An Eye Tracking Perspective on How Developers Rate Source Code Readability Rules

Cole S. Peterson, Kang-il Park, Isaac Baysinger, Bonita Sharif
University of Nebraska - Lincoln
Lincoln, Nebraska USA
{cole.scott.peterson, kangil.park, isaacbaysinger}@huskers.unl.edu, bsharif@unl.edu

***Abstract*—Writing readable source code is generally considered good practice because it reduces comprehension time for both the original developer and others that have to read and maintain it. We conducted a code readability rating study using eye tracking equipment as part of a larger project where we compared pairs of Java methods side by side. The methods were written such that one followed a readability rule and the other did not. The participants were tasked with rating which method they considered to be more readable. An explanation of the rating was also optionally provided. Eye tracking data was collected and analyzed during the rating process. We found that developers rated the snippet in the pair of methods that avoided nested if statements as more readable on average. There was no clear preference in the use of do-while statements. In addition, more developer fixation attention was on the snippet that avoided do-while loops and the snippet pairs relating to nested if statements had more equal fixation attention across the snippets.**

***Index Terms*—source code readability, program comprehension, eye tracking**

I. Introduction

Code that is difficult to read often has issues that remain undetected. Posnett et al. present a reflection of how studies an models on code readability evolved over the years [4] starting with the work of Buse and Weimer [2]. Scalabrino et al. [6] conducted a study that measured the effects of incorporating textual features along with structural code features proposed by state-of-the art readability models. Their findings show that textual features of code complement structural ones – improving accuracy of readability models. Scalabrino et al. then take this one step further into assessing understandability (for which there are currently no metrics) [5]. They conducted another study evaluating existing and new code-related, documentation-related, and developer-related metrics. The study found no correlation, in most cases, between considered metrics and code understandability. Where correlation was observed, the magnitude was small.

Johnson et al. conducted an online study [3] assessing two code readability rules: minimizing nesting and avoiding do-while loops [1]. In this paper, we extend part of their work by providing an eye tracking perspective to how developers rate source code methods. The use of an eye tracker enables us to see exactly what parts of the code the developers are looking at while they are rating the readability rules. The main contributions of this short paper are a) the first eye tracking study that investigates how developers rate code methods written in different readability styles and b) analysis of how the rating relates to the eye movements on code during the rating task. This paper is part of a larger project of readability studies being conducted with human participants. We only report on the ratings of a subset of the population on a subset of the tasks.

The two research questions we seek to answer are as follows:

RQ1 Do developers prefer source code following code readability rules?

RQ2 What is the distribution of eye movements on the code snippets following and not following rules and how does this align with their preference?

RQ1 can help determine if following code readability rules can increase the confidence of a developer's understanding. RQ2 can help determine whether eye gaze can be used to determine developer preference for code readability rules and the cognitive effort needed for comprehension.

II. Rating Study Overview

As part of our larger project on readability, each rule (minimize nesting, avoid do-while) has four algorithmic problems, and each algorithmic problem has four solutions presented as Java methods. There are a total of 32 Java methods, 16 for each readability rule. For the scope of this paper, we only focus on the rating aspect of the project where each participant carries out 4 method comparison tasks associated with R1 and 4 method comparison tasks in R2. For each rule, the participant was shown two methods side-by-side on the screen (see Figure 1) where one method was following the rule and the other was not following the rule. The method's functionality was given in plain English at the top. The prompt assigned to them was to analyze the code and choose which method they thought was more readable with an optional comment justifying their choice. The code that followed the rule was always placed on the left but we found that participants transitioned to both code snippets before making their selection.

We used the Tobii X60 eye tracker running at 60Hz and Tobii Studio to record eye gaze on the code snippets that were displayed on the screen side by side. The responses were collected via a Google form after they were done analyzing the code snippets. We run the I-VT fixation filter on the raw gazes and then map each fixation to lines of source code. Each stimulus was presented as a static image and an area of interest (AOI) was created for every line in the source code.

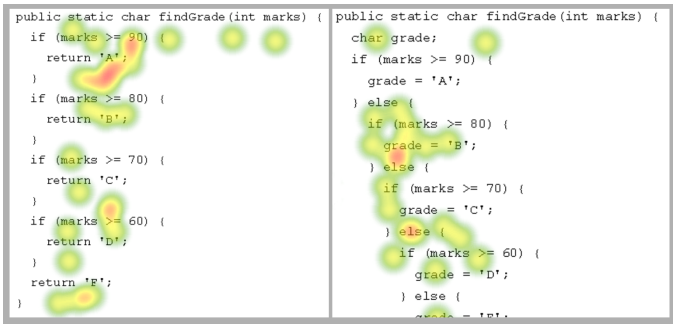

Fig. 1: A gaze heatmap from one participant for R1: minimize nesting with the left (V1) following the rule and the right (V2) not following the rule.

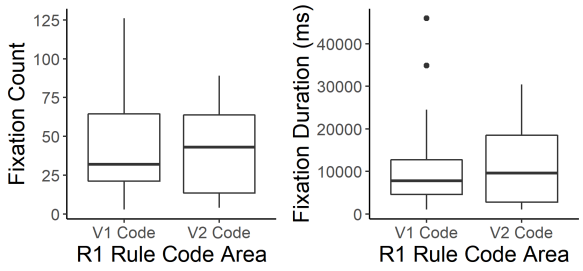

Fig. 2: Fixation counts and durations for a code snippet following R1 (V1 Code) and not Following R1 (V2 Code)

This mapping is done using eyeCode[1] using the fixations x and y coordinates. In this paper, we only analyze a subset of 14 participants on two stimuli. Of these, 4 were native English speakers and 11 rated their English proficiency as Good or Very Good. Nine participants were undergraduate students and 5 were graduate students. In terms of programming experience, 3 participants had actively programmed for less than a year, 7 had programmed between 1 and 3 years, and the remaining 4 programmed between 3 and 5 years.

III. Results and Discussion

RQ1 Results: Readability Rule Preference: Twelve out of fourteen participants in our study (85.71%) preferred the snippet that follows the readability rule R1: minimize nesting, and there was an even split (seven each, both 50%) between preference for the snippet that follows the readability rule R2: avoid do-while loops. The results corroborate the Johnson et al. study for R1, where 86.82% of participants responded with the rule-following snippet as having higher readability than the snippet that does not. R2's results somewhat differ from our data which suggest that there is some importance to avoiding do-while loops, as 67.44% of participants in that study [3] rated snippets that follow R2 as having higher readability. However, the study also found no significant impact on other readability metrics and noticed many participants commented that the selection came down to mostly personal preference. Developers also displayed stronger opinions in their comments toward minimizing nesting versus avoiding do-while loops. For example, one participant's response regarding R1 was: "*Using singular if statements rather than if else helps to avoid nesting and makes the code easier to read and understand.*"

RQ2 Results: Eye Gaze Distribution and Preference: With respect to R1-minimize nesting, Figure 2 shows the average fixation count and duration. The code snippet that follows R1 (and avoids nested if statements) had an average of 47.9 fixations with 12.6 secs. of average fixation duration compared to the one that did not follow R1 (41.1 fixations and 11.3 secs. of fixation duration). The code that does not follow R1 is longer and has 21 lines compared to the 15 lines in the code snippet that follows R1. For R2-avoid do/while, we found that the on average participants looked at the code snippet that followed the rule more often than they looked at the code snippet that did not follow the rule. The code snippet that follows R2 (and avoids do/while) had an average of 116.2 fixations with 32.0 secs. of average fixation duration compared to the one that did not follow R2 (75.6 fixations and 17.3 secs. of fixation duration). The code snippet lengths in this case differed by only 1 line. We observe that the fixation attention for the nested if statements was split evenly, whereas more fixation attention was given to the code snippet that did not use a do-while loop.

As part of future work, we will extend this analysis to all participants on all the different code snippet comparisons and provide a detailed eye tracking assessment of different chunks in the code that make up the readability rules. This will provide us finer grained insight into what specific lines were looked at during the rating process as well as during comprehension tasks.

[1]https://github.com/synesthesiam/eyecode

Acknowledgments

This work was supported in part by the National Science foundation grant numbers CCF 18-55756 and CNS 18-55753.

References

[1] D. Boswell and T. Foucher. *The Art of Readable Code: Simple and Practical Techniques for Writing Better Code*. " O'Reilly Media, Inc.", 2011.

[2] R. P. L. Buse and W. R. Weimer. Learning a metric for code readability. *IEEE Transactions on software engineering*, 36(4):546–558, 2010.

[3] J. Johnson, S. Lubo, N. Yedla, J. Aponte, and B. Sharif. An empirical study assessing source code readability in comprehension. In *IEEE ICSME*, pages 513–523, 2019.

[4] D. Posnett, A. Hindle, and P. Devanbu. Reflections on: A simpler model of software readability. *ACM SIGSOFT Software Engineering Notes*, 46(3):30–32, 2021.

[5] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and R. Oliveto. Automatically assessing code understandability. *TSE*, 47(3):595–613, 2021.

[6] S. Scalabrino, M. Linares-Vásquez, R. Oliveto, and D. Poshyvanyk. A comprehensive model for code readability. *Journal of Software Systems*, 30(6):e1958, 2018.