Using Eye-Tracking Data to Compare Differences in Code Comprehension and Code Perceptions between Expert and Novice Programmers

Sarah A. Jessup
Air Force Research Laboratory
Wright Patterson AFB, OH
sarah.jessup.ctr@us.af.mil

Sasha M. Willis
General Dynamics Information Technology, Dayton, OH
sasha.willis@gdit.com

Gene M. Alarcon
Air Force Research Laboratory
Wright Patterson AFB, OH
gene.alarcon.1@us.af.mil

Michael A. Lee
General Dynamics Information Technology, Dayton, OH
michael.lee@gdit.com

Abstract

Previous research has examined how eye-tracking metrics can serve as a proxy for directly measuring the amount of cognitive effort and processing required for comprehending computer code. We conducted a pilot study comprising expert (n = 10) and novice (n = 10) computer programmers to examine group differences in code comprehension abilities and perceptions. Programmers were asked to read two pieces of computer code, rate the code on various attributes, and then describe what the code does. Results indicate that experts and novices significantly differ in terms of their fixation counts made during the task, such that experts had more fixations than novices. This was counter to our hypothesis that experts would have fewer fixations than novices. We found no evidence that experts and novices differed in their average fixation durations, trustworthiness and performance perceptions, or willingness to reuse the code.

1. Introduction

Understanding the cognitive processes involved in code comprehension, defined as "the process of understanding program code unfamiliar to the programmer" [1], can provide meaningful information about how users make decisions on whether to reuse code. Differences in code comprehension abilities and processing strategies between experienced versus novice programmers may underlie these decisions.

Physiological indices, such as eye-tracking data, can provide quantitative measurements of the decision-making process [2]. For example, information about where a user is looking during a code comprehension task can reveal what information users find important, and the amount of time needed to make decisions about the code. The amount of time needed for text comprehension can be approximated by measuring fixation durations within specified regions of interest (ROIs) [3, 4]. Additionally, the number of fixations made within ROIs can provide data showing the location of attention for different observers [5], as well as how efficiently different types of observers process that information [6, 7].

The purpose of this research is to investigate how code comprehension differs between expert and novice coders by measuring each group's fixation counts and average fixation durations within the code region using eye-tracking technology. Additionally, we investigate how these potential differences in eye movements may be related to programmers' willingness to reuse code, as well as how trustworthiness and performance perceptions of code differ with experience.

2. Related Work

2.1. Computer Code

Over the last few decades, there has been an emphasis placed on science, technology, engineering, and math (STEM) education [8]. The prevalence of STEM courses offered in education has led to an influx of graduates in fields such as computer and information sciences, which has almost doubled in the number of bachelor's degrees awarded since the beginning of the 21st century [9]. This large increase in programmers has resulted in an expansion of the amount of computer code that is being developed/written, shared, and re-used.

Code that is available through open-source libraries may potentially be used by thousands of people. As a result, code can be vetted, modified, and rated by other users.

There are several factors that influence how programmers perceive computer code. In a cognitive task analysis [10], researchers identified three main factors that influence programmers' reliance on, or trust in, code previously written by other programmers. Those factors are perceived code performance, transparency, and reputation. Several empirical studies have been conducted to examine how these factors influence programmers' trustworthiness perceptions (e.g., [11-13]). In addition to trust in the code, whether programmers choose to reuse code can indicate their understanding of that code. If programmers do not understand what a piece of code does, then they are less likely to repurpose it for their own needs [14]. As such, the willingness to reuse code can provide an approximation of code comprehension, provided the code compiles and is error-free.

2.2. Eye-Tracking

Researchers have used eye-tracking technology as means of studying the code comprehension strategies of programmers [15]. Eye-tracking technology provides researchers with a means of obtaining quantitative information about where people are looking within a visual scene and what information is processed by the observer [16]. Additionally, the amount of time an observer spends fixating on a stimulus is assumed to be proportional to the amount of time that is needed to process that information [5, 16]. In this way, eye-tracking data is used to gain insight into cognitive processes including, but not limited to, the user's allocation of attention, text comprehension, and problem-solving strategies [4, 5]. Importantly, these visual metrics can also reveal individual differences between people, such as prior knowledge of the material, reading goals, and processing efficiency [5, 17-19].

Research in eye-tracking literature has shown an inconsistent pattern of results specifically relating to the analysis of fixation data across levels of expertise. In a map visualization study, [18] found that the fixation counts of experts were greater than those of novices due to the expert group having shorter fixation durations, affording them more time to explore more areas of the image. However, [19] found the opposite pattern in a mathematical graph reading study in which experts had fewer, longer fixations, whereas novices exhibited more fixations with a shorter average duration. In this study, experts fixated for longer durations, on average, in regions containing important information than did novices; however, this difference was not significant when these durations were calculated as a percentage of total time on task, nor was the time difference significantly different between important and less important areas for experts versus novices. Additional studies have allowed researchers to investigate this mixed pattern of results.

Some studies point to experts having more *efficient information processing strategies* compared to novices [20], whereby experts not only had shorter fixation durations, but that they also appeared to attend more to task-relevant areas and less to task-redundant areas. Others [21] suggest that fewer fixations by novices indicate a decrease in *engagement* as compared to their more experienced partners. Still other studies point to differences in *visual effort* [22] as an explanation for experts having fewer fixations and shorter fixation durations than novices.

One possible explanation for why eye-tracking research regarding differences in expertise has shown mixed results and a variety of interpretations is because eye-movement behaviors may vary as a function of the task given to participants or the domain being studied. For this reason, using visual effort as an explanation for differences in eye movements within the domain of software engineering appears to be the most relevant to this research (see [15, 23-25]). In the current study, we add to the literature by examining eye-movement data alongside self-report measures of comprehension to better understand the relationship between fixations and expertise in software engineering.

2.3. Code Comprehension and Expertise

By definition, novices do not have as much experience, skill, or knowledge as compared to experts. As such, experts and novices differ in problem-solving techniques, comprehension, and ability [e.g., 26-29]. It is important to understand these differences and how they affect performance on tasks related to programming. For example, Soloway et al. [29] found that when expert and novice programmers were asked to write a line of code that was missing from a program, experts performed better and took less time completing the task, compared to novices. Similarly, Lee et al. [30] found that experts were more efficient and more accurate on a series of code comprehension tasks compared to novices.

People attend to and process visual information along two routes often referred to as top-down and bottom-up processing [31]. Top-down processing refers to the process of using schemas, or information such as the title of a program, to infer a general idea of how the code ought to function. Bottom-up processing, in this context, refers to reading sections of code line by line to gather information then chunking this information together with other parts of previously chucked information. Chunks are combined in an iterative manner to create a mental model and an understanding of the overall code piece or software [32]. Researchers have demonstrated that there are information-processing differences that change with experience. For instance, programmers with more knowledge of a program use top-down processing, while those with less knowledge or less familiarity with a program tend to use bottom-up processing [32-34]. Experts form better mental representations (e.g., pattern recognition, hierarchical structure, etc. [35]) and have developed schemas [29, 36] of computer code based on prior experience, which leads to greater comprehension when reading computer code. Novices tend to focus on concrete information available within the code such as *how* the program works, whereas experts focus on functional information that describes *what* the program does [28, 35]. Novices are not as proficient as experts in areas such as chunking information together or debugging and encoding strategies, and often demonstrate a lack of efficiency when writing and organizing lines of code compared to experts [37-39].

Code comprehension is particularly important because it can influence decision-making. In a study of student computer programmers, code comprehension influenced their decision to reuse code functions [14]. Results indicated that if students understood the code function at an abstract level rather than an algorithmic level, they chose to reuse a code function that was provided rather than re-write a new function. Novices may not be able to adapt code that they did not write to fit their current purpose; they may not make the connections between similar code examples and their own if they do not entirely know how the code functions. While empirical research has demonstrated code comprehension abilities differ with expertise, the reviewed research is not without limitations.

A recent literature review summarized research conducted using eye trackers in the field of software engineers [15]. None of the reviewed studies compared participants' self-report (subjective) data to their behavioral (objective) data, while also accounting for experience. While behavioral data is invaluable, self-reports allow researchers to understand programmers' perceptions of code, which eye-tracking data cannot directly measure. Another limitation concerns the length of computer code used as stimuli. Researchers often used smaller snippets of code (e.g., 30 lines [40]) that were presented on a single screen, without the ability to scroll through the code [15, 40]. When programmers read, write, or edit code, the programs they view often consist of hundreds or even thousands of lines of text, sometimes across multiple screens or windows. With the development of new eye-tracking technology, researchers are now able to capture eye-tracking data while users scroll through a web page or document, or when accessing multiple windows on a single screen [40], which older eye-tracking technologies are not able to capture. Studies can now be conducted on longer pieces of code, thus increasing the ecological validity of the results that are found. We utilize this advancement in the current study by incorporating multiple pieces of code, each spanning a few hundred lines of text.

2.4. Research Questions

Based on previous findings reported above, we explored whether there were differences in programmers' code comprehension abilities and perceptions of code, depending on their expertise (i.e., experts versus novices), when longer pieces of code are provided. There is research to support that fixation metrics (e.g., fixation count and average fixation duration) approximate visual effort [23-25]. However, because there are multiple, and sometimes contradictory, interpretations of eye movements across experts and novices, we measure and present fixation counts and average fixation durations rather than combining these two metrics into a single variable (i.e., visual effort). Two additional measures were used to determine code comprehension: the ability to accurately describe the code function, and the willingness to reuse code. Additionally, two questions were used to evaluate programmers' perceptions of code: 1) trustworthiness ratings and 2) performance ratings. More specifically, we have the following hypotheses and research questions:

Hypothesis 1: Compared to novices, experts will show more effective code comprehension evidenced by A) fewer fixation counts and shorter fixation durations, B) accurately describing the code functions more frequently, and C) intending to reuse the code pieces more often.

Research Question 1: Are there differences between experts' and novices' code reuse intentions, after controlling for fixation counts and average fixation durations?

Research Question 2: Are there differences between experts and novices on perception of A) code trustworthiness, and B) code performance when controlling for fixation counts and average fixation durations?

3. Method

3.1. Participants

A total of 36 participants were recruited for pilot data as part of a larger study. Novice programmers (*n* =

22) were recruited from a Midwestern college, and Expert programmers (*n* = 14) were recruited from local industry around the college. Requirements to participate were at least three years of programming experience and participants had to know Java well enough to read and understand Java code. In total, 16 cases were excluded from analysis due to poor data quality and/or lack of experience (less than three years), or if there was an average track loss of 15% or greater on any of the stimuli pages. The remaining 20 participants ranged from 20-48 years of age (*M* = 29.85, *SD* = 8.31). The average age of Novices was 24 years (*SD* = 3.00), while the average age of Experts was 36 years (*SD* = 7.63). Total years of programming experience of participants ranged from 4-20 years, (*M* = 7.25, *SD* = 4.22), 45% listed Java as their primary programming language, 90% were male, and 50% were students. Participants were recruited from flyers, email, and by word of mouth. Participants received compensation in the form of a $50 gift card. The study was overseen by the Air Force Research Laboratory institutional review board.

3.2. Task and Stimuli

Participants viewed two pieces of computer code as part of a code comprehension task. All participants viewed each piece of code in the same order. **Code 1** was a *default properties parser* (277 lines, 952×3070 pixels), while **Code 2** was an *encryptor* (264 lines, 952×2412 pixels). Both pieces of code were described as coming from a reputable source.

3.3. Eye-tracking Metrics

Based on previous literature utilizing eye trackers in software engineering research (for review see [15]), we have included two eye-tracking metrics that are commonly collected when participants read computer code: fixation count (FC) and average fixation duration (AFD). Gaze data were collected using a Smart Eye Aurora remote eye tracker, which uses infrared light to record where a participant is looking on the screen at a sampling rate of 60 Hz. Using a remote eye tracker as opposed to head-mounted eyewear allows for the researcher to study participants in a way that is similar to how users would naturally read code. The iMotions Screen-Based Eye Tracking Module was used to conduct a calibration procedure and collect recordings of gaze data during the data collection process. Offline, the iMotions software performed preliminary analyses including estimates of data quality (e.g., track loss) and markers for fixations made within each presented screen. The iMotions software defined fixations as the periods during which eye movements did not exceed 30 degrees per second (with an average tracking error of about 0.5 degrees [41]) for a minimum of 60 milliseconds [42]. The x,y screen coordinates of each fixation were calculated by averaging all gaze positions within a fixation.

3.3.1. Fixation Count. Fixation count (FC) was defined as the number of fixations made within the pixel range of code for each participant and for Code 1 and Code 2 separately.

3.3.2. Average Fixation Duration. Average fixation duration (AFD) was computed separately for each participant and for Code 1 and Code 2 separately by computing the average duration in milliseconds of each fixation spent within the pixel range of the code regions.

3.4. Self-Report Measures

3.4.1. Programming Experience. Participants were asked if they were a student or not. Those that answered "Yes, I am a student" were classified as Novice programmers. Participants that selected "No, I am not a student" were classified as Expert programmers. Novices had a range of 5-7 years of programming experience (*M* = 5.9), and Experts ranged from 4-20 years of experience (*M* = 8.6).

3.4.2. Code Description. At the end of each page containing code, participants were asked to describe what the code does with the following prompt, "To the best of your knowledge, please describe what this code does in the text box below."

3.4.3. Code Reuse. After viewing each code, participants were asked if they would reuse the code without changes using a single-item measure. Participants could reply with the binary responses "Use" or "Don't use."

3.4.4. Perceptions of Code. Participants were asked to answer the following questions about each code using a 7-point scale: "How trustworthy is the code?" (1 = Not at all trustworthy to 7 = Very trustworthy), and "How well do you think this code will perform?" (1 = Not at all well to 7 = Very well).

3.5. Procedure

After consenting to take part in the research, participants were seated approximately 70 cm from the screen on which they completed a 4-point calibration procedure using the iMotions Screen-Based Eye Tracking Module at a 1920x1080 screen resolution. Failure to reach an appropriate level of calibration resulted in dismissal from the study; otherwise, participants continued through the experiment by completing a demographics survey.

After the survey and prior to the task, participants were shown the task instructions. All comments had been removed from the code. All packages had been modified to remove original sources. All the code compiled and was error-free. After reading the instructions, participants saw the first piece of code and then evaluated the code using the ratings provided and wrote a brief description of the code's function. These evaluations were completed separately for each code. Only one code was viewed and evaluated at a time. After the task was completed, participants were debriefed, thanked for their participation, and compensated for their time.

4. Results

4.1. Code Comprehension (H1)

4.1.1. Fixation Measurements (H1:A). The number of fixations (FC) and their average durations (AFD) collected for each Code may have varied by the function of the code, text length, etc. Because of these differences across Code 1 and Code 2, we decided to conduct separate analyses for each Code. A one-way mixed-design multivariate analysis of variance (MANOVA) was conducted for both Code 1 and Code 2 to determine the relationship between Expertise (Experts versus Novices) and FC and AFD. We analyzed the data against a null hypothesis that no significant differences exist between Experts and Novices regarding their eye-movement data collected during the task.

The results of the MANOVAs revealed that Expertise had a significant main effect on fixation measurements for Code 1, [$F(2, 17) = 6.51$, $p = .008$, $\eta_p^2 = .43$, power = .85], and for Code 2, [$F(2, 17) = 11.03$, $p < .001$, $\eta_p^2 = .57$, power = .98]. Univariate ANOVAs were conducted for Code 1 and Code 2 to determine the simple effects of Expertise for FC and AFD. Means and standard errors are listed in Table 1. There was a significant difference between Novices and Experts for FC on Code 1 [$F(1, 18) = 13.37$, $p = .002$, $\eta_p^2 = .43$, power = .93], and Code 2 [$F(1, 18) = 23.35$, $p < .001$, $\eta_p^2 = .56$, power > .99]. See Figure 1. No significant differences were found in AFD for either Code 1 or Code 2. See Figure 2.

Although the MANOVA results were significant, they were in the opposite direction hypothesized. Experts had more fixations for both Code 1 and Code 2 compared to Novices, contrary to our hypothesis that Experts would have fewer fixations compared to Novices. Thus, Hypothesis 1:A was not supported.

In general, participants fixated longer and on more aspects of Code 1 compared to Code 2. One reason for this may have been because Code 1 was the first code introduced during the task. Participants may have taken longer examining the code and fixated more as they were not only figuring out what the code does, but also discovering what information they had to glean from the code as indicated by the self-report responses. That is, participants were getting used to the task. For Code 2, participants were presumed to be more familiar and proficient with the study task.

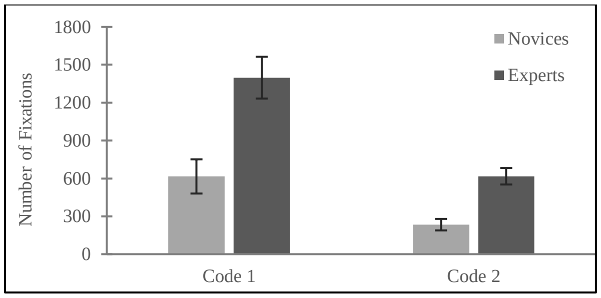

Figure 1. Number of fixations (FC) users had on Code 1 and Code 2. Error bars represent standard errors.

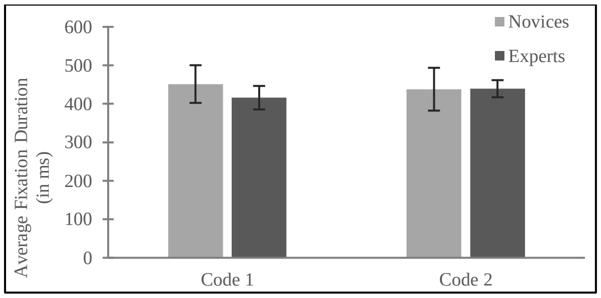

Figure 2. Average amount of time (milliseconds) of users' fixations (AFD) on Code 1 and Code 2. Error bars represent standard errors.

4.1.2. Code Description (H1:B). To test whether Experts accurately described the code functions more often than Novices, as indicated by their answers of code descriptions, a Fisher's Exact Test was calculated. Each of the programmer's answers to the question, "Describe what this code does" was screened for accuracy. For Code 1, all Experts correctly described what the code did, and 6 of the Novices were correct, while 4 Novice programmers were incorrect in describing the code's functionality. The difference between the code description accuracy of Experts and Novices was not significant. For Code 2, all programmers correctly described the code's function, regardless of Expertise. Hypothesis 1:B was not supported.

Table 1. Means and standard errors of fixation measurements and self-reports for Experts and Novices.

	Code 1		Code 2	
	Novices	**Experts**	**Novices**	**Experts**
Fixation Measurements				
Fixation Count	616.00 (135.43)	1397.90 (165.46)	233.80 (45.41)	617.20 (65.06)
Average Fixation Duration (in ms)	451.37 (48.97)	416.01 (30.49)	437.99 (55.66)	439.31 (22.25)
Self-Reports				
Trustworthiness Perceptions	5.00 (0.45)	4.40 (0.37)	5.70 (0.26)	4.40 (0.62)
Performance Perceptions	5.30 (0.30)	4.40 (0.43)	5.90 (0.35)	5.30 (0.42)
Code Reuse Intentions - Use	7	5	9	6
Code Reuse Intentions - Don't Use	3	5	1	4

Note. Standard errors in parentheses. Trustworthiness and Performance items were measured on a 7-point scale. Code reuse intentions are reported as total number of participants that chose to either Use or Don't Use the code.

4.1.3. Code Reuse (H1:C and RQ1). A Generalized Estimating Equation (GEE) analysis was conducted for each Code to evaluate differences in Code Reuse intentions between Experts and Novices. This analysis was chosen due to the binary nature of the dependent variable. Results indicated that neither Expertise nor the intercept was significant for Code 1. The intercept was significant for Code 2, [Wald $\chi 2$ (1, $N = 20$) = 4.35, β = -2.20, p = .037], though Expertise was not, indicating that factors other than programmer experience significantly contribute to Code Reuse intentions. Although not significantly different, an inspection of the means revealed Experts appeared more willing to reuse Code 1 and Code 2 than Novices (see Table 1).

Because the intercept in the above analysis was significant, we had justification for examining if fixation measurements contributed to the variance in Code Reuse that was not accounted for by Expertise. Separate GEE analyses were conducted for Code 1 and Code 2, which included Expertise, FC, and AFD (standardized for ease of interpretation), with Reuse intentions as the outcome variable. The interaction between Expertise and AFD was found to significantly contribute to the variance in Code Reuse intentions for both Code 1 [Wald χ^2 (1, $N = 20$) = 16.47, β = -5.79, p < .001], and for Code 2 [Wald $\chi 2$ (1, $N = 20$) = 4.44, β = -4.94, p = .035].

This interaction revealed that Novices (Code 1: M = 426.71, SE = 22.59; Code 2: M = 440, SE = 58.99) had shorter AFDs than Experts (Code 1: M = 482.46, SE = 20.54; Code 2: M = 475.88, SE = 20.74) when they had the intention to Reuse Code, but longer AFDs (Code 1: M = 508.92, SE = 92.48; Code 2: M = 419.66, SE = 0.00) than Experts (Code 1: M = 349.57, SE = 23.80; Code 2: M = 384.44, SE = 9.96) when they did not intend to Reuse Code (see Figures 3 and 4). The standard errors for this data should be interpreted with caution because the sample of participants who were both Novices and did not intend to Reuse Code was so small (see Table 1). The intercept was also significant for the model for Code 2 [Wald $\chi 2$ (1, $N = 20$) = 10.13, β = -3.91, p = .001], suggesting that other variables not accounted for in the model influence the relationship between Expertise and Reuse intentions.

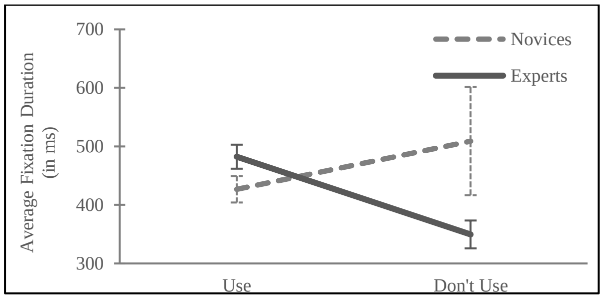

Figure 3. Interaction between Expertise and average amount of time (milliseconds) of users' fixations (AFD) on Code 1.

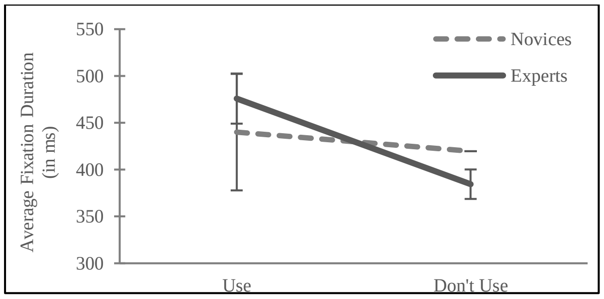

Figure 4. Interaction between Expertise and average amount of time (milliseconds) of users' fixations (AFD) on Code 2.

4.2. Code Perceptions (RQ2)

4.2.1. Perceived Trustworthiness (RQ2:A). We ran a one-way analysis of covariance (ANCOVA) on each

Code to explore if there was a relationship between Expertise and perceptions of Code Trustworthiness, while controlling for FC and AFD. There were no significant results for either Code 1 or Code 2 (see Figure 5).

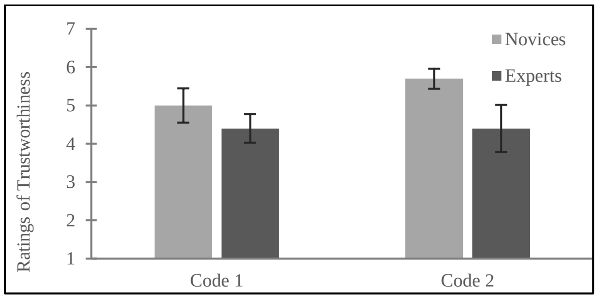

Figure 5. User ratings of Trustworthiness perceptions of Code 1 and Code 2. Error bars represent standard errors.

4.2.2. Perceived Performance (RQ2:B). Separate one-way ANCOVAs were also conducted for each Code to examine the relationship between Expertise and perceptions of Code Performance, when controlling for FC and AFD. For Code 1, there was a significant main effect of Expertise, [$F(1, 14) = 9.18$, $p = .009$, $\eta_p^2 = .40$, power = .80], on perceptions of Code Performance after controlling for FC and AFD. On average, Novices perceived the Code Performance as higher than Experts (see Figure 6). For Code 2, all results were non-significant.

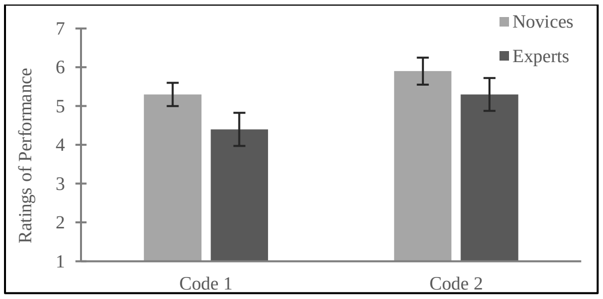

Figure 6. User ratings of Performance perceptions of Code 1 and Code 2. Error bars represent standard errors.

5. Discussion

This paper explored how code comprehension—measured by eye-tracking metrics, accuracy of code descriptions, and reuse intentions—and programmer perceptions of code trustworthiness and performance differed between expert and novice programmers. With regard to code comprehension, group differences were only observed with eye-tracking metrics. By measuring fixation counts and average fixation durations for both novices and experts across Code 1 and Code 2, we found evidence that there are differences between groups, such that experts had more fixations compared to novices. However, average fixation duration did not differ between experts and novices. We also explored how programmers' perceptions of the code pieces differed between experts and novices, after controlling for fixation counts and average fixation durations. While many of our analyses lacked the statistical power necessary to draw conclusive inferences, we found a significant interaction between expertise and average fixation duration on intention to reuse code.

While the analyses for Hypothesis 1:A did reveal statistically significant differences in fixation counts between experts and novices, these results were in the opposite direction that we predicted. Based on existing literature on code comprehension, we hypothesized that experts would have fewer fixation counts compared to novices. In this study, experts had higher fixation counts on Code 1 and Code 2 compared to novices. Past neuroscience research may shed some light on these results. In an electroencephalogram (EEG) study, Lee et al. [30] found that expert programmers, compared to novices, showed greater beta and gamma wave activation while performing comprehension tasks. The authors interpreted these findings as indicating that experts were devoting more concentration toward, and utilizing more cognitive skills in, the tasks. Similarly, eye-tracking metrics provide insight into ongoing cognitive processes with longer fixation durations and higher fixation counts indicating more complex processing [16], which may indicate that the experts in the present study were engaging in more complex processing than the novices.

Theeuwes and Belopolsky [43] explain that rewarding stimuli will draw more fixations to their locations than stimuli that are not associated with a reward. In the context of this research, certain functions or subsections of the code may have been perceived as rewarding or relevant to experts who would know how to apply those functions to the answers in their descriptions of the purpose of the code. Theeuwes and Belopolsky also note that the rewarding stimuli do not hold attention at those locations for longer durations of time than other aspects of the environment, which could explain why AFD was not significantly different between experts and novices. Future examination of this data could explore which subsections of code drew relatively more fixations to help clarify the reason why experts made, on average, more fixations than novices. In this study, we defined the region of interest as the global piece of code, but further examination into which area of the code drew more fixations would provide greater insight into the different processing strategies between groups. An analysis of scan path data may show that the way readers navigate through code can differentiate between experts and novices.

Although the results for code reuse were not statistically different between experts and novices, an examination of the means showed that overall, experts chose to use Code 1 and Code 2 less often than novices. These differences may become significant with a larger sample of participants. Because the code stimuli that were included in our task were error-free and compiled, both novices and experts could have intended to reuse the code without the need to check for syntax errors. It's possible that novices would elect to reuse code more often than experts due to their relative inability to create new code from scratch.

Even though we found no statistical differences between experts' and novices' reuse intentions, the intercept in the original GEE model was significant, which indicated that other variables significantly contributed to the variance in reuse intentions. We added fixation measurements to the model and found a significant interaction between expertise and average fixation durations. For both Code 1 and Code 2, novices had shorter fixations than experts when they intended to reuse the code but longer fixations than experts when they did not intend to reuse the code. This might have been the case because once novices indicated they intended to reuse the code, they did not need to gather as much evidence to support this decision. Experts, on the other hand, may have continued to evaluate their decision while reading through the code, such that they may have spent more time reading each line to ensure that the code could be reused. When novices did not intend to reuse the code, they spent more time on each fixation possibly because they were figuring out if they knew enough about the code's functions that they could modify it appropriately for a future purpose. In contrast to this, experts could quickly decide that they would not reuse the code after finding a section of code that did not align with their mental model of how the code should be written. Once this decision was made, they would only need to gather as much detail from each fixation as would be needed to report the code's overall function for the final code description question.

It is important to note that the accuracy of participants' code descriptions did not significantly differ with expertise. There are two explanations as to why this occurred. First, we measured expertise by whether participants indicated they were a student. Some participants that were students had more years of experience coding than programmers who were not students, and vice versa. However, exploratory analyses using years of experience, as well as age, in place of student status did not change the results of our analyses, and thus were not reported, providing support for our chosen expertise classifier. Second, at the top of every piece of code there was a line that stated what the code was used for (e.g., "public abstract class BasicAnnotationProcessor"), which may have helped guide the responses that participants gave in their code descriptions. Future analyses of differences between experts and novices may benefit from not including this preliminary description of the code and also removing cases for which the user was not able to accurately describe the code's function or purpose.

This research was conducted on pilot data that included a small sample size of programmers. Although we had some statistically significant results and data trending towards significant differences, we had low power for many of our analyses, which indicates that we need to continue collecting more data in order to obtain results that can be interpreted with confidence.

5.1. Implications

When it comes to integrating eye-tracking technology into applied research, the stimuli that comply with the allowances of the equipment can be seen as a limitation to researchers. That is, there may be the perception that eye-tracking integration requires images used for visual stimuli to be contained within a single screen length. However, the code stimuli that were used in this research were quite long and extended several screen lengths. Participants needed to scroll through the code in order to comprehend the piece in its entirety and answer the questions that followed. Our study adds to the existing literature of eye-movement behavior during computer code comprehension by including these longer pieces of code and scrolling behavior.

The combination of both behavioral (eye-tracking) and subjective (self-reports) measures of code comprehension are similarly lacking from the existing literature, although studies combining the two facets are beginning to emerge (e.g., [44]). Our study integrates these two aspects, providing a more complete picture of the factors that influence code comprehension. While not included in this research, future directions for this comparison might include directly comparing functionally similar pieces of source code with various changes to other code aspects (e.g., readability, organization). This analysis could reveal other factors that influence code reuse, such as what types of code are easier for novices to adopt, are more trustworthy, perform better, etc.

5.2. Conclusion

In summary, analyzing eye-tracking data does show that there are meaningful differences in the eye movements of experts versus novices during a code comprehension task. These results could be interpreted in different ways. There is a growing need for eye-tracking research in this area. As indicated by our results, there are alternative interpretations from what would be posited by some of the existing literature. Principally, there are multiple avenues for including eye-tracking research in code comprehension tasks that have not yet been explored, which can help explain or uncover critical differences between levels of expertise.

6. Acknowledgement

This material is based upon work supported by U.S. Air Force Research Laboratory contracts FA8650-16-D-6616, 0003. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the U.S. Air Force. DISTRIBUTION STATEMENT A. Approved for public release: 88ABW Cleared 07/10/19; 88ABW-2020-2192.

This research was also supported, in part, by an appointment to the Postgraduate Research Participant Program at the U.S. Air Force Research Laboratory administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and USAFRL.

7. References

[1] J. Koenemann and S.P. Robertson, “Expert problem-solving strategies for program comprehension”, In Proceedings of the Conference on Human Factors in Computing Systems (CHI), Association for Computing Machinery, United States, 1991, pp. 125-130.

[2] S. Kurimori and T. Kakizaki, “Evaluation of Work Stress Using Psychological and Physiological Measures of Mental Activity in a Paced Calculating Task”, Industrial Health, National Institute of Occupational Safety and Health, Japan, 1995, pp. 7-22.

[3] W.C. Li, J.J. Lin, G. Braithwaite, and M. Greaves, “The Development of Eye Tracking in Aviation (HP2) Technique to Investigate Pilot's Cognitive Processes of Attention and Decision-Making”, In Proceedings of the Conference of the European Association for Aviation Psychology, Hogrefe Publishing Group, Portugal, 2016, pp. 26-30.

[4] M.K. Eckstein, B. Guerra-Carrillo, A.T.M. Singley, and S.A. Bunge, “Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development?”, Developmental Cognitive Neuroscience, Elsevier, Netherlands, 2017, pp. 69-91.

[5] G.E. Raney, S.J. Campbell, and J.C. Bovee, “Using Eye Movements to Evaluate the Cognitive Processes Involved in Text Comprehension”, Journal of Visualized Experiments, JoVE, United States, 2014, Article e50780.

[6] N.P. Murray and C.M. Janelle, “Anxiety and performance: A visual search examination of the processing efficiency theory”, Journal of Sport and Exercise Psychology, Human Kinetics Journals, United States, 2003, pp. 171-187.

[7] M.G. Calvo, P. Avero, and D. Lundqvist, “Facilitated detection of angry faces: Initial orienting and processing efficiency”, Cognition and Emotion, Taylor & Francis, United Kingdom, 2006, pp. 785-811.

[8] N.K. Dejarnette, “America’s Children: Providing Early Exposure to STEM (Science, Technology, Engineering and Math) Initiatives”, Education, Project Innovation, United States, 2012, pp. 77-84.

[9] T.D. Snyder, C. de Brey, and S.A. Dillow, “Digest of Education Statistics 2018”, National Center for Education Statistics, United States, 2019, retrieved from https://nces.ed.gov/programs/digest/d19/tables/dt19_322.10.asp

[10] G.M. Alarcon, L.G. Militello, P. Ryan, S.A. Jessup, C.S. Calhoun, and J.B. Lyons, “A Descriptive Model of Computer Code Trustworthiness”, Journal of Cognitive Engineering and Decision Making, SAGE Publications, United States, 2016, pp. 107-121.

[11] G.M. Alarcon, A.M. Gibson, C. Walter, R.F. Gamble, T.J. Ryan, S.A. Jessup, B.E. Boyd, and A. Capiola, “Trust Perceptions of Metadata in Open-Source Software: The Role of Performance and Reputation”, Systems, MDPI, Switzerland, 2020, Article 28.

[12] G.M. Alarcon, A.M. Gibson, S.A. Jessup, A. Capiola, H. Raad, and A.M. Lee, “Effects of Reputation, Organization, and Readability on Trustworthiness Perceptions of Computer Code”, In Proceedings of the Conference on Human-Computer Interaction (HCI), Springer, Germany, 2020, pp. 367-381.

[13] G.M. Alarcon, R.F. Gamble, S.A. Jessup, C. Walters, T.J. Ryan, D.W. Wood, and C.C. Calhoun, “Application of the Heuristic-Systematic Model to Computer Code Trustworthiness: The Influence of Reputation and Transparency”, Cogent Psychology, Taylor & Francis, United States, 2017, Article 1389640.

[14] C.M. Hoadley, M.C., Linn, L.M. Mann and M.J. Clancy, “When, Why and How Do Novice Programmers Reuse Code”, W. Gray and D. Boehm-Davis (Eds.), Empirical Studies of Programmers: Sixth Workshop, Ablex Publishing, United States, 1996, pp. 109-130.

[15] Z. Sharafi, Z. Soh, and Y.G. Guéhéneuc, “A Systematic Literature Review on the Usage of Eye-Tracking in Software Engineering”, Information and Software Technology, Elsevier, Netherlands, 2015, pp. 79-107.

[16] M.A. Just and P.A Carpenter, “The Role of Eye-Fixation Research in Cognitive Psychology”, Behavior Research Methods & Instrumentation, Springer, Germany, 1976, pp. 139-143.

[17] J.K. Kaakinen and J. Hyönä, “Perspective Effects in Repeated Reading: An Eye Movement Study”, Memory & Cognition, Springer, Germany, 2007, pp. 1323-1336.

[18] K. Ooms, P. De Maeyer, and V. Fack, “Study of the attentive behavior of novice and expert map users using eye tracking”, Cartography and Geographic Information Science, Taylor & Francis, United States, 2014, pp. 37-54.

[19] V. Embse, “An eye fixation study of time factors comparing experts and novices when reading and interpreting mathematical graphs”, Doctoral dissertation, The Ohio State University, United States, 1987, pp. 1-158.

[20] A. Gegenfurtner, E. Lehtinen, and R. Säljö, “Expertise differences in the comprehension of visualizations: A meta-analysis of eye-tracking research in professional domains”, Educational Psychology Review, Springer, Germany, 2011, pp. 523-552.

[21] M. Villamor, M. Rodrigo, and T. Mercedes, “Characterizing Individual Gaze Patterns of Pair Programming Participants”, Proceedings of the 26th International Conference on Computers in Education (ICCE 2018), APSCE, Taiwan, 2018, pp. 193-198.

[22] R. Turner, M. Falcone, B. Sharif, and A. Lazar, “An eye-tracking study assessing the comprehension of C++ and Python source code”, Proceedings of the Symposium on Eye Tracking Research and Applications, ACM, United States, 2014, pp. 231-234.

[23] D. Binkley, M. Davis, D. Lawrie, J. Maletic, C. Morrell, and B. Sharif, “The impact of identifier style on effort and comprehension”, Empirical Software Engineering, Springer, Germany, 2013, pp. 219-276.

[24] B. Sharif and J. Maletic, “An eye tracking study on camelcase and under_score identifier styles”, 2010 IEEE 18th International Conference on Program Comprehension, Wiley-IEEE Press, Netherlands, 2010, pp. 196-205.

[25] N. Ali, Z. Sharafi, Y.G. Guéhéneuc, and G. Antoniol, “An empirical study on the importance of source code entities for requirements traceability”, Empirical Software Engineering, Springer, Germany, 2015, pp. 442-478.

[26] W.G. Chase and H.A. Simon, “Perception in Chess”, Cognitive Psychology, Elsevier, Netherlands, 1973, pp. 55-51.

[27] K.A. Ericsson and J. Smith, “Toward a General Theory of Expertise: Prospects and Limits”, Cambridge University Press, United Kingdom, 1991.

[28] B. Adelson, “When Novices Surpass Experts: The Difficulty of a Task Increases with Expertise”, Journal of Experimental Psychology: Learning, Memory, and Cognition, APA, United States, 1984, pp. 483-495.

[29] E. Soloway, E. Adelson, and K. Ehrlich, “Knowledge and Processes in the Comprehension of Computer Programs”, M.T.H. Chi, R. Glaser, and M.J. Farr (Eds.), The Nature of Expertise, Lawrence Erlbaum Associates, United States, 1988.

[30] S. Lee, A. Matteson, D. Hooshyar, S. Kim, J. Jung, G. Nam, and H.S. Lim, “Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis”, In Proceedings of the International Conference on Bioinformatics and Bioengineering (BIBE), Wiley-IEEE Press, Netherlands, 2016, pp. 350-355.

[31] M. Corbetta and G.L. Shulman, “Control of goal-directed and stimulus-driven attention in the brain”, Nature Neuroscience, Springer, Germany, 2002, pp. 201-215.

[32] C.L. Corritore and S. Wiedenbeck, “An Exploratory Study of Program Comprehension Strategies of Procedural and Object-Oriented Programmers”, International Journal of Human-Computer Studies, Elsevier, Netherlands, 2001, pp. 1-23.

[33] A.J. Ko and B. Uttl, "Individual Differences in Program Comprehension Strategies in Unfamiliar Programming Systems", In Proceedings of the International Workshop on Program Comprehension (ICPC), Wiley-IEEE Press, Netherlands, 2003, pp. 175-184.

[34] A. von Mayrhauser, and A.M. Vans, “Program Understanding During Software Adaptation Tasks”, In Proceedings of International Conference on Software Maintenance (ICSME), Wiley-IEEE Press, Netherlands, 1998, pp. 316-325.

[35] V. Fix, S. Wiedbeck, and J. Scholtz, “Mental Representations of Programs by Novices and Experts”, In Proceedings of the Conference on Human Factors in Computing Systems (CHI), Association for Computing Machinery, United States, 1993, pp. 74-79.

[36] A.C. Graesser, Prose Comprehension: Beyond the Word, Springer-Verlag, United States, 1981.

[37] W. Barfield, “Skilled Performance on Software as a Function of Domain Expertise and Program Organization”, Perceptual and Motor Skills, SAGE Publications, United States, 1997, pp. 1471-1480.

[38] I. Vessey, “Expertise in Debugging Computer Programs: A Process Analysis”, International Journal of Man-Machine Studies, Academic Press, United States, 1985, pp. 459-494.

[39] N. Ye and G. Salvendy, “Quantitative and Qualitative Differences between Experts and Novices in Chunking Computer Software Knowledge”, International Journal of Human-Computer Interaction, Taylor & Francis, United Kingdom, 1994, pp. 105-118.

[40] B. Sharif, and J.I. Maletic, “iTrace: Overcoming the Limitations of Short Code Examples in Eye Tracking Experiments”, In Proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME), Wiley-IEEE Press, Netherlands, 2016, pp. 647.

[41] G. Funke, E. Greenlee, M. Carter, A. Dukes, R. Brown, and L. Menke, “Which eye tracker is right for your research? Performance evaluation of several cost variant eye trackers”, In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, SAGE Publications, United States, 2016, pp. 1240-1244.

[42] “I-VT Fixation Filter: Configurations in iMotions”, iMotions A/S, Denmark, n.d., retrieved from https://help.imotions.com/hc/en-us/articles/207696189-I-VT-Fixation-Filter-Configurations-in-iMotions.

[43] J. Theeuwes and A.V. Belopolsky, “Reward Grabs the Eye: Oculomotor Capture by Rewarding Stimuli”, Vision Research, Elsevier, Netherlands, 2012, pp. 80-85.

[44] S. Papavlasopoulou, K. Sharma, and M.N. Giannakos, “How do You Feel about Learning to Code? Investigating the Effect of Children’s Attitudes Towards Coding Using Eye-Tracking”, International Journal of Child-Computer Interaction, Elsevier, Netherlands, 2018, pp. 50-60.