *Behavior Research Methods*, 45(4):1058–1072, Dec 2013.

[11] M. Yousoof, M. Sapiyan, and K. Kamaluddin. Measuring cognitive load–a solution to ease learning of program- ming. *World Academy of Science, Engineering and Technology International Journal of Computer and Systems Engineering*, 1(2):32 – 35, 2007.

[12] L. Fridman, B. Reimer, B. Mehler, and W. T. Freeman. Cognitive load estimation in the wild. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI'18, pages 652:1–652:9, New York, NY, USA, 2018.

[13] K. Chang, B. Chiao, S. Chen, and R. Hsiao. A programming learning system for beginners – a completion strategy approach. *IEEE Transactions on Education*, 43(2):211–220, May2000.

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

[15] S. Asai and H. Shimakawa. Automatic scoring system of fill-in-the-blank tests to measure programming skills. In *Proc. of the 6th the International Conference on Information Technology and Its Applications*, pages 23 – 29, 2017.

[16] I. Milne and G. Rowe. Difficulties in learning and teaching programming— views of students and tutors. *Education and Information Technologies*, 7(1): 55–66, March 2002.

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