Time Series Analysis of Programmer's EEG for Debug State Classification

Toyomi Ishida
National Institute of Technology, Nara College
Nara, Japan
a0932@stdmail.nara-k.ac.jp

Hidetake Uwano
National Institute of Technology, Nara College
Nara, Japan
uwano@info.nara-k.ac.jp

ABSTRACT

Appropriate support for debugging contributes to efficient software development. Several previous studies used bio information such as brain activity to classify the inner-state of programmers without any interruption. In this paper, we measure programmer's brain waves while they comprehend the source code. In an experiment, we analyze difference of time-series brain wave features between success/failure for tasks involving source code comprehension and bug judgement. The result of the experiment showed the participants who successfully understand source code significantly increased power spectrum of α and β wave with the time passage.

KEYWORDS

EEG, Program comprehension, Debugging

ACM Reference Format:
Toyomi Ishida and Hidetake Uwano. 2019. Time Series Analysis of Programmer's EEG for Debug State Classification. In *Companion of the 3rd International Conference on Art, Science, and Engineering of Programming (Programming '19), April 1–4, 2019, Genova, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328433.3328459

1 INTRODUCTION

Appropriate support for program debugging contributes to efficient software development. Classification of programmer's states during debugging enables us to real-time support based on their situations; e.g. visualization of unit test coverage for programmer who seeks problematic lines of code. Many studies proposed process model of debug and classification of programmer's activities during debugging. Araki et al. define the debug as a process that repeats hypothesis set, modification, selection and verification [1]. Xu and Rajlich [2] classify activities during debug into six processes (knowledge, comprehension, application, analysis, synthesis and evaluation) based on Bloom's taxonomy, one of the intellectual activity classification [3]. These studies organize a set of sub-processes in debug process from view point of logical architecture.

Our long term research goal is to empirically analyze the debug process using bio information measurement. Bio information is a metric which is measured from human body and have relation with thinking and psychological state. Debugging (and most activity in software development) is mainly performed mentally, so it is hard to understand their "state of understanding" from visual observation. In Software Engineering, many studies use bio information to analyze programmers' thinking and psychological state [4, 5].

Debugging requires understanding of operation in target source code (how it works) and requirement specification of the program (what to do) Understanding the operation of source code involves to comprehend process flow, variables role, method functionality, syntax of programming language, API architecture and other miscellaneous activities. Also to find bugs, programmers must confirm the source code implementation matches the specification. These understandings is accomplished by read each lines of the source code and the requirement specifications, then lead to understand of block, method, class and entire program. In this paper, we define a "micro process" as an activity to line-wise comprehension of source code and other software documents, such as variable (and method) declaration, calculation, conditional expression, method call, requirement explanation, etc. Programmers during a debug process perform a continuous set of micro processes to lines of source code and other document, then understand a larger architecture such as process flow and caller-callee relationship by combining the understandings from micro processes. Therefore, line-wise analysis allows us to fine-grained understandings of programmer's debug process; Such knowledge contributes a real-time assist of the programmer based on a state of understanding.

Micro processes during debug change frequently based on a literature in lines. These different micro processes activate different brain functionality (such as syntax recognition, calculation, memorization) in different intensities. In this paper, authors measure EEG (electroencephalogram, i.e. brain waves) during the debug process. Frequency bands of EEG, such as α wave and β wave reflect mental (emotional and intellectual) states and the brain activity [6]. We assume the brain activities in each micro process are reflected rapidly to the EEG, because EEG has a high time resolution. Therefore, time series changes of programmer's EEG may synchronize with their reading of lines.

Eye movements analysis is well used technique to analyze the comprehension process in programming [7–9]. Programmers read source code based on their comprehension strategies, hence the strategies are reflected in their eye movements. That is, eye movements reflect how programmers try to understand the source code through sequence of micro processes, while the EEG reflects the result of each micro process. In this paper, we analyze programmer's EEG during debug process to combine with eye movements recorded at same time. We evaluate whether the EEG reflect the result of debug or not from pilot experiment. In the experiment,

Programming '19, April 1–4, 2019, Genova, Italy
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6257-3/19/04.
https://doi.org/10.1145/3328433.3328459

debug process is divided into two steps, 1) understanding step and 2) bug judgement step. A power spectrum of two frequency bands (α wave and β wave) is compared between who succeeds each step and who fails. We examine three research questions from an experiment:

- RQ1: At completion of understanding step, is EEG of programmers who succeed understanding differ from EEG of programmers who fail the understanding?
- RQ2: At completion of bug judgement step, is EEG of programmers who succeed bug judgement differ from EEG of programmers who fail the bug judgement?
- RQ3: Is time series changes different between success and failure at understanding/bug-judgement step?

2 RELATED WORK

Some bio information is used for quantitative analysis of program comprehension processes in prior works. Siegmund et al. investigated the activation at each brain portion during program comprehension by using fMRI [5, 10]. The fMRI is a device that measures brain activity from blood flow changes in a human head. In the experiment, participants read short source code snippets for two types of error detection, 1) syntax error and 2) logic error. As the results show participants who understand the source code had activated brain regions that related to problem solving, memorization, and sentence comprehension.

The fMRI has a high spatial resolution compares with other brain measurement devices such as EEG, and therefore is suitable for understanding which brain region is activated in program comprehension. On the other hand, the fMRI is not suitable for measurement in practical environment because the device requires participants to lie on the device.

The combination of EEG and eye movements is well used in research of program comprehension. Fritz et al. predicts the programmer's states and task characteristics using machine learning with combination of several bio information [11]. Their method predicts the subjective difficulties (high and low) from EEG (α, β, and other Frequency bands), eye movements (pupil size, saccades, and fixations), and EDA (skin conductance). Zuger et al. predicts the programmer's condition which good for interruption [8]. Muller et al. builds a psychological model for estimate the programmer's emotion (concentration and happiness) [9].

These researches show that EEG and eye movements are useful metrics for analysis and prediction for programmer's states. In this paper, authors analyze the difference of brainwave features during debugging between who succeed to find bug and who fail. We focus on high resolution of EEG via time series analysis; changes α wave and β wave in comprehension/debugging task is analyzed.

3 EEG AND EYE MOVEMENTS

3.1 EEG

An EEG is an electrical activity that arises from brain and recorded through electrodes placed on the scalp. Brain electric potential is measured as difference of two electrodes [12]. Each electrode is placed on a point specified by the International 10-20 system shown in Figure 1 [13]. The 10-20 system designates 19 electrode placements except the electrodes defining ground potential.

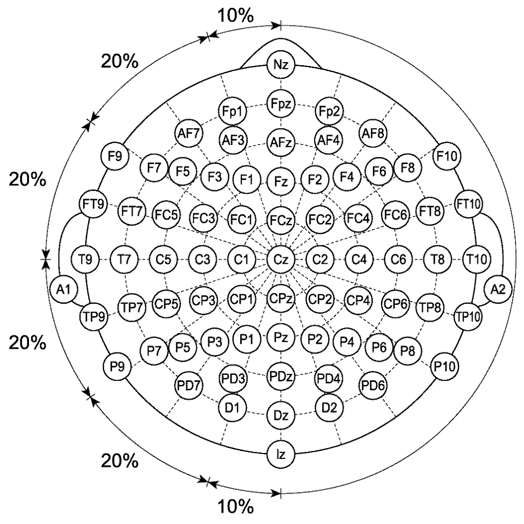


Figure 1: International 10-20 system of electrode placement

The purpose of measurement determines place and number of electrodes. There are two electroencephalographic derivation methods; standard and bipolar method. The standard method is used when two electrodes are placed near each other. The bipolar method is used to measure the difference of two specific positions and remove the irrelevant background components. EEG waveforms are generally classified according to their frequency [12]. The frequency is calculated by FFT (Fast Fourier Transform), or STFT (Short Time Fourier Transform). The familiar classification of waveforms is following bandwidths.

- δ wave: 0.1 - 4 Hz
- θ wave: 4 - 8 Hz
- α wave: 8 - 13 Hz
- β wave: 13 - 30 Hz
- γ wave: 30 - 100 Hz

EEG frequency differs depending on different behavior and mental conditions of the brain [6]. For example, α wave appears strongly when a person is relaxed or concentrating. When the person is thinking or stressed, β wave becomes stronger and α wave is weakened relatively. The EEG is also used to evaluate mental (not emotional but intellectual) activities [8, 11]. In this paper, we use the EEG as metric to classify programmers who understand a program behavior and/or succeed to judge the program contains a bug.

3.2 Eye Movement

Eye movement is recorded as a series of the coordination (x and y) on display that calculated from the eyeball movement [14]. Analyzing the eye movements shows us where the participant is looking at and how long each gaze takes place. Eye movement is used to analyze the difference between novices and experts, especially in cognitive engineering. The eye movement analysis is also used to program comprehension and debugging research. Behroozi et al. distinguish who understands source code via eye movements analysis [15]. Busjahn et al. found differences in eye movement while reading source code between experts and novices [16]. Software documents such as requirement specifications or source code consists of lines. Developers read each line to comprehend entire document and correspondence between documents. The analysis of eye movements in software engineering research enabled us to understand the line-wise analysis of software reading/comprehension process.

On the other hand, eye movement analysis does not allowed to know the states of understandings. In general, the longer gaze time to one line means that the line is important for entire understanding or difficult to understand meaning of the line. Eye movement analysis can not distinguish them because the eye movement hardly contains "result" of the reading; the programmer understood mean of the line or not. In this research, we combine the eye movement and time series EEG analysis for fine-grained understanding of program comprehension. We believe the combination tells us which part of source code is read by programmer and level of the programmer's understanding to the part, e.g. "She understands variable x stores result of calculation, but not sure how the calculation is work."

4 EXPERIMENT

Five undergraduate students in information engineering department of our college (All male, aged from 19 to 20) participated in this experiment. All student finished basic course of Java programming at least. Each participant read Java source code as the debugging tasks. During the task, we record EEG to classify whether participants succeeded or failed tasks.

4.1 Environment

This experiment is performed in a quiet room with only one participants and two experimenters. In order to suppress artifacts due to body movements, the participants sit on a chair with armrests and footrests, and are instructed to reduce their body movement as much as possible.

We use the NeXus-10 MARK II manufactured by Nanotech Image Ltd. as our EEG measurement device. The device measures EEG with 256Hz sampling frequency, and a measured data is transferred to a PC via Bluetooth. Electrodes are located using the standard electrode derivation method. The ground electrode is located at right ear (A2), standard electrode is located at left ear (A1) and measurement electrode is located at back of the head (Pz), which same allocation of our previous study that distinguish whether programmer found an implementation strategy from requirement specification [17]. Figure 2 shows appearance of the subject during this experiment. We use one PC for device control, data recording, and another one is for task presentation. The display size and resolution for the task presentation are 21.3-inch, 1920 x 1080.

4.2 Task

4.2.1 Two Steps in Debug Process. We consider a debug process as being composed of two steps, 1) understanding step and 2) bug judgement step. In debug process, programmer understands the objectives of program from the specifications, then comprehend the program behavior from source code. After both understandings,

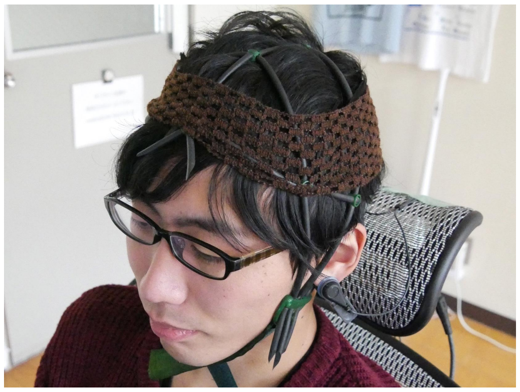


Figure 2: Appearance during EEG measurement

programmers judge the source code implements the specification correctly or not. These two different steps may require different brain functions, therefore brain activity during the steps have the differences.

In this experiment, we design the task to measure EEG in each step at single debugging process. In each task, participants perform understanding step first, then continues to judgement step. In both steps, specification written in Japanese and Java source code are showed to the participants.

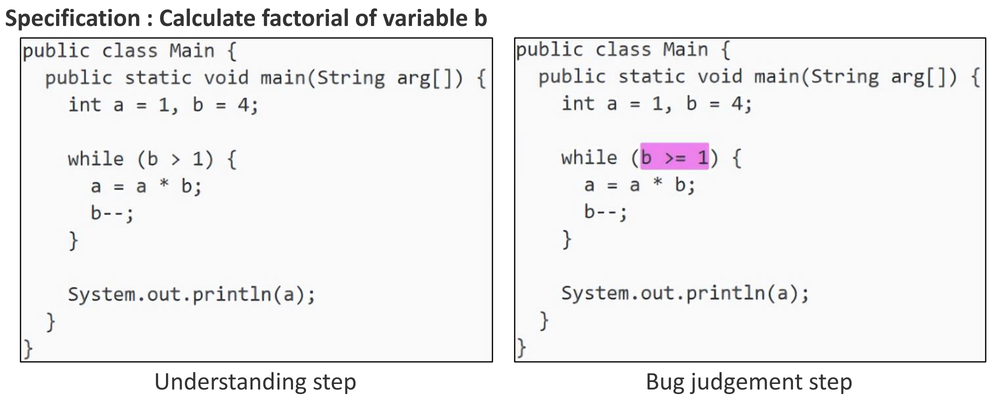
4.2.2 Understanding Step. In the understanding step, participants read the specification and the source code to comprehend program behavior. The participants answer a question that confirms s/he comprehends the program behavior correctly; i.e. question: "What is the value of variable x at the sixth line in the second loop?" and answer: "x is 7." The question is displayed with the specification and the source code at display, the participant answer the question verbally. The definition of *success* and $failure$ in understanding step is as follow:

- *success*: A participant answers question within a time limit (2 minutes 30 seconds) and the answer match with prepared one.
- $failure$: An answer of a participant is not match with prepared one, or no answer within the time limit.

In this step, participants perform micro processes to comprehend program behavior such as syntax comprehension, calculation, and variable memorization. Source code used in the step does not contain any fault, therefore comparison of specification and source code or fault judge is not performed.

4.2.3 Bug Judgement Step. In the bug judgement step, participants read source code that modified a part of the source code showed in understanding step. The participants are instructed to judge whether the modified source code fulfills the specification. To avoid participants spending time to find modified lines, we colored the line. Figure 3 shows example of source code before and after the modification. The definition of *success* and $failure$ in bug judgement step is as follow:

- *success*: A participant answers question within a time limit (1 minute) and the answer match with prepared one.



```
public class Main {
  public static void main(String arg[]) {
    int a = 1, b = 4;

    while (b > 1) {
      a = a * b;
      b--;
    }

    System.out.println(a);
  }
}
```

```
public class Main {
  public static void main(String arg[]) {
    int a = 1, b = 4;

    while (b >= 1) {
      a = a * b;
      b--;
    }

    System.out.println(a);
  }
}
```

Figure 3: Modification of source code in bug judging step

- *failure*: An answer of a participant is not match with prepared one, or no answer within the time limit.

In this step, participants perform micro processes to decide the modified source code fulfills the specification. Participants already read a specification and source code before the modification at previous step, hence program comprehension seldom occurs.

4.2.4 Material. Table 1 shows the list of tasks. Each task contains a Java source code and a short sentence that explain the specification of the program in Japanese. In this experiment, we classify the EEG in each step into two groups, success and failure, hence it is preferred the number of tasks in each group is similar. To achieve this, we prepare eight difficult tasks and eight easy tasks.

The source code of the easy task consists of main method with single loop block and/or a single conditional branch; easy to finish both steps in the task within time limit. The source code of the difficult task uses a complex algorithm that has multiple methods, recursive structure; hard to finish both steps in the task within time limit. The order of each task is counterbalanced to minimize the effect of task order.

4.3 Analysis

EEG in each step is extracted to calculate power spectrum of α and β waves. Each EEG is divided to 0.2 seconds length, then calculated power spectrum by STFT. The extracted powers include a large individual difference. Therefore, each power spectrum is normalized by the median value of each step. The normalized value means the difference of powers at each time zone during a step.

We analyze the difference between two groups (*success* and *failure*) in each step. First, we focus on the start/end time of each step. The difference between start and end reflects the result (success or failure) of each step; brain activities should increase when the step is success. We compare the average of normalized power spectrum at first five seconds in each step (*start*) and the last five seconds in each step (*end*). Second, we compare the time series of normalized α and β power spectra in each step; brain activity should increase over time in each step of success group.

Table 1: Tasks used in the experiment

	difficulty	specification
1	easy	Calculate factorial
2		Search maximum number
3		Judge prime number
4		Search median number
5		Calculate power of number
6		Swap two numbers
7		Judge string match
8		Output reversed string
9	difficult	Tower of Hanoi
10		Count route combination
11		List-up permutation
12		List-up combinations by recursion
13		Count the combination of coins
14		List-up string combination
15		Predict clouds movement
16		Calculate G.C.D. and L.C.M.

Table 2: ***Success*** **and** ***failure*** **at each step**

Participant ID	Understanding		Bug judgement	
	success	*failure*	*success*	*failure*
1	10	6	10	6
2	6	10	11	5
3	9	7	12	4
4	11	5	12	4
5	11	5	10	6

5 RESULTS AND DISCUSSION

We obtained 80 data (16 tasks × 5 participants) as result of experiment, and every data is used in the analysis. Large body movements and immediate answer (without thinking) is not observed during the experiment. Table 2 shows the number of *success* and *failure* at each step.

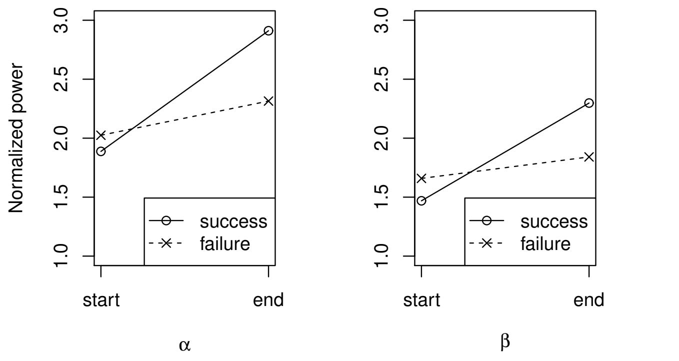


Figure 4: Power spectra in understanding step

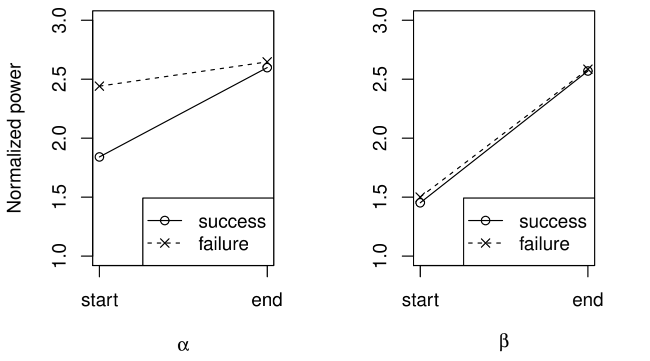


Figure 5: Power spectra in bug judgement step

5.1 EEG between Start and End of Understanding Step

5.1.1 Data. Figure 4 shows the average of normalized α and β power spectra at understanding step. The vertical axis shows the power spectrum, and the horizontal axis shows the start/end time zone. The figure shows both α and β of *success* group increases greatly compared with $failure$ group. The result of Welch's t-test between $start$ and end described α ($p = 0.002$) and β ($p < 0.001$) of *success* group have a significant difference respectively. In $failure$ group, α ($p = 0.385$) and β ($p = 0.420$) have no significant differences.

5.1.2 Discussion. The result shows that success of source code understanding increases participant's brain activity. The result suggests that participants who comprehend source code correctly employ their brain function sufficiently. On the other hand, participants who cannot comprehend source code stuck during the comprehension, therefore the brain function is not fully used. This characteristic of α and β can use to classify whether programmers understood source code or not. However, is it also possible that participant relaxed as their understanding progressed, then the α wave is increased in both group (*success* and $failure$.) We need a further study to interpretation of the result.

As a result of the analysis, we can answer RQ1 as follows: At completion of understanding step, α and β wave of programmers who succeed understanding become higher compared with programmers who fail the understanding.

5.2 EEG between Start and End of Bug Judgement Step

5.2.1 Data. Figure 5 shows the average of normalized α and β power spectra at bug judgement step. The figure shows similar tendency with understanding step, the α of the *success* group increases greatly compared with the $failure$ group. On the other hand, the β power spectrum of *success* and $failure$ shows same tendency from $start$ to end; both *success* and $failure$ increase greatly. The result of Welch's t-test between $start$ and end described α ($p = 0.005$) and β ($p < 0.001$) of *success* group have a significant difference respectively. In $failure$ group, β ($p = 0.003$) has a significant difference. There is no significant difference at α ($p = 0.617$).

5.2.2 Discussion. The result shows that success of bug judgement increases participant's α wave, however, β in both group increase. The difference between the two steps may indicate the difference of brain activities in these steps. Participants at understanding step read each line of source code and combine to one meaningful chunk of process. In this step, participants read a specification as guideline of source code comprehension. Participants at bug judgement step already understand the specification and most part of the source code, they compare the specification and the modified source code to find a gap between them. We assume that these differences of activities between the two steps lead a different set of micro processes, then appear in β power spectrum. Detailed analysis of EEG at micro processes is an important future work.

As a result of the analysis, we can answer RQ2 as follows: At completion of bug judgement step, α wave of programmers who succeed bug judgement become higher compared with programmers who fail the bug judgement.

5.3 EEG Changes during Steps

In this analysis, we focus on the α wave which has significant difference between start and end in both step. Figure 6 and figure 7 show Power spectrum of α wave during each step. The vertical axis shows the power spectrum, and the horizontal axis shows the time range (five second increments) from the start of step. Here, the number of data in each time range is different, because participants finish their step any time when they understand/judge the source code; the later the fewer. Each line in these figures is drawn while the number of data is more than five.

5.3.1 Data. Figure 6 shows the power spectrum of *success* increased rapidly in 16-20 seconds or later at understanding step. We conduct the Welch's t-test between first time range (1-5 seconds) and each after to clarify when the power spectrum increase. As a result, significant difference ($p < 0.05$) at 41-45 seconds and

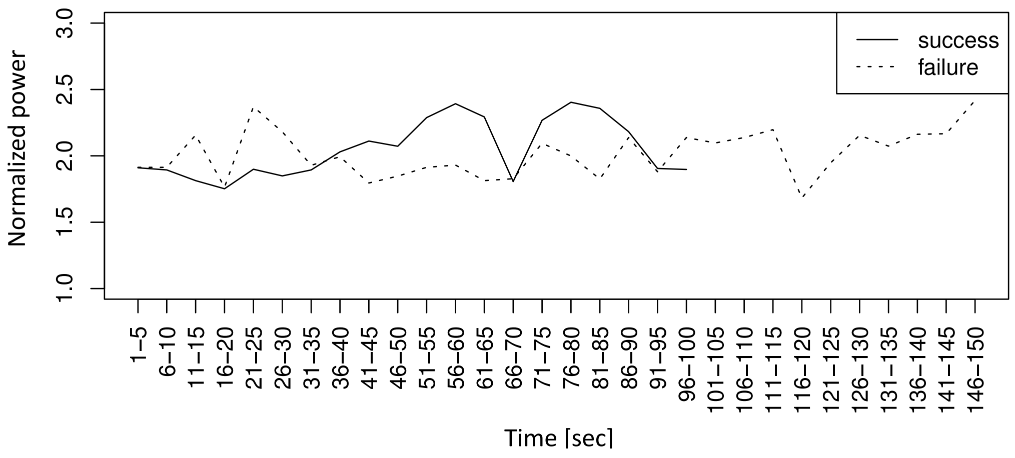


Figure 6: Power spectrum of α wave during the understanding step

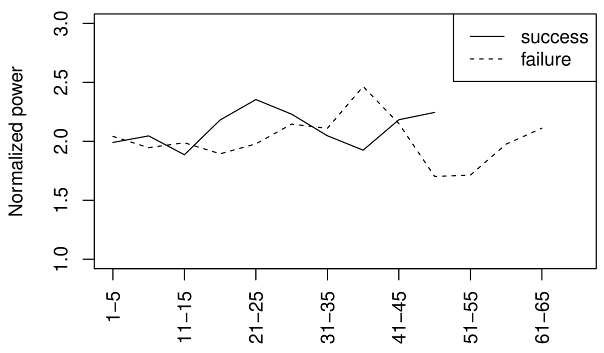


Figure 7: Power spectrum of α wave during the bug judge step

all of after the time range was observed in *success*. Compared with *success*, the power spectrum of *failure* slowly increased from 41-45 seconds or later. The result of the Welch's t-test shows significant difference ($p < 0.05$) at 106-110 seconds and all of after the time range.

Figure 7 shows the power spectrum of the *success* group increased in 11-15 seconds or later at bug judgement step. The result of the Welch's t-test shows a significant difference ($p < 0.05$) at 11-15 seconds and all of after the time range in *success* group. On the other hand, the value of *failure* become the lower than the first time range, after the largest value at 36-40 seconds. There was significant difference ($p < 0.05$) at 41-45 seconds and all of after the time range.

5.3.2 Discussion. These results indicate that the brain activity increased in the early stage in both steps when succeed. That is, EEG measurement during the source code comprehension and bug judgement is a promising metric for classification of programmers who need support in their task. In this analysis, we averaged the power spectrum of each step in different participant and task, hence different behavior (i.e. just before they finish understanding or in the midst of) may be mixed in same time range. In future work, the behavior during the same time range is required to standardize via some criteria such as the place of eye movement fixation.

As a result of the analysis, we can answer RQ3 as follows: Time series changes different between success and failure at understanding and bug-judgement step. In both step, programmers who success their understanding/bug-judgement have an early increase of α wave compared with programmers who fail the step.

6 CONCLUSION AND FUTURE WORK

In this paper, we analyzed time series changes of EEG during debugging to classify the success and failure in program comprehension and bug judgement. The result of the experiment showed the participants who success source code understanding significantly increased power spectrum of α and β waves over time. Also, the participants who succeed in bug judgement step significantly increased power spectrum of α wave. These results support three RQs and suggest time series analysis of EEG is useful for classification of programmers who need support during their program comprehension and/or debugging.

These results are useful for trainer/teacher to educate programmers efficiently. For example, the EEG measurement during training or class allows us to classify programmers who need supports of trainer in real-time. The proper timing support for proper programmer improves the learning efficiency.

To implement an environment that measure the EEG and eye movements during training/education requires devices inexpensive and easy to measure. Recently, small and inexpensive devices appear and utilized in entertainment field such as gaming. These easy-to-use devices will realize the measurement environment in educational field.

As future work, detailed analysis of EEG power spectrum change during each step is interesting. EEG during each step is affected by performed micro processes. The change of micro process can be observed by eye movement analysis, therefore combined analysis of EEG and eye movement will lead us to understanding of "what result occurred when the programmer read what." Employment of machine learning for accurate classification of program understanding and/or bug judgement is also an interesting future work.

ACKNOWLEDGMENTS

Part of this work was supported by JSPS KAKENHI Grant Number JP16K00114.

REFERENCES

[1] Araki, K., Furukawa, Z., and Cheng, J., "A general framework for debugging," IEEE Software, pp. 14-20 (1991).
[2] Xu, S., and Václav, R., "Cognitive process during program debugging," In Proceedings of the Third IEEE International Conference on Cognitive Informatics, pp. 176-182 (2004).
[3] Bloom, B. S., Engelhart, M. B., Furst, E. J., Hill, W. H. and Krathwohl, D. R., "Taxonomy of educational objectives. The classification of educational goals. Handbook 1: Cognitive domain," New York: Longmans Green (1956).
[4] Torii, K., Matsumoto, K., Nakakoji, K., Takada, Y., Takada, S., and Shima, K., "Ginger2: an environment for computer-aided empirical software engineering," in IEEE Transactions on Software Engineering, Vol. 25, No. 4, pp. 474-492 (1999).
[5] Siegmund, J., Peitek, N., Parnin, C., Apel, S., Hofmeister, J., Kastner, C., Begel, A., Bethmann, A., and Brechmann, A., "Measuring neural efficiency of program comprehension," In Proceedings of the 11th Joint Meeting on Foundations of Software Engineering(ESEC/FSE), pp. 140-150 (2017).
[6] Birbaumer, N., Elbert, T., G M Canavan, A., Rockstroh, B., "Slow potentials of the cerebral cortex and behavior," Physiological Reviews, Vol.70, pp. 1-41 (1990).
[7] Rodeghero, P., Liu, C., McBurney P. W., and McMillan, C., "An eye-tracking study of java programmers and application to source code summarization," in IEEE Transactions on Software Engineering, Vol.41, No.11, pp. 1038-1054 (2015).
[8] Zuger, M., and Fritz, T., "Interruptibility of software developers and its prediction using psycho-physiological sensors," In Proceedings of 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 2981-2990 (2015).
[9] Muller, S. C., and Fritz.T., "Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress," In Proceedings of 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol.1, pp. 688-699 (2015).
[10] Siegmund, J., A. Brechmann, Apel, S., Kastner, C., Liebig, J., Leich, T., and Saake, G., "Toward measuring program comprehension with functional magnetic resonance imaging," In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE'12), No.24 (2012).
[11] Fritz, T., Begel, A., Müller, S. C., Yigit-Elliott, S., Züger, M., "Using psycho-physiological measures to assess task difficulty in software development," In Proceedings of the International Conference on Software Engineering (ICSE), pp. 402-413 (2014).
[12] Satheesh, Kumar, J., Bhuvaneswari, P., "Analysis of electroencephalography (EEG) signals and its categorization - A study," In Proceedings of the International Conference on Modeling, Optimization and Computing (ICMOC), Vol.38, pp. 2525-2536 (2012).
[13] Klem, GH, Luders, H, Jasper, HH, Elger, C., "The ten-twenty electrode system of the international federation," The International Federation of Clinical Neurophysiology. Electroencephalography and clinical neurophysiology, Vol.52, pp. 3-6 (1999).
[14] Duchowski, A. T., "Eye tracking methodology," Springer (2006).
[15] Behroozi, M., Lui, A., Moore, I., Ford, D., and Parnin, C., "Dazed: measuring the cognitive load of solving technical interview problems at the whiteboard ," In Proceedings of the International Conference on Software Engineering (ICSE), pp. 93-96 (2018).
[16] Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J. H., Schulte, C., Sharif, B., Tamm, S., "Eye movements in code reading: Relaxing the linear order," In Proceedings of the 23rd International Conference on Program Comprehension (ICPC), pp. 255-265 (2015).
[17] Yamamoto, A., Uwano, H., and Ikutani, Y., "Programmer's electroencephalogram who found implementation strategy," In Proceedings of 4th International Conference on Applied Computing & Information Technology (ACIT), pp. 164-168 (2016).