Information Engineering Express
International Institute of Applied Informatics
2022, Vol. 8, No. 1, IEE622

Evaluating the Usefulness of C Language Learning Support System as a Learning Analytics Tool

Akifumi Ohno [*], Shimpei Matsumoto [*], Kosuke Kaida [†]

Abstract

We have been developing a learning support system of C programming language for novices named Hello C. This paper aims to describe Hello C's details, such as its concept and functions. Besides, this paper aims to verify the possibility of Hello C as a learning analytics tool for improving programming education. Hello C has an analyzer and management system for instructors, and they can use these functions to monitor and analyze the learners' activities. We examined the usefulness of Hello C as a learning analytics tool using the coding logs from an actual lecture. We could detect the signs of the "impass", which is defined as the time interval data that the learner did not compile. From this analysis result, we concluded that Hello C is useful for supporting learners who cannot perform the instructor's activities.

Keywords: programming education, C language, learning analytics, impass.

1 Introduction

In recent years, the importance of programming education in higher education has been increasing [1]. On the other hand, the current educational environment seems insufficient for all learners because there are a certain number of learners who cannot learn programming [2]. Building a learning environment easy for learners to learn would be one of the most effective ways to improve programming education better. It is also important that the system should provide functions easy for learners to learn and for its instructors to teach. One of the essential factors as an easy environment for instructors to teach is the progress management function where the learning history/learning activity data are easily obtainable. It helps instructors to grasp the level of understanding of learners as necessary. In addition, a user-friendly system for learners is also essential. If learners are given a difficult system to use, they will not be able to perform the desired learning activities. Naturally, the learning log will not be desirable either. Therefore, to collect appropriate data from the learners, it needs to provide a user-side system facilitating proper learning activities intended by instructors. Considering this point of view, we think it is inevitable to build an environment

[*] Hiroshima Institute of Technology, Hiroshima, Japan
[†] Hello C Development Community, Tokyo, Japan

that makes it easy for learners to learn and develop a system that can simultaneously collect and analyze the learning log.

Based on the background mentioned above, we have developed a learning support system of C programming language for novices named Hello C [3][4][5]. Hello C is an editor application for C programming on Microsoft Windows OS. Hello C provides a coding, compiling, and executing environment of C programming that allows learners who are not familiar with elementary computer operation to make it easier to engage in its learning. In addition to that, Hello C has a management system for instructors named Hello C Server[6]. Hello C Server is to support the delivery of assignments and the progress management of learners. These two systems, called Hello C client and server, are server-client systems that work together (hereinafter collectively called Hello C). Hello C is designed as a programming environment specialized for basic C language learning. Hello C is a learning application for only beginners, and is not suitable for full-scale software development. On the other hand, Hello C is now able to provide a unique UX because it is specialized for a specific user segment. In the case of a typical class of C language in university, a specialized environment such as Linux or a rich integrated development environment such as Visual Studio is usually used. However, these professional and advanced learning environments are challenging for some learners [7]. Some learners who are new to programming are not familiar with computers because they do not have much opportunity to use them daily. Moreover, some beginners do not fully understand OS and the basic concept of file management. Besides, some do not know how to use the editor and are not used to typing. Therefore, such environments would be inappropriate for them. Another problem is that programming requires various tasks such as setting up a development environment, file operations, and tasks required to write and execute a program. These might hinder the learner's intrinsic programming learning, and they cause a decline in learners' motivation for learning. Then, the main purpose of developing Hello C is to solve these problems. Hello C mainly focuses on learners who are not used to programming. They can focus on essential learning by experiencing an easy-to-use UI of Hello C. Besides, Providing a user-friendly UI is a great advantage for the instructors. Because a user-friendly UI allows the learners to focus on the instructor's task, and it leads to reduce the number of unintentional logs. Less noisy logs enable instructors to reflect appropriately.

This study describes the details of Hello C and verifies the possibility of Hello C as a learning analytics tool. Hello C has an analyzer and management system for instructors, and they can use these functions to monitor and analyze the learners' activities. We examined the usefulness of Hello C as a learning analytics tool using the coding logs from an actual lecture. We succeeded in detecting the signs of the "impass", which is defined as the time interval data that the learner did not compile. From this analysis result, we concluded that Hello C is useful for supporting learners who cannot perform the instructor's activities.

2 Hello C

The appearance of the Hello C client is shown in Figure 1. Hello C Client includes simple project (source codes) management, advanced editor with auto-completion, static analysis, easy debugging, auto-completion, static syntax check, and assignment management. These features are designed to reduce the burden of non-programming computer operations as much as possible, most of which are assumed as unessential learning tasks while allowing instructors to focus on their intended learning activities. In particular, the assignments man-

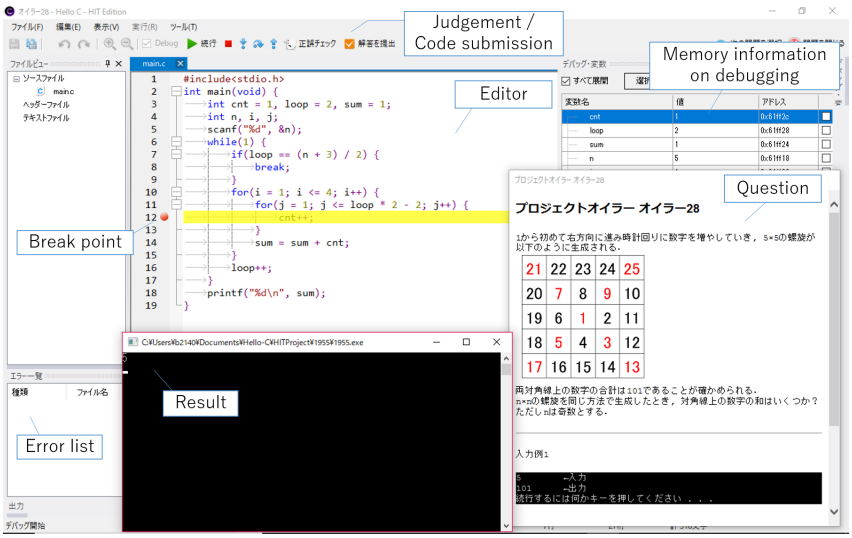

Figure 1: Appearance of Hello C Client

agement function for obtaining, marking, and submitting assignments are unique to Hello C and not found in other C language learning support systems. This function is realized by interfacing with the Hello C server. The Hello C client can send log data of the Hello C server's learning activities when submitting an assignment. The log data includes the learning process information for each learner's task so that pre-and post-learning monitoring and post-analysis are easily possible. Therefore, Hello C server can support instructors' reflection activities and their learning analytics. Hello C client itself is available without the Internet connection if the user writes a program and does not need the assignment management function. The Hello C client is developed using the Visual Basic .NET Framework 4.5. Compilation and execution are performed by gcc for Windows.

Hello C server is a web application and a learning support system for instructors. It provides various APIs for the Hello C client, learners, classes, learning tasks management, and learning analytics. Hello C server runs on Ubuntu 16.04 as OS, Apache 2.4.29 as web server software, and MySQL 10.1.28-MariaDB as a database management system in the LAMP environment, and was developed using PHP 7.1.11 and CakePHP 3.6, a software framework package.

As there are many other C language learning environments for beginners, Hello C would be the original because it has an assignment sending and receiving function and log data collection function. The server-client system implements this service. With the learning data, instructors can also grasp the source code's state per compiling, these judgment results, and compiling timing. The system configuration with Hello C client and server is shown in Figure 2.

Figure 3 shows the details of the log data recording function. After downloading assignment data, the learner can work on these with the editor pane of Hello C Client. Each time a learner compiles, source codes and the message of compiling results are automatically saved. The log data consists of learner ID, assignment ID, source code, timestamp, and the message of compiling result, mainly the error message. When a learner submits an assignment, Hello C Client sends log data and the response data to the assignment to the Hello C server. With this information, instructors can centrally grasp all learners' activities,

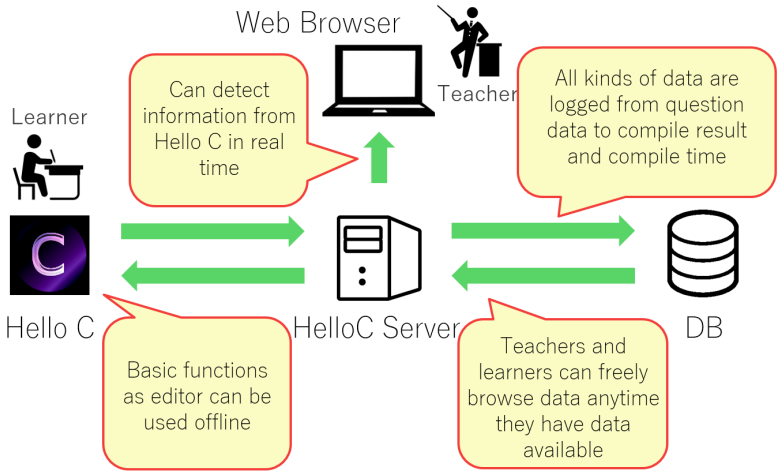

Figure 2: System correlation

so this information would be available for learning support, such as lesson planning and individual guidance. Hello C client and server communicate with JSON. In Hello C, the execution of programs, sometimes it becomes a heavily-loaded process, is the work of the client-side. Since Hello C adopts such a system configuration, the load is not concentrated on Hello C Server. Therefore, Hello C server can run on a general-purpose computer and simple server package like XAMPP.

3 Evaluating the Usefulness for Learning Analytics

3.1 Basic guideline for learning analytics

It has been shown the existence of various types of stumblings, stagnation of learning activities, when beginning learners are learning C programming, and various studies on stumblings have been conducted[10]⁻[12]. Adachi et al. suggested that prediction of stumblings may be possible based on learners' affinity for computers and aptitude tests and pointed out that stumblings in the problem abstraction process tend to reduce learning motivation. Therefore, it is ideal for avoiding learning tasks triggering stumblings because it leads to decreased learning efficiency and motivation. However, presenting only learning tasks that can completely suppress the occurrence of stumbling is not easy. Therefore, research on detecting stumbling blocks has been actively pursued. Suppose we can develop a learning support system that can detect stumblings. In that case, it will be possible to support learners who face stumblings at the appropriate time, which is expected to sustain their motivation to learn and thereby sustain and improve their learning efficiency. This paper focuses on defining the signs of stumbling and detecting the signs of stumbling using the acquired log data as a preliminary step for stumbling detection. Yamashita et al. defined stumblings of programming as "impasse", and stated that programming learners have various impasse during learning [8]. They showed the conditions of impasse to detect, and some of these conditions were about the time intervals between compilations. It suggests the relationship between "impasse" and the time interval between compilations. In this paper, to show the usefulness of Hello C for learning analysis, we focus on impasse on the

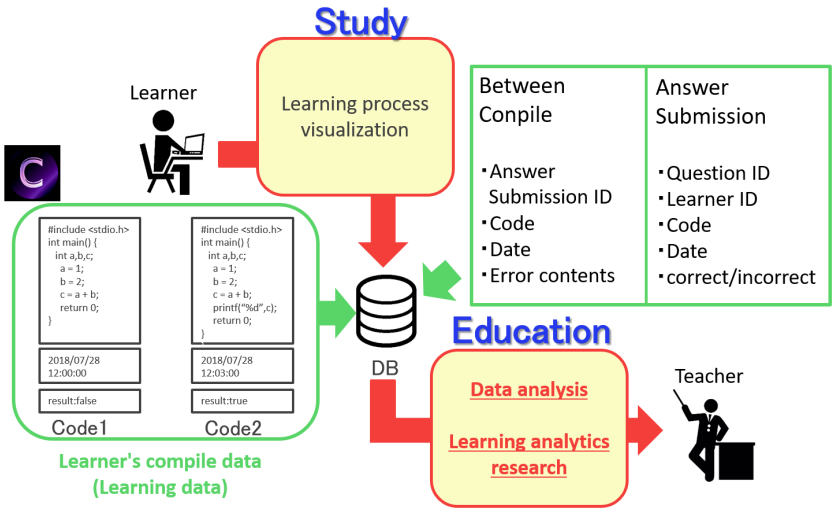

Figure 3: Automatic log data storage function

time interval between compilations. From the log data collected in Hello C, we reveal the variance of the time intervals between compilations for each user and each question and investigate whether the variances are helpful to detect impasses or not. This study used log data of 71 first-year students from an actual class in a college. The total number of assignments was 31. The learners got several questions at the same time once every two weeks as the assignment. The instructor intervened only when the students were in trouble.

3.2 Problem-by-problem analysis

The time intervals between compilations were analyzed for each problem. The behavior of two or more compilations to the same source code is likely to I/O check, miss-operation, or Operations performed suspecting a system-side problem. Therefore, we assumed that two or more compilations to the same source code were not the learning activities performed for program development and would not be directly related to impasse. Then, we removed such compilation activities from the analysis.

Figure 4 and Figure 5 shows the analysis results. Figure 4 focuses only on the percentages of two kinds of time intervals for each question: 2 to 10 minutes and more than 10 minutes. Figure 5 shows the distributions of time intervals for each question as a box plot. Here, the lower bar is the 1st quartile -1.5 IQR, the bottom of the box is the 1st quartile, the line in the box is the median, the top of the box is the 3rd quartile, the upper bar is the 3rd quartile + 1.5 IQR, and the others are the outlier. These results show that the time intervals depend on the questions. The result shown in Figure 5 is a natural tendency since the requirements and level of difficulty vary from question to question. Let us explain the detail of Question ID 8 and ID 23 as examples that have two completely different distributions in Figure 5. The requirement of Question ID 8 is to display text messages multiple times on the screen. Question ID 8 doesn't require any variables, and the distribution of the time interval was small so that it can be regarded as an easy question. Furthermore, the learners had received guidance about the concept of the loop statement just before this assignment. So, the learners might be able to imagine the control structure to fulfill the requirement easily. On the other hand, the distribution of Question ID 23 was fully different from Question

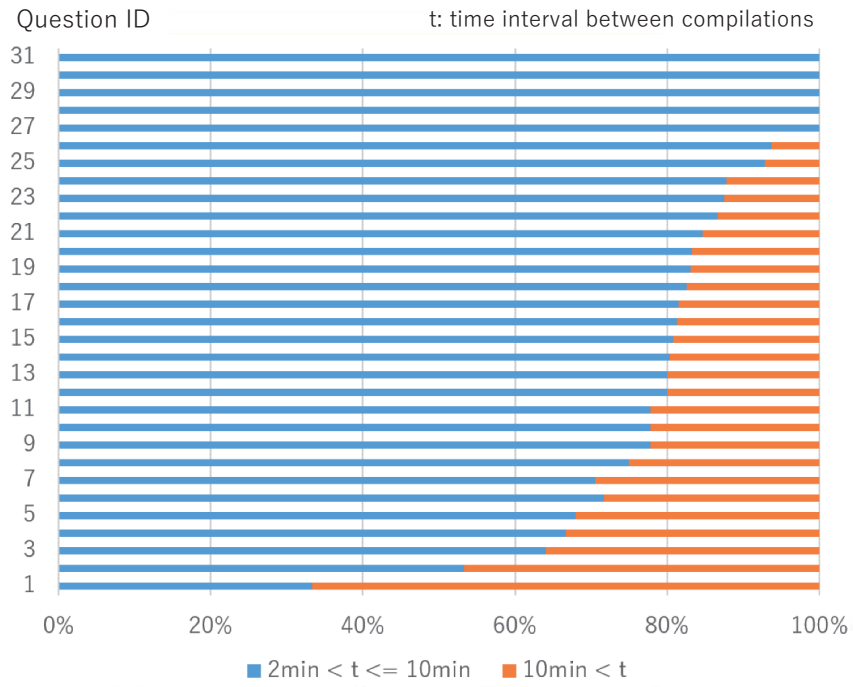

Figure 4: Percentages of two groups of time intervals between compilations for questions

ID 8. The requirement of Question ID 23 is to draw a figure using a double loop. Also, the learners had to apply the structure of the double loop. Seeing its distribution, the control structure required by Question ID 23 would be difficult for many learners. In this way, the time intervals would depend on the difficulty level of the requirement and the timing in which the learners got assignments. Therefore, to detect impasse more properly, it is effective to set the threshold of the time intervals that determines whether the measured time interval is an impasse or not. As shown in Figure 5, the value of an outlier in a question sometimes is normal in another question. Based on this fact, if we assume the value of the outlier as a sign of impasse, we need to set the threshold to define the outlier for each question in advance based on past trends. Fortunately, Hello C can satisfy this requirement. Since Hello C can store all past time intervals in its database, the lecturer has all the information to determine the threshold. Also, Hello C can keep the value of the threshold for each question. Besides, Hello C can realize real-time notification of impasse by using asynchronous communication technologies such as Ajax. In the previous research [8], the threshold to detect impasse was constant for all questions. On the other hand, the analysis result showed the need to give different thresholds for each question to detect impasse more appropriately. Hello C has the capability to provide the different time intervals between compilations for questions, and it would be more desirable to detect impasse. This point is considered the contribution of this study, so we can conclude that Hello C is useful for learning analytics.

3.3 Learner-by-learner analysis

We analyzed the time intervals between compilations for each learner. The analysis results are shown in Figure 6 and Figure 7. The notation of Figure 7is the same as that of Figure 4 and Figure 5. As shown in Figure 6 and Figure 7, the time intervals differed greatly among

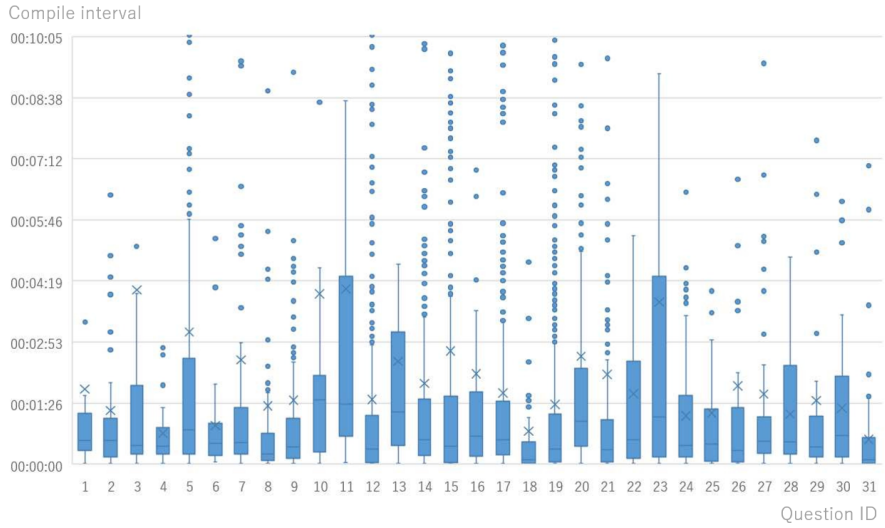

Figure 5: Variances of time intervals between compilations for questions

learners. This may be due to the coding style of the learners.

To investigate the results of Figure 6 and Figure 7 in more detail, we examined the time intervals for each learner; Figure8 and Figure9 show typical behaviors. From Figure8 and Figure 9, first, we can see the trend that the distributions differed depending on questions, even the same learner. Therefore, the time intervals would depend more on the assignments' requirements and the assignments' timing rather than the learner's coding style. Second, we can see the case where an outlier of a learner is normal for others in a question. From this result, to realize the automatic detection of outliers as impasse, we should calculate the threshold for each learner's each question. For example, in learner ID 1101 as shown in Figure 8, this learner usually compiled within 2 minutes or less time interval. Still, the learner had about 8 minutes time interval in question ID 19. Similarly, we can see some outliers in question ID 20 in this learner. However, in learner ID 1083 shown in Figure9, there are no outliers in question ID 20. In question ID 19, the variance of learner ID 1101 is large, but the values of learner ID 1083 are gathered in a narrow range. These results suggest that we should set the thresholds for each learner in each question to detect impasse more appropriately.

Given that the time intervals vary from learner to learner, we need to consider the difficulty level of each question and the characteristics of each learner if we aim to detect the impasse based on the time intervals between compilations. Here shows an example of the procedure for detecting and notifying of impasse. First, when a learner has a time interval t that exceeds a threshold θ_i set for each question i, we calculate a value that indicates the degree. Next, the value of x, the degree to which the learner's average time interval (learning style) deviates from the average of all learners (typical learning style), is calculated by referring to past data and assuming a normal distribution. After that, the threshold is updated by $\theta_i + w_i x$ (w_i is a parameter for adjustment) to θ_i', and if $\theta_i' < t$, it is finally alerted to the instructor as impasse. In this paper, thanks to Hello C, we were able to confirm such a possibility. We can consider that this is firm evidence that Hello C would be an available tool for learning analytics in the future.

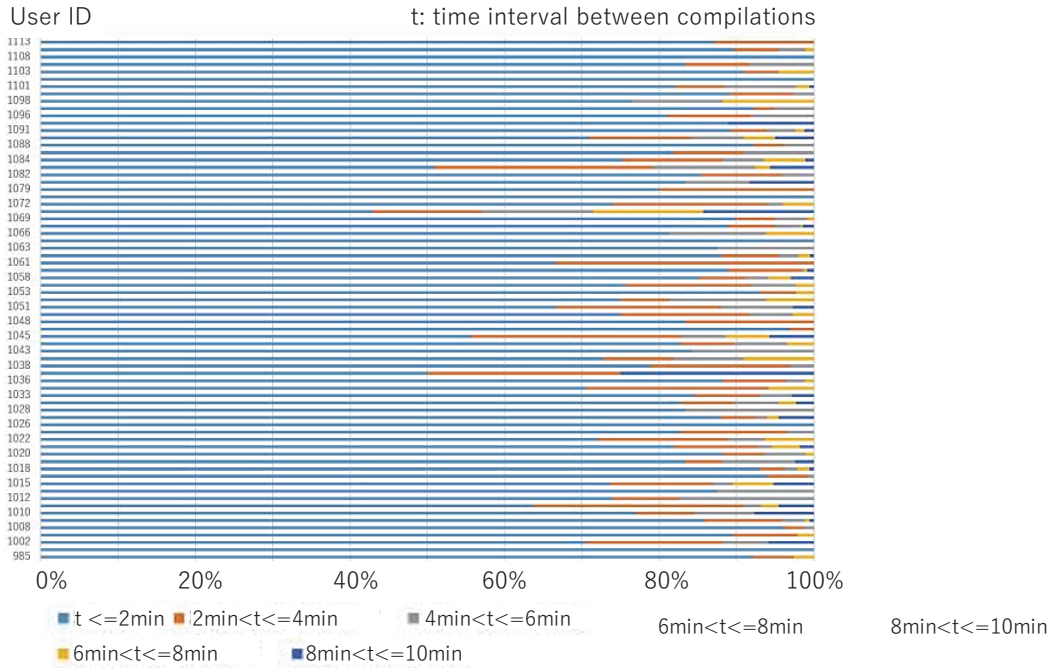

Figure 6: Percentage of five groups of time intervals between compilations for learners

3.4 Defining learning progress and detecting impasse

By analyzing the sufficiency of requirements in each compilation, we evaluated whether the learner was getting closer to the correct answer. Figure 10 and Figure 11 show the chronological changes of the fulfillment of 6 requirements. For example, in Figure 10, the time interval between the compilations at 1 minute 26 seconds and 4 minutes 19 seconds is longer than the others. However, during this time, the learner satisfies two requirements. In such a case, even if the time interval is significantly longer, it can be considered that the learning is progressing and well. Therefore, the time interval with such characteristics should be excluded as a sign of impasse. On the other hand, in Figure 11, the number of fulfilled requirements has not changed during the time interval between the complications at 4 minutes 10 seconds and 7 minutes 08 seconds. So, the learner is not progressing and is likely to be stumbling. Therefore, it would be more effective to alert them as the impasse only in such a situation.

In summary, adding “learning progress” to the time intervals and detecting a learner who “does not satisfy a requirement in the learning task when the time interval is an outlier” as a sign of the impasse is significant as an example to improve the accuracy of detecting the impasse. By the way, Hello C has enough capability for implementing this procedure. Therefore, we can say that Hello C is useful for further supporting learners because it can provide such a service to the instructor.

4 Discussion

First, we examined the distribution of the time intervals between compilations in each question to detect impasse more accurately. From the result, we found that the time intervals are different for each question. Therefore, we examined the time intervals according to the difficulty level of questions. The analysis results suggested that the time intervals might differ

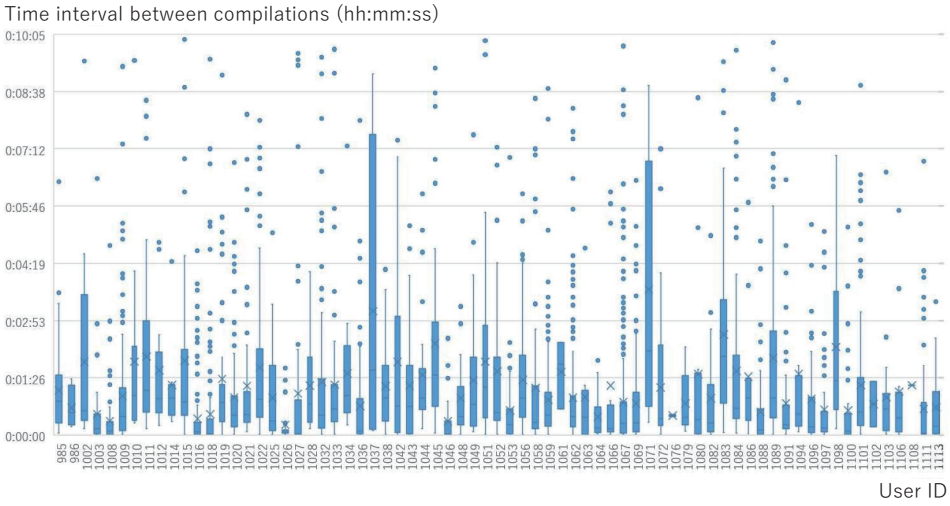

Figure 7: Variances of time intervals betweens compilations for learners

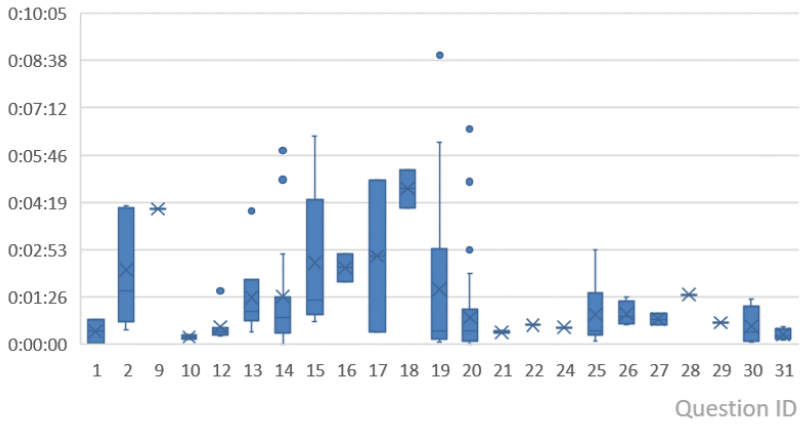

Figure 8: Time intervals between compilations by question for the learner with UserID 1101

in the difficulty level and the timing of assignments given to learners. From the above result, giving different thresholds from the time intervals would help find impasse and supporting learners and their instructor.

Next, we examined the distribution of the time intervals between compilations for each user. From the analysis result, we firstly could find that coding activities greatly vary from learner to learner. For example, there were learners whose time intervals between compilation are always short, and also, there were learners whose time intervals were not stable. This result suggests that many coding patterns exist, and they are absolutely different depending on learners. Therefore, it suggests that we need to give different thresholds of the time intervals for detecting impasse for each problem for each learner. For example, it would be effective to calculate the distribution of time intervals between each learner's compilations from the past coding history and alert it when a newly obtained value of the time interval deviates from the normal value range.

In addition to the above results, although not verified in this paper, we proposed a

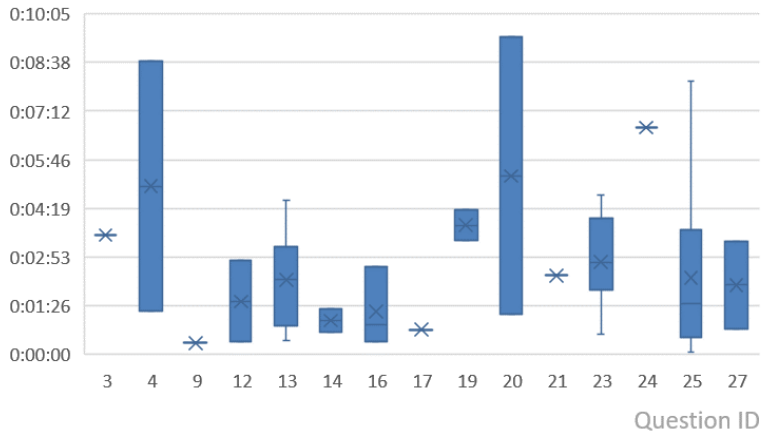

Figure 9: Time intervals between compilations by question for the learner with UserID 1083

method to classify two kinds of time intervals in which learning is progressing and not. We will address the classification of such two kinds of time intervals and implement it to Hello C as the future work.

5 Conclusion

In this paper, we showed the details of Hello C and examined its possibility as a learning analytics tool. First, we explained the system configuration of Hello C and especially emphasized the original capability of Hello C different from the current many other C language learning environments for beginners. Especially, we mainly showed the detail of the assignment sending and receiving function and log data collection function. Based on the explanation, we clarified the contribution of Hello C. Second, to show the usefulness of Hello C for learning analysis, we focused on the time intervals between compilations to detect impasse. From the log data collected in Hello C from an actual class in a college, we revealed the variance of the time intervals between compilations for each user and each question. Then, we investigated whether these data are helpful to detect impasses or not. From the analysis, we found that the variances of the time intervals were different for each question. Therefore, we compared the time intervals for all questions. The analysis results suggested that the time intervals might differ depending on their difficulty levels and the timing provided to learners. From the above, giving different thresholds would be more helpful to detect impasse and also to support learners and their instructor. In the previous research, the threshold to detect impasse was constant. On the other hand, thanks to Hello C, this paper showed that giving the different thresholds for questions is more desirable to detect impasse. Since this result can enhance conventional programming education, we concluded that Hello C is useful for learning analytics.

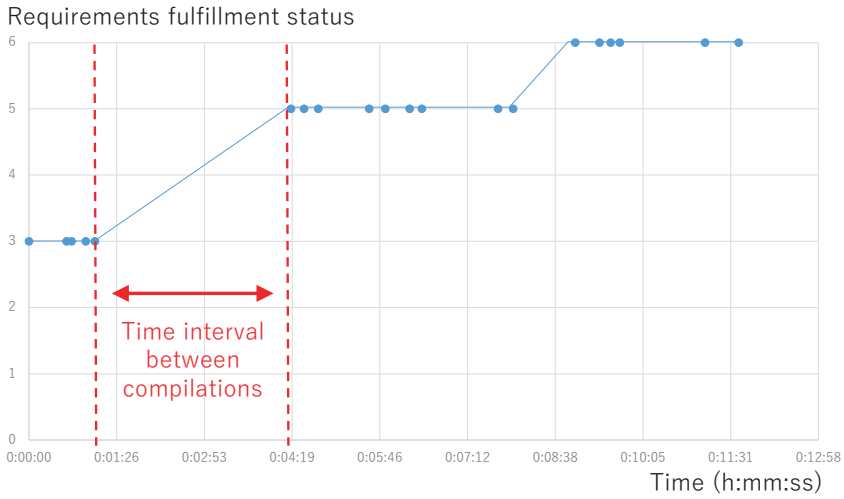

Figure 10: A user whose learning is proceeding

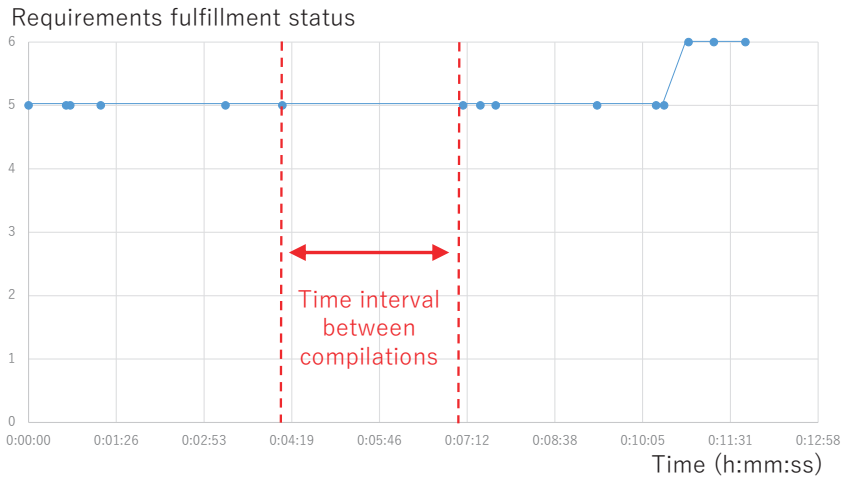

Figure 11: A user whose learning is stalling

Acknowledgment

This work was partly supported by Foundation for the Fusion of Science and Technology 2018 and Japan Society for the Promotion of Science, KAKENHI Grant-in-Aid for Scientific Research(C), No.20K03194, and No.19K02987.

References

[1] A. Aho, Computation and Computational Thinking. The Computer Journal, Vol.55, No.7, pp.832-835 (2012).

[2] J. Bennedsen, M. Caspersen, Failure rates in introductory programming. AcM SIGcSE Bulletin, 39(2), 32-36 (2007).

[3] K. Kaida, M. Ohshita, S. Matsumoto, Developing an Editor of C Programming Language for Collage Students, Proceedings of the Student Research Presentation Meeting o Japanese Society for Information and Systems in Education, pp.207-208 (2018), In Japanese.

[4] M. Oshita, K. Kaida, S. Matsumoto, A Basis Analysis on Novice Programmers with a Server-Client System for Learning C Programming Language, Proc. of The Twenty-Fourth International Symposium on Artificial Life and Robotics 2019, GS5-4, pp.134-137 (2019).

[5] M. Oshita, K. Morita, S. Matsumoto, Developing a Web-Based Programming Editor for Novice Learner, Proc. of 2018 ISCEAS, ISCEAS-0130, pp.399-410 (2018).

[6] M. Oshita, K. Kaida, S. Matsumoto, Developing C Language Programming Learning Environment and its Application for Detecting Exploratory Programming Activities. 2018 IEEE SMC Hiroshima Chapter Young Workshop Conference Papers, pp.48-49 (2018), In Japanese.

[7] M. Okamoto, First programming and stumbling. Information processing vol.56 No6 June 2015, pp.580-583 (2015), In Japanese.

[8] K. Yamashita, T. Sugiyama, S. Kogure, Y. Noguchi, T. Konishi, Y. Itoh, An Educational Support System based in Automatic Impasse Detection in Programming Exercises, Proc. of the 25th International Conference on Computers in Education, pp.288-295 (2017)

[9] M. Oshita, K. Kaida S. Matsumoto, Developing C Language Programming Learning Environment and its Application for Inspecting Characteristic Description Pattern, 2018 IEEE SMC Hiroshima Chapter Young Researchers' Workshop, pp.48-49 (2018), In Japanese.

[10] M. Okamoto and H. Kita, A Study of Novices Missteps in Shakyo-Style Learning" of Computer Programming , Memoirs of the Center for Educational Research and Training, Shiga University, Vol. 22, pp. 49-53 (2014) (in Japanese)

[11] K. Adachi and S. Nakao, An Analysis of Tripping Tendency in Programming Learning, Journal of JSEI, Vol. 10, No. 4, pp. 11-20 (1995) (in Japanese)

[12] K. Yamashita, T. Sugiyama, S. Kogure, Y. Noguchi, T. Konishi, and Y. Itoh, An Educational Support System Based on Automatic Impasse Detection in Programming Exercises, Proceedings of CCE2017, pp. 288-295 (2017)

[13] J. R. Carbonell, Ai in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction, IEEE Transaction on Man-Machine Systems, Vol. 11, No. 4, pp. 190-202 (1970)

[14] S. Vossoughi, B. Bevan, Making and tinkering: A review of the literature. National Research Council Committee on Out of School Time STEM, 1-55 (2014).