Yang, J. C. et al. (Eds.) (2018). Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education

Identifying Common Code Reading Patterns using Scanpath Trend Analysis with a Tolerance

Christine Lourrine TABLATIN[a,b*]& Ma. Mercedes RODRIGO[a]
[a]*Ateneo de Manila University, Philippines*
[b]*Pangasinan State University, Urdaneta City, Philippines*
*tablatinchristine@gmail.com

Abstract: Eye tracking data, particularly scanpath, provides valuable insights about code reading patterns which could describe actual comprehension strategies used. However, aggregating multiple scanpaths into one representative path is challenging since individual scanpaths tend to be different and are highly individualistic. The differences may affect the identification of a representative path which could decrease its similarity to individual scanpaths. Thus, we aim to identify a trending scanpath using Scanpath Trend Analysis (STA) with a tolerance to reveal common code reading patterns of high and low performing students while finding bugs in a static source code. Results show that variance exists in the scanpaths of high performing students which suggests that they follow varied code reading patterns while low performing students follow similar code reading patterns. Further, high performing students read code in a logical manner and a somewhat linear code reading pattern along with chunking of program code was employed which makes it possible to perceive the program better and hence, error regions are fixated. In contrast, low performing students jump directly to certain statements without following program's logic. This study addresses the challenge of identifying common code reading patterns that could help us determine effective strategies to be explicitly taught to students and develop learning materials to help improve their code reading and code comprehension skills.

Keywords: Eye tracking, code reading, code reading patterns, scanpath analysis, Scanpath Trend Analysis, tolerance level parameter.

1. Introduction

One of the interesting approaches to study how programmers perform comprehension tasks, is the use of eye tracker. Eye tracking technology is used to record eye movements while reading source code which shows the focus of attention and how focus travels within the stimuli (Melo, et al., 2017). Analyzing eye tracking data, particularly scanpath which is defined as a sequence of consecutive fixations and saccades (Goldberg and Helfman, 2010), provides valuable insights about code reading patterns which could help describe actual comprehension strategies used (Bednarik, et al,. 2016).

Studies using eye tracking technology to uncover the cognitive processes of programmers while performing code reading and code comprehension tasks is becoming widespread (Busjhan, et al., 2015; Jbara and Feitelson, 2017; Lin, et al, 2016). Understanding how programmers read and comprehend source code could provide information on how programmers think while performing the task as well as insights on how we can improve the overall learning process by improving learning materials based on eye tracking data (Bednarik, et al., 2014). Further, exploring the strategies used by experts while performing comprehension tasks particularly code reading, would allow us to uncover effective strategies that could be explicitly taught to improve code reading and code comprehension skills of students (Bednarik, et al., 2016). However, finding common patterns in the visual strategies used by experts is challenging, since their individual scanpaths tend to be different from each other (Eraslan, et al., 2017) and are highly individualistic (Jbara and Feitleson, 2017). The differences may affect the identification of the common scanpath of a group which could decrease its similarity to individual scanpaths. To identify an appropriate technique for a sequential

analysis of eye tracking scanpaths to generate a common scanpath that closely resembles the individual scanpath to describe common code reading patterns, we review existing common scanpath identification techniques. Dotplots algorithm (Goldberg and Helfman, 2010) and eMINE algorithm (Eraslan, et al., 2016) use hierarchical clustering to generate a common scanpath. However, Dotplots uses an aggregate strategy that is entirely dependent on the sequential matches found between the pair of scanpaths which generates uncertain result if there are only few matches found in the pair of scanpaths. eMINE on the other hand, is likely to produce short common scanpath which is not useful for describing reading patterns. Scanpath Trend Analysis (STA) algorithm (Eraslan, et al., 2016) uses a multi-pass approach consists of three main stages to generate a common scanpath. The generated common scanpath will be based from the minimum total number of fixations and minimum total durations, and their positions in the individual scanpaths. STA generated common scanpath which is more similar to individual scanpaths compared to Dotplots and eMINE. However, STA did not consider the variance between individual scanpaths in generating common scanpath which could negatively affect the identification of a common scanpath. To address this problem, a parameter called tolerance level was added to STA (Eraslan, et al., 2017). This parameter is used in the second stage of the algorithm which allows researchers to adjust the tolerance level in the stage of identifying trending AOIs. Therefore, STA with a tolerance can be used to identify a common scanpath which tolerates variances between scanpaths of a group. Thus, we aim to identify a common scanpath using STA with a tolerance to reveal common code reading patterns of high and low performing students while finding bugs in a static source code.

Previous studies were conducted to determine programmer's pattern of eye movements using sequential analysis of scanpaths to understand their cognitive processes. We review these studies to gain information about the code reading patterns found using scanpath analysis, the task and programming language used to investigate reading patterns, and method used to determine reading patterns of experts and novices. The differences between the reading patterns of novices and experts were examined by Busjahn, et al. (2015) and (Lin, et al., 2016) which is similar to our study since we aim to identify common code reading patterns of high and low performing students, while the studies of Uwano, et al. (2006) and Jbara and Feitelson (2017) describes the scanpaths of the individual subjects which is in contrast with our study. Our study is different in terms of the task used in (Busjahn, et al., 2015; Jbara and Feitleson, 2017) and is more similar in (Uwano, et al., 2006; Lin, et al., 2016). Our study investigates the code reading patterns of students while finding bugs in a static source code written in Java. Furthermore, the method used in our study is different from the methods used by previous studies. As of this writing, no studies have been conducted to describe code reading patterns while finding bugs using STA algorithm with a tolerance. This algorithm was applied to eye tracking data while subjects are browsing and searching information on web pages to find their trending scanpath (Eraslan, et al., 2017) which is different in the context of this study.

2. Methods

This paper is an analysis of a larger eye tracking study on programmer tracing and debugging skills as well as development of higher education's capacity to conduct eye tracking research. The methods discussed here are also discussed in the study of (Villamor and Rodrigo, 2017). Eye movements data were collected from 44 students from three private schools in the Philippines. We identified two types of participant groupings based on their pre-test scores. The first group consists of 25 students with High Scores while the other group consists of 19 students with Low scores. The groupings were used to identify trending scanpath of high and low performing groups.

There were 12 short Java programs with defects used in the experiment. Out of the 12 short programs, 9 of them had 3 defects while 3 had 1 defect each. For this study, we selected one program with 3 defects out of the 12 programs. The selected program code used in this analysis is a program code that display all prime numbers from a set of input. The program contains conditional statements and repetition structures with 33 lines of codes. OGAMA tool was used to identify the coordinates of the Areas of Interests (AOIs) which were defined per line of codes to determine which lines are frequently fixated and to identify the vertical code reading patterns employed by the participants.

To identify the common scanpath of each group, we used Scanpath Trend Analysis (STA) with a tolerance (Eraslan, 2017). The STA algorithm with a tolerance consists of three main stages. The first stage is the preparation of the individual scanpaths. Each fixation points in the individual scanpath is matched to the identified AOIs of the program code. The individual scanpaths in this stage are represented as a series of AOIs with fixation durations. The second stage identifies the trending AOIs in the scanpaths. The trending AOIs were identified using their number of fixations and their fixation durations. The final stage in the algorithm is the identification of the common scanpath using the trending AOIs based on their positions in the individual scanpaths. The scanpaths are abstracted by combining the same AOI instances and then computes the priority value of each AOI instance in each individual scanpath. When all the priority values are, the total priority value for each AOI instance is calculated. The algorithm then positions the instances into the common scanpath based on their total priority values from highest to lowest. Once the trending visual element instances are positioned in the common scanpath, their numbers are eliminated (e.g. E1 → E) and consecutive repetitions are removed (e.g. JEJJ → JEJ). Thus, the common scanpath is represented in terms of the AOIs. For a detailed discussion of the algorithm, see (Eraslan, et al., 2017).

To determine how similar the common scanpath to the individual scanpaths, we use String-edit algorithm (Duchowski, et al., 2010) to calculate the distance between the common scanpath and individual scanpaths of each group of students. The String-edit distance is then used to calculate the similarity between the two scanpaths by using the equation $S = 100 * (1 - (D/L))$ where D refers to the String-edit distance and L is the length of the longer scanpath. In this study, each group of high and low performing students from each schools will have three common scanpaths based on the three tolerance level parameters used. Statistical tests were used to determine which of the three parameters will be selected to generate a common scanpath for each group of students to reveal their common code reading patterns. To test the difference of the tolerance level parameters for each group based on the similarity scores, a repeated measures test was used. The same test was used to determine the difference between the similarity scores of the common and individual scanpaths.

3. Results and Discussion

The stages of STA algorithm with a tolerance was employed to generate three common scanpaths for each group of high and low performing students from all schools. To identify the tolerance level parameters for each group, we first consider all individual scanpaths and then adjust the parameter levels by decreasing it by 5. We repeatedly decrease the current parameter level by 5 if no changes in the identified trending AOI instances were observed. We identified three parameters for each group which ranges from 100% to 60% to check for differences on the resulting common scanpaths.

After identifying the three common scanpaths for each group with the chosen tolerance level parameters, the similarity scores were calculated as described in Section 2. To determine a common scanpath that is more similar to the individual scanpaths, the median similarity score of the common scanpath should be equal to or greater than the median similarity scores between the individual scanpaths. Statistical tests were conducted to help us decide which common scanpath will be selected to describe the common code reading patterns of the groups. Repeated measures ANOVA was used to determine the difference of the tolerance level parameters for each group of students based on the similarity scores. The same test was also used to determine the difference between the similarity scores of the common scanpath and the similarity scores of the individual scanpaths.

4.1. Common Scanpath of High Performing Students

The result of the repeated measures ANOVA with Greenhouse-Geisser correction determined that there was no significant difference between the tolerance levels for School A ($F(1.251, 25.028) = 2.525$, $p = .731$), there was a significant difference for School B ($F(1.888, 18.882) = 7.496$, $p = .005$), and there was a highly significant difference for School C ($F(1.995, 27.934) = 12.234$, $p = .000$). With regards to the test for difference between the similarity scores of the common scanpath to the individual scanpaths, the results reveal that there was a highly significant difference for School A ($F(1, 20) = 12.527$, $p = .002$), there was no significant difference for School B ($F(1, 10) = .543$, $p = .478$), and there was also no significant difference for School C ($F(1, 14) = .859$, $p = .370$).

The result of the tests for School A reveals that the similarity scores obtained across tolerance levels would be the same and that the overall mean of the common scanpath is greater than that of the individual scanpaths. Thus, we selected the common scanpath with 100% tolerance level to include all individual scanpaths and since greater similarity scores of the common scanpath is observed. The result of the tests of the three tolerance levels for School B and School C suggest that choosing one from the tolerance levels would have significant effect in the generation of the common scanpath. While no significant difference was observed from the similarity scores of the common and individual groups, we checked which of the tolerance levels have greater similarity scores in the common scanpath group. The common scanpath with a tolerance level of 80% have the highest similarity score among the three tolerance levels of School B. Therefore, we assumed that there is variance between the individual scanpaths. To verify, we checked the total fixation durations and the total number of fixations of the 6 students. We found that one student spent 149 seconds which is more than twice the time spent by the other 5 students (56, 72, 71, 43, 72). This observation is the same for the number of fixations. Therefore, we selected the common scanpath with 80% tolerance level for School B. For School C, we selected the trending scanpath with 100% tolerance level since it has the highest similarity score.

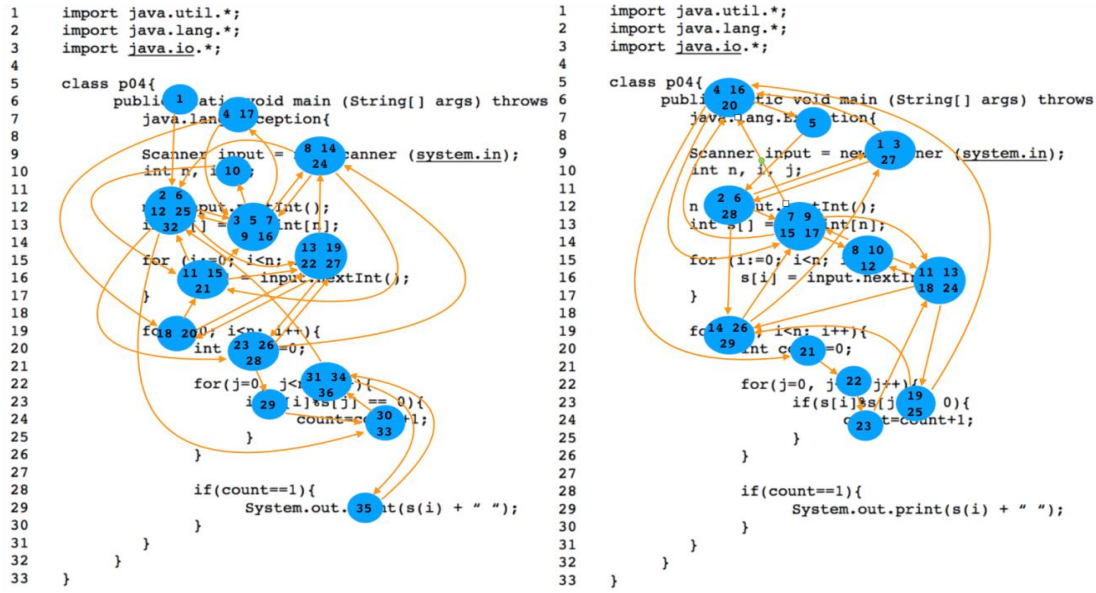
The selected common scanpaths were clustered using the priority values of the AOI instances of the individual scanpaths to generate the common scanpath. A graph was generated to show the sequence of eye movements of high performing students and is presented in Figure 1.a. The size of the blue circles represents the attention given for the AOI of the line number and numbers inside represents the common scanpath sequence of gazes for that line. The directional arrows represent the sequence origin and destination. The first 10 fixations of the common scanpath of high performing students concentrate on lines 6-13. Line 13 consists of declaration of array `s` of type `int` and instantiation of value from `n` which was assigned a value after accepting an input from the user located on line 12 and input was made possible by the statement located on line 9 and the variable `n` of type `int` was declared on line 10. We can say that these sequence of eye movements reflect logical thinking process. We also observe regression to line 13 after line 12 was looked at which can be associated to the retrace declaration pattern (Uwano, et al., 2006) or jumping back to verify details pattern (Jbara and Feitelson, 2017). Fixations 11-20 reflect a close examination of three new statements located on lines 15, 16 and 19. We note that at this point, two fixations occurred on line 15 where the first error was located. The next set of fixations (21-30) also reflect understanding of three new statements located in lines 20, 23 and 24 which are located in the body of the second and third for loop. Line 15 received two additional fixations which mean that students suspect that this line has an error. The last set of fixations (31-36) concentrates on lines 22, 24 and 29. Half of the six fixations are located on line 22 which means that this line is suspected to contain an error. The only fixation on line 29 was the second to the last fixation in the scanpath. To summarize, the generated trending scanpath exhibits code reading patterns of experts or high performing students. The study of Lin, et al., (2016) found that high performing students read code in a more logical manner compared to low performing students. This is reflected on the first set of fixations. A somewhat linear code reading pattern was observed and is used along with chunking or slicing of code. This pattern can be associated to the nature of the task which is finding bugs in the program code. We also note that the scanpath was able to scan the entire program code and passed through all the lines where errors are located.

4.2. Common Scanpath of Low Performing Students

The result of the tests for difference between the tolerance levels based on the similarity scores of the low performing students reveal that there was no significant difference between the tolerance levels for all groups: ($F(1.722, 13.775) = 3.275$ $p = .074$) for School A, ($F(1.926, 26.964) = 1.790$, $p = .187$) for School B, and ($F(1.228, 12.275) = .629$, $p = .475$) for School C. For the test for difference between the similarity scores of the common scanpath group to the individual group, result of repeated measures ANOVA with Greenhouse-Geisser correction determined that there was a significant difference for School A ($F(1, 8) = 11.150$, $p = .010$), there was also a significant difference for School B ($F(1, 14) = 11.243$, $p = .005$), while there was no significant difference for School C ($F(1, 10) = .4.863$, $p = .052$). These results suggest that the variance between the scanpaths

is negligible since the results are almost the same across all schools with a slight difference in the result of School C since the p-value is slightly greater than .05. For all groups, the similarity scores of the common scanpath group are higher than the individual group. Therefore, we selected the common scanpath generated with 100% tolerance level which includes all scanpaths of the group.

The selected common scanpaths were clustered using the priority values of the AOI instances of the individual scanpaths to generate the common scanpath. Figure 1.b. shows the eye movement patterns of low performing students. Their first ten fixations covers line 6 to 15. We can say that based from these sequence of fixations, linear pattern was observed along with regressions or look back movements to lines 13 and 15. Students then looked at line 16 then back to line 15 then line 16 again then line 19 then back to line 13. After that, they looked at line 16 then 13 then back to 16 then 23 then line 6. The last set of fixations is similar to the non-linear reading pattern observed from the previous ten fixations. The code reading pattern exhibited is in line with the finding of the study of Lin, et al. (2016) that low performing students often jump directly to certain statements to find bugs without following the program's logic. Furthermore, compared to reading natural language text, novices read source code in a less linear manner (Busjahn, et al., 2015) which is observed in the scanpath of low performing students.



(a) (b)

Figure 1. (a) Common Scanpath of High Performing Students and (b) Common Scanpath of Low Performing Students

4. Conclusion and Future Works

Individual scanpaths tend to be different from each other and are highly individualistic and may affect the identification of a representative path of a group. Thus, this study addresses the challenge of determining a common scanpath that is more similar to individual scanpaths using STA with tolerance to reveal common code reading patterns of high and low performing students while finding bugs in a static source code. This algorithm gives us the ability to find an appropriate tolerance level for achieving the highest similarity of the common scanpath to individual scanpaths. Results show that by adjusting the tolerance level parameter in the stage of identifying trending AOIs, we were able to generate a common scanpath that is more similar to the individual scanpaths which could be used to generalize the code reading patterns of high and low performing students while finding bugs in a static source code. We also found that variance exists in the scanpaths of high performing students which suggests that they follow varied code reading patterns while low performing students follow similar code reading patterns. This also confirmed that high performing students read code in a logical manner while low performing students jump directly to certain statements to find bugs. Gaining insights into the code reading patterns of students would allow us to identify effective strategies that could be explicitly taught to students and develop learning materials to improve their

code reading and code comprehension skills. However, the code reading patterns found in this study are limited to the program code used in the analysis. Future work will look at the reading patterns of the same group of students on different program codes with various code complexity and relating the patterns to the type of programs to be able to validate and generalize the result of this study.

Acknowledgements

We thank the Ateneo de Manila, Ateneo de Davao, University of Southeastern Philippines, University of San Carlos, Private Education Assistance Committee of the Fund for Assistance to Private Education for the grant entitled "Analysis of Novice Programmer Tracing and Debugging Skills using Eye Tracking Data" and Ateneo de Manila University's Loyola Schools Scholarly Work Faculty grant entitled "Building Higher Education's Capacity to Conduct Eye-tracking Research using the Analysis of Novice Programmer Tracing and Debugging Skills as a Proof of Concept". We also thank Bobby Roaring for helping us in conducting the statistical analysis of this study.

References

Bednarik, R., Busjahn, T., Schulte, C., and Tamm, S. (Eds.). (2016). Eye Movements in Programming: Models to Data. *Proceedings of the Third International Workshop*, University of Eastern Finland, Joensuu Finland, 25-26.

Bednarik, R., Busjahn, T., and Schulte, C., (Eds.). (2014). Eye Movements in Programming Education: Analyzing the Exper's Gaze. *Proceedings of the First International Workshop*, University of Eastern Finland, Joensuu Finland, 13-15. Retrieved from http://epublications.uef.fi/pub/urn_isbn_978-952-61-1539-9/urn_isbn_978-952-61-1539-9.pdf

Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J.H., Schulte, C., Sharif, B., and Tamm, S. (2015). Eye movements in code reading: relaxing the linear order. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension* (ICPC '15). IEEE Press, Piscataway, NJ, USA, 255-265. Retrieved from https://dl.acm.org/citation.cfm?id=2820320

Duchowski, A. T., Driver, J., Jolaoso, S., Tan, W., Ramey, B. N., and Robbins, A. 2010. Scanpath Comparison Revisited. In *Proceedings of the 2010 Symposium on Eye Tracking Research and Applications*. ACM, New York, NY, USA, 219-226. Retrieved from https://dl.acm.org/citation.cfm?id=1743719

Eraslan, S., Yesilada, Y., & Harper, S. (2017). Engineering web-based interactive systems: trend analysis in eye tracking scanpaths with a tolerance. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 3-8). Retrieved from https://dl.acm.org/citation.cfm?id=3102116

Eraslan, S., Yesilada, Y., and Harper, S. (2016). Scanpath Trend Analysis on Web Pages: Clustering Eye Tracking Scanpaths. *ACM Trans. Web 10, 4, Article 20 (November 2016)*, 35 pages. DOI: https://doi.org/10.1145/2970818.

Goldberg, J. H., and Helfman, J. I. (2010). Scanpath Clustering and Aggregation. In *Proceedings of the 2010 Symposium on Eye Tracking Research and Applications*, ACM, New York, USA, 18-27. Retrieved from https://www.researchgate.net/publication/220811044_Scanpath_clustering_and_aggregation

Jbara, A., and Feitelson, D. G. (2017). "How Programmers Read Regular Code: A Controlled Experiment Using Eye Tracking," *Empirical Software Engineering*, June 2017, Volume 22, Issue 3, pp. 1440-1477 DOI 10.1007/s10664-016-9477-x

Lin, Y. T., Wu, C. C., Hou, T. Y., Lin, Y. C., Yang, F. Y.,& Chang, C. H. (2016). Tracking students' cognitive processes during program debugging—An eye-movement approach. *IEEE transactions on education, 59(3)*, 175-186. Retrieved from https://ieeexplore.ieee.org/document/7312518/

Melo, J., Narcizo, F.B.,Hansen, D.W., Brabrand, C., and Wasowski, A. (2017). "Variability through the Eyes of the Programmer," *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, Buenos Aires, 2017, pp. 34-44. doi: 10.1109/ICPC.2017.34

Uwano, H., Nakamura,M., Monden, A., and Matsumoto, K. (2006). Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 symposium on Eye tracking research & applications (ETRA '06)*. ACM, New York, NY, USA, 133-140. DOI=http://dx.doi.org/10.1145/1117309.1117357

Villamor, M. and Rodrigo, M. M. (2017). Characterizing Collaboration in the Pair Program Tracing and Debugging Eye-Tracking Experiment: A Preliminary Analysis. In *Proceedings of the 10th International Conference on Educational Data Mining*. (pp. 174-179). Retrieved from http://educationaldatamining.org/EDM2017/proc_files/papers/paper_31.pdf