Effects of Coding Norm Violations on Visual Effort, Trustworthiness Perceptions, and Reuse Intentions

Sasha M. Willis
Wright State University
willis.149@wright.edu

Sarah A. Jessup
Consortium of Universities
jessup.11@wright.edu

Gene M. Alarcon
Air Force Research Laboratory
gene.alarcon.1@us.af.mil

Michael A. Lee
GDIT
mlee209@student.gsu.edu

Abstract

Efficient evaluation strategies are essential when reviewing computer code for potential reuse. Previous researchers have examined the factors that influence these assessments. However, researchers have yet to empirically demonstrate the direct influence of the specific factors that affect visual/cognitive effort, which can be inferred through eye tracking metrics.

Programmers were recruited to complete a Java code review task, providing evaluations for a code file's trustworthiness and reusability after various errors had been introduced to the file's source reputation, readability, and organization. Analyses of the eye-tracking data revealed increases in fixation counts and durations for manipulated code. An exploratory analysis of areas containing readability and organization errors revealed misuses of case and misuses of declarations garnered the most attention from participants relative to the rest of the code. Implications of the current study extend to recommendations for writing code that is easily reusable by decreasing the visual effort needed for code review.

Keywords: Information Processing, Eye Tracking, Human-Computer Interaction, Software Reusability, Trust in Code.

1. Introduction

The interest in finding efficient strategies for creating computer code has become increasingly accelerated following a continuous expansion in the demand for automated systems and novel programming applications. Reusing previously written code and writing reusable code are practices that have proven beneficial due to increased efficiency in the code creation process (Frakes & Kang, 2005; Mäkitalo et al., 2020). Code that reviewers perceive as trustworthy is more likely to be reused, and ensuring that programmers appropriately calibrate their trust in existing code helps maintain security and decrease vulnerabilities (Ryan et al., 2019).

In a cognitive task analysis, three primary factors were identified that influence a programmer's code trustworthiness perceptions: performance, reputation, and transparency of the code (Alarcon et al., 2017b). Performance is the ability of the code to meet the objective of the project. Reputation assessments are based on external information about the code that may or may not be related to the code itself. Transparency is defined by the understandability of the code from viewing or reading it, which includes aspects of the code related to organization and readability (e.g., style, architecture, etc.; Busjahn et al., 2011). As defined by Alarcon and colleagues (2017b), readability is the ability of the programmer to evaluate and comprehend the intended behavior of the code, and organization concerns aspects of the code that relate to its orientation and control structure. The current study specifically manipulated transparency by introducing errors to the readability and organization of the code and manipulated reputation by altering the code source's stated reputation. Eye-tracking data can provide valuable insight into programmers' perceptions of the code and cognitive processing (Busjahn et al., 2011), and thus could reveal how these coding conventions might affect a programmer's trustworthiness perceptions and reuse intentions for a piece of code.

Importantly, data from a programmer's eye movements made during a code review task could indicate the reviewer's type or depth of cognitive processing (Eckstein et al., 2017). The current study leverages eye-tracking data to infer the visual effort that is required of participants while completing a code review task and explore how differences in eye movements vary as a function of degree or type of violation made to the code. We further attempt to replicate the effects of introduced errors made to a code's source, readability, and organization and report how these manipulations affect a code's perceived trustworthiness and reusability from previous research (Alarcon et al., 2017a). We also report the specific types of manipulations that seem to be the driving forces of attention capture during the code review process.

2. Related Literature

2.1. Eye-tracking Metrics and Cognitive Processing

Eye-tracking metrics derived from gaze location and time data allow researchers to identify what information observers are looking at and for how long. Eye-tracking technology can provide a means of inferring the depth of cognitive processing and amount of visual effort required for a task (Binkley et al., 2013; Sharafi et al., 2015). Researchers can leverage data from this technology to approximate cognitive processes such as text comprehension, attention allocation, and approaches to problem solving (Eckstein et al., 2017; Raney et al., 2014). Recently, researchers have used eye-tracking technology to examine how programmers review code. These studies include an examination of the differences in code comprehension between expert and novice programmers (Jessup et al., 2021) and performance on code review tasks (Liu et al., 2020).

Gaze data can be analyzed in several ways; two such metrics include calculating the number of fixations made by a participant (i.e., fixation count; FC) and the time participants spend fixating on a stimulus (i.e., fixation duration). Fixation metrics can be analyzed over an entire visual stimulus or for specific areas of interest (AOIs), which are defined by the researcher as particularly relevant regions of the visual stimulus (Orquin et al., 2016). Fixations are commonly used as a metric in eye-tracking research because much of the cognitive processing of visual input occurs during these momentary pauses in eye movements (Hauser et al., 2018). For instance, FC has been used as a measure of visual effort and can signal if a stimulus is viewed often, as it may be relatively complex or interesting (Uzzaman & Joordens, 2011). Similarly, variables derived from fixation durations approximate how long it takes to evaluate a stimulus (Sharafi et al., 2015; Hauser et al., 2018). As information processing increases in complexity, observers fixate longer (i.e., longer average fixation durations; AFD; and total fixation durations; TFD) and more often (i.e., increased FC) on relevant material (Raney et al., 2014).

2.2. Effects of Code Reputation and Transparency

In past research on code review tasks, Alarcon and colleagues (2017a; 2020a; Walter et al., 2017) have focused on tasks that have manipulated or measured a code's reputation (by manipulating source), and transparency (by manipulating readability and organization). They have theorized programmers use different processing strategies depending on the presence and type of coding norm violations existent within a code file (Alarcon & Ryan, 2018). When evaluating computer code for trustworthiness and reusability, there is a balance between making well-informed yet time-efficient decisions. Combined with other research focusing on how these manipulations affect eye movements and behaviors, we merge these research streams to develop the goals and expected outcomes for the current study.

In the following subsections, we review research on eye tracking data following manipulations made to code source, readability, and organization. We propose that the introduced errors to each of the three manipulations will result in higher visual effort. We note that FC, AFD, and TFD are viewed as proxies rather than direct indicators of cognitive processes and visual/cognitive effort; see Holmqvist et al., 2011). When there was sufficient extant literature that indicated a direction of differences across our manipulations, we formulate hypotheses below. For the relationships with less supporting research, we pose research questions of a more exploratory nature.

2.2.1. Source. We begin with our first manipulated variable of interest, code source, which is one aspect of the reputation factor (Alarcon et al., 2017a). Bertram and colleagues (2020) examined participants' scanning patterns while they reviewed repair patches made to a sample of Java code. The authors manipulated the code's source by randomly labeling the patches as being written by either a human or a machine (two of the six patches were in fact written by an automated program, whereas the other four were written by a human). The authors found no significant differences in eye-tracking metrics (FC, AFD, and average saccade length) or in overall time spent on the task between the two author labels. Interestingly, however, significant differences were found between human- and machine-labeled patches in the distribution of visual attention; the code's class and methods garnered more attention (as measured by higher FC, longer AFD, and shorter average saccade length) in the human-labeled patches. Because Bertram and colleagues' code source manipulation is quite different from the current study's operationalization of source reputation, we pose the following research question rather than a directional hypothesis:

RQ1: What effect, if any, does code source have on a) FC, b) AFD, and c) TFD?

As code source relates to trustworthiness and reuse intentions, Alarcon and colleagues found code labeled as originating from a reputable source led to higher trustworthiness perceptions (Alarcon et al., 2017a; Alarcon et al., 2020b; Walter et al., 2017) and increased reuse intentions (Alarcon et al., 2020b; Walter et al., 2017) compared to code from an unknown source.

H1: When code is from a reputable source, participants will a) rate the code as more trustworthy, and b) be more willing to reuse the code.

2.2.2. Readability. Connections between eye-tracking data and manipulations made to a code's readability have also been found. In one such study, Katona (2021) varied a piece of code's readability with "clean" code, which included proper naming schemas, readable implementation of functions, and correct code formatting, compared to "dirty" code, which was not as readable. The researcher also collected self-report data about the code in addition to eye-tracking data. Results revealed that the clean code was rated as easier to read and understand and resulted in lower average FC and lower AFD compared to the dirty code. However, this study neglected to discuss the specific types or instances of errors that were present in the dirty code, and the author did not include any analysis of eye-tracking data between specific regions of code that contained errors versus regions that did not contain errors. From Katona's evidence, we pose the following hypothesis:

H2: Participants will have a) fewer FC, b) shorter AFD, and c) shorter TFD when code readability is higher.

Additionally, Alarcon and colleagues found that higher code readability led to higher trustworthiness perceptions (Alarcon et al., 2017a; Walter et al., 2017) and increased reuse intentions (Walter et al., 2017). The following hypothesis is developed:

H3: When code is higher in readability, participants will a) rate the code as more trustworthy, and b) be more willing to reuse the code piece.

2.2.3. Organization. At present, there is little research investigating overall code organization using eye-tracking metrics. This is likely due to the inability of many eye-tracking programs to allow participants to scroll vertically on the computer screen. As such, there is a difficulty of presenting stimuli of sufficient length to study many types of large-scale organization errors while also staying within the confines of one vertical screen length. Taking a more granular approach, a limited number of eye-tracking studies were found relating to the subtypes of our chosen organization error types [selected from Alarcon and colleagues (2020a) and listed in the Measures section below]. Namely, a study by de Oliveira and colleagues (2020) found that computer code containing "Logic as Control Flow" patterns (which present a confusing order of logical operators in execution statements) caused a significant increase in visual attention. This was evidenced by increased gaze transitions to the execution statement as well as increased time on areas containing those patterns of code. Because of the relative novelty of our approach to studying the effects of code organization (e.g., code consisting of numerous nested blocks spanning multiple screen lengths), we pose the following for exploration:

RQ2: What effect, if any, does code organization have on a) FC, b) AFD, and c) TFD?

In contrast to their expectations and those that the eye tracking research above would suggest, Alarcon and colleagues found that higher code organization led to lower trustworthiness perceptions (Alarcon et al., 2017a; Walter et al., 2017) and lower reuse intentions in code (Walter et al., 2017). Despite these counterintuitive findings of higher organized code leading to worse perceptions, we believe that the changes (methodological and otherwise) made to the current study from these prior studies may align better with the original expectations of Alarcon and colleagues (Alarcon et al., 2017a; Walter et al., 2017). As such, we pose the following hypothesis:

H4: When code is higher in organization, participants will a) rate the code as more trustworthy, and b) be more willing to reuse the code piece.

2.2.4. Effects of Readability Versus Organization. It should be noted that Alarcon and colleagues did find interactions across their source, readability, and organization manipulations. Specifically, they found an interaction between organization and readability, such that when readability was low and organization was either high or low, participants trusted the code more than when code had a medium level of organization (Alarcon et al., 2020a). We believe that adding eye-tracking data will allow the current study to delve into the attention capture driven by these two types of manipulations, though no previous studies to our knowledge have measured how readability errors differently capture a program reviewer's attention as compared to organization errors. Therefore, we pose the following as an exploratory research question:

RQ3: Do participants differently attend to readability versus organization manipulations or their subtypes (AOIs, listed in the Measures section below), as compared to the full stimulus?

2.3. Current Study

Although the results from Alarcon and colleagues were replicated across several studies (Alarcon et al., 2017a; Alarcon et al., 2020b; Walter et al., 2017), there are still issues with these replications. First, the researchers used within-subjects experimental designs in which each piece of code that participants reviewed was a unique Java class and may have had its own initial trustworthiness level (e.g., code that was high risk), regardless of manipulations. Second, these prior studies used a metric of overall page time to capture participants' time spent on code, though this measure was not sensitive enough to fully capture differences in visual effort and attention across their manipulations. That is, their page time measure was not able to differentiate between when the participants were processing the code stimuli versus other aspects of the study, such as the code evaluation survey items located beneath the code on the same page.

In the current study, we ameliorate some of the limitations of Alarcon and colleagues' previous research (Alarcon et al., 2017a; Alarcon et al., 2020b; Walter et al., 2017) and further investigate their conclusions by modifying several aspects of their experimental approach. First, we used a between-subjects design in which participants evaluated a single piece of code that was manipulated by each combination of source (reputable, unknown), readability (high, medium, low), and organization (high, medium, low) factors. This adaptation from previous research (Alarcon et al., 2017a) removes the risk of confounding code functionality or inherent risk with our variables of interest. Second, this study is differentiated from Alarcon and colleagues (2020a) by incorporating eye-tracking technology to evaluate the visual effort needed for evaluating code. When eye-tracking technology is incorporated, it is also possible to separate the amount of time participants spend reviewing the code stimuli from the time they spend outside of the stimulus region (e.g., responding to survey items and short answer questions). Eye tracking also allows us to see which manipulations (i.e., subtypes of readability or organization error categories) are the driving forces of attention capture and more effortful processing. Contributions of our research include:

1. Leveraging eye-tracking data to infer the visual effort that is required during a code review task.
2. Exploring how differences in eye movements vary as a function of degree or type of violation made to the code.
3. Attempting to replicate the effects of errors introduced to a code's source, readability, and organization and report how these errors affect a code's perceived trustworthiness and reusability.
4. Finding the specific types of errors that seem to be the driving forces of attention capture during the code review process.
5. Merging previous research streams to develop the goals and expected outcomes for the current study.

3. Method

3.1. Participants

Participants were recruited (N = 52) during initial data collection via flyer postings, email, and by word of mouth from a Midwest university and local industries to obtain a sample of both student and professional programmers. Participants were required to have at least three years of programming experience and be familiar with the Java coding language. We removed six participants for poor eye-tracking data quality (designated by meeting a minimum track loss threshold) and two participants for not meeting the minimum experience requirements, leaving a total of 44 participants for our analyses. Of these participants, 79.55% identified as male, with a mean age of 30.34 years (SD = 10.54) and a range of 4–20 years of programming experience.

3.2. Task and Procedure

After completing informed consent procedures, participants were asked to sit approximately 70 cm from a computer monitor (1920x1080 screen resolution), which had been affixed with a Smart Eye Aurora remote eye tracker (60-Hz sample rate) and was running the iMotions Screen-Based Eye Tracking Module (version 7.1). Through the iMotions software, participants completed a four-point gaze calibration procedure. Prior to the code review task, participants were informed that: 1) all comments were removed from each code piece (to reduce biases related to commenting practices and to deter participants from using comments to infer the codes' functionality rather than the code itself); 2) all packages had been modified to remove the original source origin; 3) all code successfully compiles; and 4) there would be additional information above each piece of code. After being presented with instructions, participants reviewed and evaluated one target Java code file, which was altered from the original file by a randomly assigned combination of the study's manipulated variables (see below).

In an offline version of a Qualtrics survey, the target Java code was presented to participants as a screen-captured image. Although participants could not modify nor interact with the code other than directly viewing the image, the iMotions program did allow the participants to vertically scroll through the approximate 350 lines of code. The iMotions software is capable of recording gaze locations as X, Y pixel coordinates of the stimulus image rather than X, Y pixel coordinates of the screen, allowing the researcher to map where on the image the participant is looking, even during scrolling behaviors.

The target code file was preceded by two distractor Java classes and followed by three additional distractor Java classes, each presented on a separate page, to ensure that participants were not aware of the location of the experimental stimulus. Above each code piece was a description of the code's source. Once they had finished viewing each code file, participants were asked to provide their evaluations of it on the same page as well as a brief description of their understanding of the code's function. All code sections were written to contain no major errors or issues, excluding the target piece of code. Participants were thanked for their time, compensated $50 USD, and dismissed from the study following the completion of the task.

3.3. Manipulations

The original target Java code file was manipulated across three different parameters: source (reputable, unknown), readability (high, medium, low), and organization (high, medium, low) according to previous research (see Alarcon et al., 2020a). This created 18 total conditions to which each participant was randomly assigned (e.g., one condition was code from a reputable source with medium readability and low organization). The source factor was displayed to participants above each code piece as reading either "Source: Reputable" or "Source: Unknown," referring to the reputation of the code's origin repository. In reality, all target code piece variations originated from the same repository and were degraded by the same programmer. The high readability and high organization conditions contained no changes to the code for those error types, while the medium and low iterations of each altered the code across different parameters, increasing in severity from the medium to low conditions.

3.4. Measures

Three eye-tracking metrics were calculated across the vertical pixel range containing the Java code. FC was determined by counting the number fixations made within this pixel range. AFD was calculated for each participant by computing the average duration (in milliseconds) of all fixations located within this pixel range. TFD was calculated for each participant by summing the duration (in seconds) of all fixations within this pixel range.

In addition to analyzing the eye-tracking data across the entire code stimulus, we also defined 10 AOIs, each consisting of both adjacent and non-contiguous sections of code that had readability or organization errors introduced on those lines. AOIs were composed of each span of vertical pixels that contained a line of manipulated code, grouped into two higher-order categories (readability and organization manipulations) as well as four lower order categories for each of these two manipulations. Consistent with past research (Alarcon et al., 2017a), readability was manipulated by increasing the number of offending instances of: a) misuse of case (e.g., for packages, methods, variables); b) misuse of braces (e.g., missing a space before an opening or closing brace); c) misuse of indentation (e.g., inconsistent indentation); and d) improper line wrapping (e.g., use of too many and unnecessary blank lines). Organization was manipulated by introducing errors of: a) misuse of declarations (e.g., import statements used improperly) b) ambiguous control flow (e.g., unnecessary or confusing nesting of blocks); c) improper exception handling (e.g., ignoring exceptions); and d) inconsistent block spacing. We report the area-normalized Rate of Relevant Fixations (RRF) by calculating the proportion of fixations that fall within each of the AOIs (i.e., the higher-order AOIs consisting of all Readability and all Organization errors and the four subtypes for each) compared to the number of fixations made over the entire code stimulus. AOI size was area-normalized by dividing the number of vertical pixels in the AOI by the total number of vertical pixels for the entire code piece (formula adapted from Smilek et al., 2006).

Perceptions of the code's trustworthiness as well as participants' willingness to reuse the code were assessed with the following questions below each piece of code, respectively: a) "How trustworthy is the code?", responding with a 7-point Likert scale ranging from 1 ("Not at all trustworthy") to 7 ("Very trustworthy"); and b) "Would you use this code?", using the binary options "Use" or "Don't use" to respond.

4. Results

We conducted a series of between-subjects analysis of variance (ANOVAs) using a 2 (Source: Reputable, Unknown) × 3 (Readability: High, Medium, Low) × 3 (Organization: High, Medium, Low) design to determine the effects of the manipulations on eye-tracking metrics (FC, AFD, TFD) and self-reported Trustworthiness Perceptions of code. Post hoc comparisons, when conducted, were adjusted with the Bonferroni correction, and estimated marginal means were used for computation of reported mean differences between groups. To analyze the manipulations' effects on Reuse Intentions, we conducted a generalized estimating equation (GEE), as the outcome variable is binary and would violate the assumptions of ANOVA testing.

4.1. Eye-Tracking Metrics

4.1.1. FC. We conducted a three-way between-subjects ANOVA on FC. There was a statistically significant main effect of Source [$F(1, 26) = 7.78$, $p = .010$, $\eta_p^2 = .23$] on FC, such that code from an Unknown Source had significantly more fixations than code from a Reputable Source ($M_{diff} = 221.35$, $SE = 79.38$, $p = .010$). The main effect of Organization was marginally significant [$F(2, 26) = 3.12$, $p = .061$, $\eta_p^2 = .19$], and the main effect of Readability was not statistically significant. Results are illustrated in Figure 1.

These main effects were qualified by two interactions. There was a statistically significant interaction between Source and Readability [$F(2, 26) = 7.86$, $p = .002$, $\eta_p^2 = .38$]. Participants fixated on the code more when it was from an Unknown Source compared to a Reputable Source in both the High ($M_{diff} = 383.56$, $SE = 160.15$, $p =.022$) and Low Readability conditions ($M_{diff} = 344.29$, $SE = 162.42$, $p = .041$). There were no differences between the Source manipulations when Readability was Medium. There was also a significant interaction between Readability and Organization [$F(4, 26) = 3.56$, $p = .020$, $\eta_p^2 = .35$]. Participants had more fixations when Organization was High compared to Medium for code that was High in Readability ($M_{\text{diff}} = 497.40$, $SE = 195.40$, $p = .046$). There were no other significant interactions.

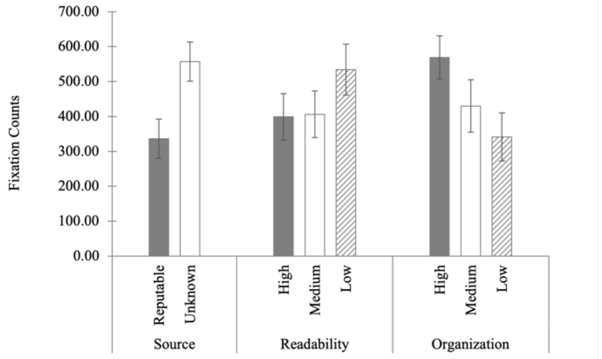

Figure 1. Fixation Counts by main effects of Source, Readability, and Organization. Bars represent estimated marginal means. Error bars represent standard errors.

4.1.2. AFD. We conducted a three-way between-subjects ANOVA on AFD. No statistically significant main effects nor interactions were found (all p's > .220).

4.1.3. TFD. We conducted a three-way between-subjects ANOVA on TFD. There was a statistically significant main effect of Source [$F(1, 26) = 7.19$, $p = .013$, $\eta_p^2 = .22$] and Organization [$F(2, 26) = 3.64$, $p = .040$, $\eta_p^2 = .22$] on TFD. However, no significant main effect of Readability on TFD was detected. Post hoc comparisons illustrated that participants fixated longer when Source was Unknown compared to Reputable ($M_{\text{diff}} = 99.45$, $SE = 37.09$, $p = .013$). Participants also fixated longer when Organization was High compared to Low ($M_{\text{diff}} = 116.59$, $SE = 43.24$, $p = .036$). Results are presented in Figure 2.

There was a significant Source by Readability interaction [$F(2, 24) = 5.47$, $p = .010$, $\eta_p^2 = .30$]. Participants fixated on the code for longer when it was from an Unknown Source compared to a Reputable Source in the High Readability condition ($M_{diff} = 184.54$, $SE = 74.76$, $p = .018$). There was also a significant Readability by Organization interaction [$F(4, 26) = 3.33$, $p = .025$, $\eta_p^2 = .34$]. Participants fixated longer when Organization was High compared to Medium in the High Readability condition ($M_{\text{diff}} = 257.99$, $SE = 87.56$, $p = .017$). No other interactions were significant.

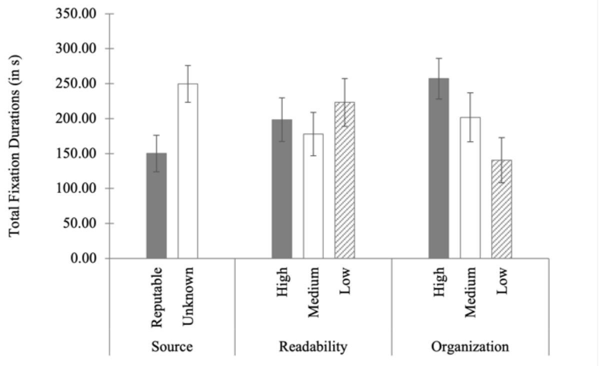

Figure 2. Total Fixation Duration (in seconds) by main effects of Source, Readability, and Organization. Bars represent estimated marginal means. Error bars represent standard errors.

4.1.4. AOIs. When errors were introduced to the code, the area-normalized RRF for those AOIs are reported. Overall, Readability errors ($M = 1.91$, $SD = 0.26$) were attended to significantly more often than Organization errors ($M = 0.97$, $SD = 0.31$) compared to the full code piece [$F(1, 64) = 5.49$, $p = .022$, $\eta_p^2 = .08$]. Looking further into the subcategories of Readability, the most attended Readability error type was *misuse of case*, with a mean area-normalized RRF of 2.49 ($SD = 1.87$) fixations. This was followed by *misuse of braces*, *misuse of indentation*, and *improper line wrapping*. For Organization errors, the most attended type was *misuse of declarations* ($M = 1.48$, $SD = 1.08$), as the mean area-normalized RRF was the highest for this subgroup. This was followed by *inconsistent block spacing* and *ambiguous control flow*. Lastly, *improper exception handling* did not seem to elicit a relevant proportion of fixations as compared to the entire Java class; however, the number of lines containing these errors was comparatively low, so more research is needed to obtain an accurate measure of relative attention allocation. Descriptive statistics are shown in Figure 3.

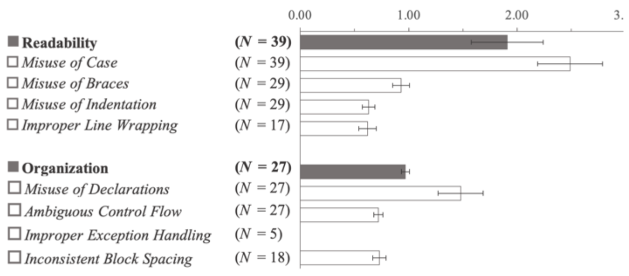

Figure 3. Area-normalized mean RRF by error type. Gray bars represent general type of error, while white bars represent specific forms of errors. Error bars represent standard errors.

4.2. Perceived Trustworthiness Ratings

We also evaluated whether our manipulations of Source, Readability, and Organization had any effect on Perceived Trustworthiness Ratings. There was a marginally significant main effect of Readability on Perceived Trustworthiness Ratings [$F(2, 26) = 3.28$, $p = .053$, $\eta_p^2 = .20$]. Participants had higher Trustworthiness Ratings with the code that was High in Readability compared to Low Readability ($M_{\text{diff}} = 1.64$, $SE = 0.64$, $p = .050$). Effects of Source and Organization were not significant. Results of the analyses are depicted in Figure 4.

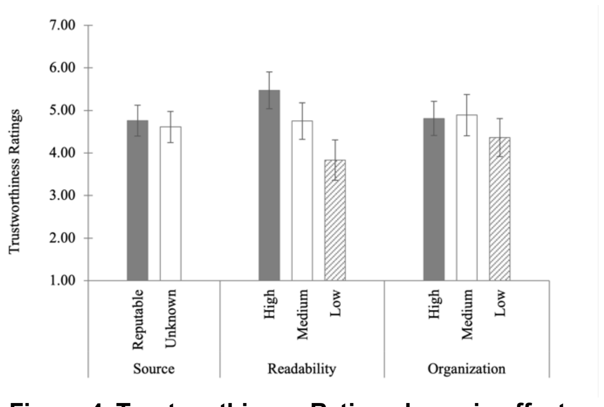

Figure 4. Trustworthiness Ratings by main effects of Source, Readability, and Organization. Bars represent estimated marginal means. Error bars represent standard errors.

4.3. Reuse Intentions

Due to the binary nature of the Reuse Intentions outcome variable, a GEE analysis was conducted to examine whether there were differences in Reuse Intentions across the Source, Readability, and Organization manipulations. We found a significant effect of Readability on Reuse Intentions [Wald χ^2 (2, $N = 44$) = 6.55, $p = .038$]. Participants in the High (Use: 11, Don't Use: 4) and Medium Readability (Use: 11, Don't Use: 4) conditions had significantly higher Reuse Intentions than those in the Low Readability condition (Use: 4, Don't Use: 10). Effects of Source and Organization were not significant.

5. Discussion

The current study sought to explore differences in visual effort across errors introduced to source, readability, and organization of computer code in a between-subjects design utilizing eye-tracking metrics, as well as self-reported perceived trustworthiness and reuse intentions of computer code. More specifically, this study explored eye-tracking and self-report data to infer differences in visual/cognitive effort and attention allocation when the source, readability, and organization of computer code were manipulated to varying degrees. We first review the behavioral metrics from eye-tracking data compared to previous research, then discuss the self-report results of trustworthiness and reuse intentions.

Interestingly, we only found a marginal effect of readability on trustworthiness, with differences between high and low readability conditions, whereas source and organization did not have significant effects on trustworthiness. The relationship between trustworthiness and source may not be significant in the current study due to the low-risk nature of the code. Source may be of higher importance in code that will be used in high-risk scenarios. The relationship between organization and trustworthiness may not have been significant because errors introduced to the code's organization were less likely to cause bugs or compiling issues in the code (e.g., inconsistent block spacing) and could have been attributed to the original programmer's writing style rather than ability.

We found that certain subtypes of readability and organization errors were fixated on relatively more than the rest of the code piece as compared to other subtypes within the same higher-order manipulation category. Specifically, we found misuse of case was the most fixated aspect of readability. Since Java is a case-sensitive programming language, it is important that programmers follow proper naming conventions. It is reasonable that lines degraded by misuse of case would attract a relatively higher amount of attention than other areas of code because these instances (e.g., incorrect case for packages, classes, methods, variables, and constants) could result in program errors and require debugging procedures. Misuse of case errors may also be more visually salient compared to other readability error types, such as misuse of braces (e.g., missing a break before a closing or opening brace), and are more problematic for program functionality compared to improper length and wrapping (e.g., use of too many blank lines).

We found that misuse of declarations was the most fixated aspect of organization, relative to the entire Java class. Lines with misuse of declarations errors drew relatively more attention than the rest of the code as compared to the other organization error subtypes. The higher amount of attention drawn to this type of error is also understandable because it would be difficult for programmers to understand program functionality when, for example, variables are not initialized as soon as possible and could cause errors when running the program (e.g., import statements used improperly). These errors may also be more visually salient than other types of organization issues (e.g., compressed if statements or unnecessary use of "break" or "continue") because they often appear at the beginning of a block of code, where much of the information about the function or purpose of that block is located, rather than in the middle where the execution of the block's contents are contained. Software engineers tend to focus on elements of the code that provide key information about the code's functionality, which are referred to as beacons (elsewhere referred to as notations, naming schemas, comments, etc.; Katona, 2021; Crosby et al., 2002; Busjahn et al., 2011). Beacons can facilitate different depths of processing depending on factors such as a programmer's level of experience, the program's purpose or functionality, and the type of information contained within the beacon (e.g., comments versus variable names; Crosby et al., 2002).

There are apparent differences between the results from the current study and previous research (Alarcon et al., 2017a; Alarcon et al., 2020b; Walter et al., 2017) which may have occurred for several reasons. First, in the current study participants only had to review one target piece of code, whereas in previous research they were required to review 18 pieces of code (Alarcon et al., 2017a; Walter et al., 2017). Differences between the results of the current study and previous studies may have resulted from a different number or different type of introduced readability and organization errors. The current study did not replicate the same number of manipulations as previous research (e.g., same number of lines containing a misuse of case error, etc.). It is also possible that the participants may have viewed each code piece in relation to the other code pieces being reviewed within Alarcon and colleagues' previous research. This may have influenced programmers' relevant processing times and code perceptions, as they may have used the other pieces they were reviewing as a sort of relative reference.

Second, the code's functional purpose was held constant in the current study. Whereas previous research (Alarcon et al., 2017a) had manipulations specific to each piece of code type (i.e., each code piece had a different basic function or purpose), the current study manipulated the variables of interest across participants on the same original piece of code. Previous research demonstrated that when organization was manipulated, both high and low organization led to programmers spending more time on the code compared to when the code contained a medium level of organization errors (Alarcon et al., 2017a). However, when code was highly organized in the current study, participants spent more time on the code, as evidenced by higher FC and longer TFDs, compared to when code was either medium or low in organization. This may have occurred for several reasons. First, the code of interest in the current study is not used in a high-risk environment (e.g., servers), The nature of the code itself has been noted to be of importance in previous research, such that high-risk code (e.g., server code, bank wire code, etc.) was scrutinized more than low-risk code, especially when it violated computer programming norms (Alarcon et al., 2017a). The code manipulated in the current study was considered low risk, so manipulations to lower risk code may not have the same impact on processing strategies or visual effort during code evaluation. Second, previous research has relied on time spent on the page rather than tracking of time spent on the code. In the current study we were able to differentiate between time spent reviewing the code and time spent answering the questions about the code using eye-tracking data. This may have influenced the assessment of time spent on the code in previous research, as the time also included time answering questions about the code. Code that is less organized may take more time to write about, as there are more questions about its processes.

5.1. Limitations and Future Research

The current study is not without limitations. First, the study is likely underpowered to detect many of the interactions and effects proposed. The onset of the COVID-19 pandemic has hindered in-person data collection in many areas of research, including ours. As high-accuracy eye trackers necessitate in-person data collection, the current study may be not have reached the sample size needed for adequate power for the conducted analyses. More specifically, with a final sample size of $N = 44$ and a $2 \times 3 \times 3$ within-subjects design, we were left with just three participants per unique cell to test the interaction effects between the readability, organization, and source manipulations. For example, the readability manipulation was marginally significant despite accounting for 20% of the variance in trustworthiness perceptions. Future researchers may benefit from a dichotomization of the readability and organization variables (rather than having three levels of each of these factors) to lower the required sample size needed to find significant effects. Despite our small sample size, we believe the significant effects that we detected do promote the current state of research in this area.

Second, although Alarcon and colleagues (2017a) have demonstrated different effects of readability and organization on various outcomes (e.g., trustworthiness perceptions) and that the two are distinct constructs, it may be difficult for a reader to parse these factors apart. We therefore suggest that future research add subjective rating items for each of these factors to serve as manipulation checks as well as tests for the perceptual impacts of these manipulations.

Third, as the errors that appear in code can manifest in dramatically varied forms, further research is required to assess whether the results seen here and elsewhere are the products of the unique qualities of the experimental code employed and their manipulations, or truly emblematic of broader effects of code manipulations on the efficient review and evaluation of code. Further, the specific types of errors that we introduced to our code are not wholly representative of all types of reputation and transparency errors. Future research would benefit from looking at many types of errors that were not included here. For example, code reputation could be explored with a variety of other manipulations including ratings and comments made by other programmers. The effect of comments made within the code would also be an interesting area for future investigation.

5.2. Theoretical and Practical Implications

Our findings of behavioral differences in eye movements indicate that there are differences in cognitive processing or visual effort between code that does and does not contain errors. Therefore, there is evidence that manipulating source, readability, and organization aspects of code have practical implications for recommendations towards writing reusable code. For instance, programmers should be careful to check for misuses of case and misuses of declarations in their code if they want others to be able to evaluate it for reuse in an efficient manner. The results suggest these forms of code errors specifically draw more attention relative to other types, which may reduce the ability to evaluate code quickly and effectively for reuse.

Second, we found theoretical support for the source and readability manipulations on time spent reviewing the code. This serves as both a recommendation and caution, as code whose source was designated as reputable evoked less visual effort when readability was high and low. While this can enhance efficiency with highly readable code, it may lead to oversight with code that is well regarded but less readable; as such, code may have other issues beyond readability alone. Third, we observed some of the highest FC in conditions that featured incongruent code manipulations (e.g., low readability with high organization). Code of inconsistent quality may cause programmers to devote excessive cognitive resources and time in evaluating its quality. Programmers interested in developing reusable code should ensure that their code is of high quality in all respects of readability and organization but should consider focusing on fixing instances of misuse of case and misuse of declarations in order to garner higher perceptions of readability and organization, respectively.

6. Conclusion

The growing prominence of code reuse across industries brings significant potential for both risk and reward. While the effective reuse of code can optimize development time and contribute to a higher quality final product, implementation of poor-quality code can result in security risks, lost time and revenue, and be detrimental to a project overall. This study leveraged eye-tracking data to examine how manipulations made to a Java code file's source reputation, readability, and organization affected program reviewers' visual effort as well as their perceptions of code trustworthiness and reuse potential. Our findings indicate that reviewers expended more visual effort (higher FC, higher TFD) on code whose source reputation was unknown compared to reputable code. More visual effort was expended on code that contained many organization errors versus code that was high in organization (marginally higher FC, higher TFD). Readability errors did not significantly affect reviewer's eye movement behaviors; however, errors introduced to the code's readability lowered participant's perceptions of trustworthiness and reuse intentions. The most fixated readability and organization error types were misuse of case and mises of declarations, respectively, relative to the rest of the Java file. The results found here are a crucial first step in understanding roles that each of the unique forms of code errors can play in the assessment and evaluation of code for reuse.

8. Acknowledgement

The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense, or of the United States Air Force. Distribution A. Approved for public release: AFRL-2023-4568.

7. References

Alarcon, G. M., & Ryan, T. J. (2018). Trustworthiness perceptions of computer code: A heuristic-systematic processing model. *Proceedings of the Hawaii International Conference on System Sciences*, *51*, 5384–5393.

Alarcon, G. M., Gamble, R., Jessup, S. A., Walter, C., Ryan, T. J., Wood, D. W., & Calhoun, C. S. (2017a). Application of the heuristic-systematic model to computer code trustworthiness: The influence of reputation and transparency. *Cogent Psychology*, *4*, Article 1389640.

Alarcon, G. M., Gibson, A. M., Jessup, S. A., Capiola, A., Raad, H., & Lee, M. A. (2020a). Effects of reputation, organization, and readability on trustworthiness perceptions of computer code. *Proceedings of the Human Computer Interaction International Conference*, *12183*, 367–381.

Alarcon, G. M., Gibson, A. M., Walter, C., Gamble, R. F., Ryan, T. J., Jessup, S. A., Boyd, B. E., & Capiola, A. (2020b). Trust perceptions of metadata in open-source software: The role of performance and reputation. *Systems*, *8*(3), 1–14.

Alarcon, G. M., Militello, L. G., Ryan, P., Jessup, S. A., Calhoun, C. S., & Lyons, J. B. (2017b). A descriptive model of computer code trustworthiness. *Journal of Cognitive Engineering and Decision Making*, *11*(2), 107–121.

Bertram, I., Hong, J., Huang, Y., Weimer, W., & Sharafi, Z. (2020). Trustworthiness perceptions in code review: An eye-tracking study. *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, *14*, 1-6.

Binkley, D., Davis, M., Lawrie, D., Maletic, J. I., Morrell, C., & Sharif, B. (2013). The impact of identifier style on effort and comprehension. *Empirical Software Engineering*, *18*(2), 219–276.

Busjahn, T., Schulte, C., & Busjahn, A. (2011). Analysis of code reading to gain more insight in program comprehension. *Proceedings of the Koli Calling International Conference on Computing Education Research*, 1–9.

Crosby, M. E., Scholtz, J., & Wiedenbeck, S. (2002). The roles beacons play in comprehension for novice and expert programmers. *Proceedings of the Workshop of the Psychology of Programming Interest Group*, *14*, 58–73.

De Oliveira, B., Ribeiro, M., Da Costa, J. A. S., Gheyi, R., Amaral, G., De Mello, R., Oliveira, A., Garcia, A., Bonifacio, R., & Fonseca, B. (2020). Atoms of confusion: The eyes do not lie. *Proceeding of the Brazilian Symposium on Software Engineering*, *34*, 243–252.

Eckstein, M. K., Guerra-Carrillo, B., Miller Singley, A. T., & Bunge, S. A. (2017). Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development? *Developmental Cognitive Neuroscience*, *25*, 69–91.

Frakes, W. B., & Kang, K. (2005). Software reuse research: Status and future. *IEEE Transactions on Software Engineering*, *31*(7), 529–536.

Hauser, F., Mottok, J., & Gruber, H. (2018). Eye tracking metrics in software engineering. *Proceedings of the European Conference of Software Engineering Education*, *3*, 39–44.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Weijer, J. (2011). *Eye tracking: A comprehensive guide to methods and measures*. Oxford.

Jessup, S. A., Willis, S. M., Alarcon, G. M., & Lee, M. A. (2021). Using eye-tracking data to compare differences in code comprehension and code perceptions between expert and novice programmers. *Proceedings of the Hawaii International Conference on System Sciences*, *54*, 114–123.

Katona, J. (2021). Clean and dirty code comprehension by eye-tracking based evaluation using gp3 eye tracker. *Acta Polytechnica Hungarica*, *18*(1), 79–99.

Liu, L., Liu, W., Li, X., Wang, W., & Cheng, W. (2020). Eye-tracking based performance analysis in error finding programming test. *Proceedings of the International Conference on Computer Science and Education*, *15*, 477–482.

Mäkitalo, N., Taivalsaari, A., Kiviluoto, A., Mikkonen, T., & Capilla, R. (2020). On opportunistic software reuse. *Computing*, *102*(11), 2385–2408.

Orquin, J. L., Ashby, N. J. S., & Clarke, A. D. F. (2016). Areas of interest as a signal detection problem in behavioral eye-tracking research. *Journal of Behavioral Decision Making*, *29*(2–3), 103–115.

Raney, G. E., Campbell, S. J., & Bovee, J. C. (2014). Using eye movements to evaluate the cognitive processes involved in text comprehension. *Journal of Visualized Experiments*, *83*, 1–7.

Ryan, T. J., Walter, C., Alarcon, G. M., Gamble, R. F., Jessup, S. A., & Capiola, A. (2019). The influence of personality on code reuse. *Proceedings of the Hawaii International Conference on System Sciences*, *52*, 5805–5814.

Sharafi, Z., Shaffer, T., Sharif, B., & Gueheneuc, Y. G. (2015). Eye-tracking metrics in software engineering. *Proceedings of the Asia-Pacific Software Engineering Conference*, *22*, 96–103.

Smilek, D., Birmingham, E., Cameron, D., Bischof, W., & Kingstone, A. (2006). Cognitive Ethology and exploring attention in real-world scenes. *Brain Research*, *1080*(1), 101–119.

Uzzaman, S., & Joordens, S. (2011). The eyes know what you are thinking: Eye movements as an objective measure of mind wandering. *Consciousness and Cognition*, *20*(4), 1882–1886.

Walter, C., Gamble, R. F., Alarcon, G. M., Jessup, S. A., & Calhoun, C. S. (2017). Developing a mechanism to study code trustworthiness. *Proceedings of the Hawaii International Conference on System Sciences*, *50*, 5817–5826.