Shih, JL. et al. (Eds.) (2023). Proceedings of the 31st International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Visual Attention Patterns in Processing Compiler Error Messages

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

Abstract: The difference in the visual attention of subjects while performing a debugging task can be measured using fixation count and fixation duration metrics. Thus, this paper investigated the visual attention patterns of high and low performing students engaged in a defect-finding task on multiple programs using these metrics. We performed statistical tests on the proportional fixation durations and fixation counts on the error lines and the compiler error messages to determine the difference in the visual attention patterns between the groups. The results of the statistical analysis revealed a significant difference between the high and low performing students across all programs. This implies that high performing students were associated with significantly higher visual attention on the error lines of the programs than the low performing students. However, the analysis of the proportional fixation duration and fixation count on the compiler error messages revealed no significant difference between the groups. The results suggest that both groups showed similar visual attention to the compiler error messages. The findings of this study provide insights into the visual attention patterns of student programmers in processing compiler error messages. High and low performing students could be distinguished based on their visual attention patterns on the error lines but not on the compiler error messages. Further, high performing students prefer a more analytical processing approach and pay attention to relevant code elements in debugging to correctly identify the errors while low performing students choose a more holistic approach.

Keywords: visual attention, compiler error messages, debugging, eye-tracking, fixation metrics

1. Introduction

Debugging is a critical skill that remains challenging to acquire for student programmers. Students rely on compiler error messages to aid them in fixing the errors in their programs while debugging them. Compiler error messages are pedagogically significant to student programmers since they give feedback on what went wrong in their programs (Becker et al., 2019). However, according to Du Boulay & Matthew (1984), computer programming students are unable to relate compiler error messages to actual code errors. One of the possible reasons for this is that compilers are well-known for generating cryptic and uninformative error messages (Barik et al., 2017). This problem still exists and contributes to a high attrition rate of students in computer programming (Becker et al., 2018). Therefore, researchers invested efforts in developing systems that offer students more insightful error messages (Nienaltowski et al., 2008; Dy & Rodrigo, 2010; Denny et al., 2014; Prather et al., 2017). The work of Nienaltowski and colleagues (2008) found that providing additional information regarding an error did not necessarily result in greater debugging accuracy. Denny et al. (2014) corroborated their finding. In contrast, the study of Prather, et al. (2017) found that enhanced compiler error messages are more helpful than standard compiler error messages through an eye tracking study.

By observing people's eye movements, researchers can determine what attracts them and possibly understand how information is interpreted (Bol et al., 2017). As a result, researchers have used eye tracking to measure the difficulty student programmers encounter in understanding the error messages they read. Barik et al. (2017) found that interpreting error messages is as challenging as reading source code. Some of the reasons comes from the lack of familiarity with error messages (Marceau et al., 2011) and the need for programmers to move between the error message and source code to understand the full context of the problem. Other researchers have examined the reading patterns that students employ to parse through code to find an error and how these patterns differed depending on student ability. The source codes used in some studies were written in C (Chandrika & Amudha, 2017; Nivala et al., 2016; Sharif, Falcone & Maletic, 2012), C++ and Python (Turner et al., 2014), and Java programming languages (Villamor & Rodrigo, 2022; Tablatin & Rodrigo, 2022). Although research on understanding how programmers read and comprehend source code has been conducted, research on processing compiler error messages is still limited, especially using C++ as the source code stimuli. Through an eye-tracking study, we hope to add to what is known about how student programmers read and process compiler error messages. In this paper, we investigated the visual attention patterns of high and low performing students in processing compiler error messages.

2. Methodology

2.1 Participants

The participants of this study were Information Technology students who have at least taken a college-level introductory programming course using C++ as the programming language. A total of 32 students aged 18 years old and above were recruited from a state university in the Philippines. The study used two participant groups: high performing and low performing. The scores of the participants in the debugging tasks were used to assign them to a particular group. High performing group consisted of students who scored above and equal to the mean score while the low performing group consisted of students who scored lower than the mean score.

2.2 Experimental Setup and Procedure

All participants underwent a screening process to determine their eligibility to take part in the study. Students who passed the initial screening were given an informed consent form to fill out and sign. The students who signed the informed consent forms were given two types of pre-tests: a personality test and a self-efficacy test. After the pre-tests, the participants were required to undergo a calibration test before starting with the eye-tracking experiment that was designed to last for about 30 minutes.

The experiment was conducted in a laboratory setting. The experimental setup consisted of a laptop, a 17-inch monitor, a mouse, and a Gazepoint GP3 table-mounted eye tracker with a sampling rate of 60 Hz and an accuracy range of 0.5–1 degree. The hardware was set up by extending the laptop's display to the monitor connected to it. The eye tracker was placed in front of the monitor to allow the participant to view the program codes while the eye tracker records the participant's eye movements. The participant sat in front of the monitor, eye tracker, and mouse. After successful calibration of the eye tracker, a custom-built slide viewer loaded with C++ programs was activated. The slide viewer enabled the participant to navigate through the programs using Previous and Next buttons. It also enabled the participant to mark error locations with a red ellipse using the mouse. A Reset button can be used to clear the marked errors on a given slide and a Finish button saves the participant's answers and ends the session.

2.3 Comprehension Task and Measures

The main task of the participants was to find the errors of the programs based on the compiler error message. Each program had been injected with one syntax error and there is no need to correct them. Participant performance was measured based on the correctness of the answers. Eye tracking metrics were also collected to measure the visual attention patterns of the students while performing the debugging task.

2.4 Stimuli Characteristics

The code pool size was limited to 5 unique programs with different syntax errors. The programs shown to the participants were brief (15 to 35 lines) and represent C++ constructs discussed during the first 6 to 8 weeks of a first collegiate programming course. These include data types, variables, operations, conditionals, and loops. Further, the level of difficulty of the code was limited to the types of exercises given to students who are being introduced to the constructs mentioned herein. We have limited the error types to syntax errors that are usually encountered by students who are trying to complete these programming exercises. Only 1 syntax error was injected in each program and no semantic or logical errors were included. We have chosen 5 syntax errors from the syntax errors identified in the study of Denny, et al. (2014) to be injected in the programs. Note that, despite this limitation, some of the errors may be non-literal. That is, they may not accurately reflect the error or its location. The programs used in the experiment are novice-friendly, self-contained, and did not require any special domain knowledge. The language used was familiar to the students, brief enough to fit on a single screen, and did not use unfamiliar APIs. The stimuli used in the experiment were static. Participants only have to point to the location of the error in each program. They do not need to correct it.

2.5 Data Pre-processing

The gaze movements of the participants were stored in a CSV file format. The time of the recording (timestamp) when fixations occur, the location of the fixations (values of x and y coordinates), and the fixation duration of each fixation were extracted from the CSV file for the visual attention patterns analysis. Areas of Interests (AOIs) of the 5 programs were drawn using the OGAMA Areas of Interest module (Vosskühler, 2009) to get the AOI coordinates. The AOIs marked in this study are the line where the error is located and the compiler error message of each program. Figure 3 shows the AOIs of one of the programs used in this study. The AOI coordinates extracted from OGAMA returns coordinates with respect to the setting of the screen resolution specified when the AOIs were defined. To map the location of fixations to the program codes, the x and y coordinates from the eye-tracking data were converted by multiplying the x coordinates with 1366 and the y coordinates with 768. This was done to match the coordinates of the program codes during the experiment since the screen resolution used was 1366 x 768. In addition, the fixation durations were recorded in terms of seconds by the eye tracker and were converted into milliseconds by multiplying the duration with 1000. These processes were done for the eye-tracking data of the 32 students to determine their visual attention patterns.

2.6 Data Analysis

To determine the difference in the visual attention patterns of high performing and low performing students, fixation count and fixation duration were used. The proportional fixation count and proportional fixation duration on the error lines and the proportional fixation count and proportional fixation duration on the compiler error messages were computed to measure the visual effort exerted by high performing and low performing students.

Statistical analysis was conducted to compare the visual efforts between the high and low performing students using the visual effort metrics stated above. Independent samples t-test was used to determine whether there is statistical evidence that the visual efforts are significantly different between high and low performing students.

3. Results and Discussion

The eye tracking data of 32 students from a state university in the Philippines were used in the analysis. Eighteen (18) students were identified as high performing while fourteen (14) students were considered low performing based on their debugging scores. The proportional fixation duration and count on the identified AOIs were computed as a ratio of fixation duration and fixation count on an AOI to the overall fixation durations and counts on each program, respectively. These data were used to determine the difference in the visual attention patterns of students while finding errors in the program.

3.1 Visual Attention Patterns on the Error Lines

To determine the difference in the visual attention patterns of the groups in the error lines, independent samples t-tests were used. An independent samples t-test was used to determine if there is a significant difference in the average proportional fixation duration of the high and low performing students. The result of the analysis revealed that there is a significant difference in the average proportional fixation duration of high performing students (*M* = 0.014, *SD* = 0.007) and low performing students (*M* = 0.007, *SD* = 0.002), *t*(17.823) = -3.738, *p* = .002. The result suggests that the average proportional fixation durations of the high performing students are significantly higher than the low performing students across all programs. Further, an independent samples t-test was also performed to determine if there is a significant difference in the visual attention of high and low performing students in terms of the average proportional fixation count on the error lines. The result of the analysis revealed that there is a significant difference in the average proportional fixation count of the high performing students (*M* = 0.013, *SD* = 0.006) and low performing students (*M* = 0.007, *SD* = 0.002), *t*(16.411) = -4.165, *p* = .001. The result suggests that the average proportional fixation count of the high performing students is significantly higher than the low performing students across all programs.

The results of the analysis of the visual attention patterns suggest that high performing students were associated with significantly higher proportional fixation durations and fixation counts on the error lines of the programs than the low performing students. Fixation durations are influenced by the complexity and difficulty of the visual content, task being performed (Bylinski, et al., 2015; Nivala, et al., 2016), and AOIs that are engaging the cognitive resources of the observer (Bylinski, et al., 2015). High performing students have exerted more visual attention in terms of the proportional fixation duration on the error line to confirm the compiler error message that error does exist in that line. This finding corroborates the findings of Chandrika & Amudha (2017) and Sharif et al. (2012) that experts tend to concentrate more on areas where the errors are located while novices read the codes more broadly. In contrast with the findings of the latter, Nivala, et al. (2016) and Turner, et al. (2014) observed that novices spent more time on the buggy lines of code than experts. The difference in the findings of this study with previous studies may be related to the characteristics of the stimuli used and the tasks employed by the researchers during the eye tracking experiment. Furthermore, the number of fixations on an AOI can be linked to its importance (Bylinski, et al., 2015). Thus, more fixation counts can be observed from the high performing students on the error lines. This result is in line with the findings of (Chandrika & Amudha, 2017; Sharif, et al., 2012; Turner, et al., 2014), that experts or advanced programmers have more fixation counts on the buggy lines of code.

3.2 Visual Attention Patterns on the Compiler Error Messages

An independent samples t-test was performed to determine the difference of the visual attention of high and low performing students on the compiler error messages in terms of the average proportional fixation duration. The result of the statistical analysis revealed that there was no significant difference in the visual attention of the high (*M* = 0.023, *SD* = 0.016) and low performing students (*M* = 0.018, *SD* = 0.024), *t*(16.066) = -0.696, *p* = .497. The result suggests that the average proportional fixation duration on the compiler error messages of the high and low performing students is similar for both groups across all programs. The same statistical test was used to determine the difference of the visual attention of the groups on the compiler error messages in terms of the average proportional fixation count. The result revealed that there was no significant difference in the visual attention of the high (*M* = 0.025, *SD* = 0.015) and low performing students (*M* = 0.018, *SD* = 0.022), *t*(16.256) = -0.926, *p* = .368. The result suggests that the average proportional fixation counts is similar for both high and low performing students across all programs.

The results of the analysis of the visual attention patterns imply that both high and low performing students have similar average proportional fixation durations and fixation counts on the compiler error messages. Although no significant difference was observed, high performing students have slightly higher average proportional fixation duration and fixation count on the compiler error messages. The visual behavior of high performing students may be related to the field-independent (FI) cognitive style theory of human cognition. The FI individuals tend to choose a more analytical processing approach and they pay attention to relevant details. Conversely, the visual behavior of low performing students may be related to field-dependent (FD) cognitive style wherein they choose a more holistic way of processing visual information and experience difficulties in identifying relevant details in the complex visual stimulus (Raptis et al., 2017).

4. Conclusion and Future Works

This study contributes to the evidence of the effectiveness of eye tracking as a method to enrich computing education research. The analysis conducted using the proportional fixation duration and fixation count provided considerable insights about the visual attention patterns of high and low performing students in processing compiler error messages. High performing students exert more visual attention on the lines indicated by the compiler error messages. They prefer a more analytical processing approach and pay attention to details while low performing students choose a more holistic approach. We may conclude that more proficient students read error messages as much as the less proficient students but limit their reading to the line number and not to the whole description of the error message. By exploring the visual strategies employed by the high performing students using eye tracking data, we could develop learning materials and activities that could help low performing students improve their code reading and debugging skills. Further, debugging should be taught as a program comprehension task rather than a search task. Students should focus their attention on the lines mentioned in the compiler error message to identify source code defects because it provides feedback on the location and description of the error. Teaching students to consciously employ debugging strategies would enhance their debugging ability and will help in increasing the retention rate of students taking programming courses.

Future analysis on the collected eye tracking data will be conducted to determine if students with different programming ability, self-efficacy levels, and personality characteristics vary in the way they read and process compiler error messages.

Acknowledgements

We thank Ateneo de Manila University and Pangasinan State University administration for allowing us to conduct the experiment for this study. We also thank the Dagstuhl Seminar 22402 on Foundations for a New Perspective of Understanding Programming that enabled us to identify the shortcomings in our previous study design and helped us improved the study design of this paper.

References

Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E., & Parnin, C. (2017, May). Do developers read compiler error messages?. In 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE) (pp. 575-585). IEEE.

Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P-M., Pearce, J. L & Prather, J. (2019). Compiler error messages considered unhelpful: The landscape of text-based programming error message research. Proceedings of the working group reports on innovation and technology in computer science education, 177-210.

Becker, B. A., Murray, C., Tao, T., Song, C., McCartney, R., & Sanders, K. (2018, February). Fix the first, ignore the rest: Dealing with multiple compiler error messages. In Proceedings of the 49th ACM technical symposium on computer science education (pp. 634-639).

Bol, N., Boerman, S.C., Romano Bergstrom, J.C., & Kruikemeier, S. (2016). An Overview of How Eye Tracking Is Used in Communication Research. In: Antona, M., Stephanidis, C. (eds) Universal Access in Human-Computer Interaction. Methods, Techniques, and Best Practices. UAHCI 2016. Lecture Notes in Computer Science(), vol 9737. Springer, Cham. https://doi.org/10.1007/978-3-319-40250-5_40

Bylinskii, Z., Borkin, M. A., Kim, N. W., Pfisher, H., and Oliva, A. (2015). Eye fixation metrics for large-scale evaluation and comparison of information visualizations. In Eye Tracking and Visualization, eds M. Burch, L Chuang, B. Fisher, A. Schmidt, and D. Weiskopf (Cham: Springer), 235-255. Doi: 10.1007/978-3-319-47024-5_14

Chandrika. K. R., and Amudha, J. (2017). An Eye Tracking Study to Understand the Visual Perception Behavior while Source Code Comprehension. International Journal of Control Theory and Applications. International Science Press vol. 10(19), 169-175.

Denny, P., Luxton-Reilly, A., & Carpenter, D. (2014). Enhancing syntax error messages appears ineffectual. In Proceedings of the ITiCSE '14: Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 273-278).

Du Boulay, B., & Matthew, I. (1984). Fatal error in pass zero: How not to confuse novices. Behaviour & Information Technology, 3(2), 109-118.

Dy, T., & Rodrigo, M. M. (2010, October). A detector for non-literal Java errors. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research (pp. 118-122).

Marceau, G., Fisler, K., & Krishnamurthi, S. (2011, October). Mind your language: on novices' interactions with error messages. In Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software (pp. 3-18).

Nienaltowski, M. H., Pedroni, M., & Meyer, B. (2008, March). Compiler error messages: What can help novices?. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (pp. 168-172).

Nivala, M., Hauser, F., Mottok, J., and Gruber, H. (2016). Developing visual expertise in software engineering: An eye tracking study. *2016 IEEE Global Engineering Education Conference (EDUCON)*, 613-620.

Prather, J., Pettit, R., McMurry, K. H., Peters, A., Homer, J., Simone, N., & Cohen, M. (2017, August). On novices' interaction with compiler error messages: A human factors approach. In Proceedings of the 2017 ACM Conference on International Computing Education Research (pp. 74-82).

Raptis, G. E., Katsini, K., Belk, M., Fidas, C., Samaras, G., & Avouris, N. (2017). Using Eye Gaze Data and Visual Activities to Infer Human Cognitive Styles: Method and Feasibility Studies. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (UMAP '17). (Bratislava Slovakia, 2017), ACM, New York, NY, USA, 164-173.

Sharif, B., Falcone, M. and Maletic, J. I. (2012). An eye-tracking study on the role of scan time in finding source code defects. In Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA'12), ACM, 381–384.

Tablatin, C. L. S., & Rodrigo, M. M. T. (2022). Identifying Code Reading Strategies in Debugging using STA with a Tolerance Algorithm. APSIPA Transactions on Signal and Information Processing, 11(1).

Turner, R., Falcone, M., Sharif, B., and Lazar, A. (2014). An eye- tracking study assessing the comprehension of C++ and Python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14)*. (Florida, 2014) ACM, New York, NY, USA, 231-234.

Villamor, M. M., & Rodrigo, M. M. T. (2022). Predicting Pair Success in a Pair Programming Eye Tracking Experiment Using Cross-Recurrence Quantification Analysis. APSIPA Transactions on Signal and Information Processing, 11(1).

Vosskühler, A. (2009). OGAMA description (for Version 2.5). Berlin, Germany: Freie Universität Berlin, Fachbereich Physik. Retrieved April 23, 2023 from http://www.ogama.net/sites/default/files/pdf/OGAMA-DescriptionV25.pdf