Studying Developer Eye Movements to Measure Cognitive Workload and Visual Effort for Expertise Assessment

SALWA D. ALJEHANE, University of Tabuk, Kingdom of Saudi Arabia
BONITA SHARIF, University of Nebraska - Lincoln, USA
JONATHAN I. MALETIC, Kent State University, USA

Eye movement data provides valuable insights that help test hypotheses about a software developer's comprehension process. The pupillary response is successfully used to assess mental processing effort and attentional focus. Relatively little is known about the impact of expertise level in cognitive effort during programming tasks. This paper presents a quantitative analysis that compares the eye movements of 207 experts and novices collected while solving program comprehension tasks. The goal is to examine changes of developers' eye movement metrics in accordance with their expertise. The results indicate significant increase in pupil size with the novice group compared to the experts, explaining higher cognitive effort for novices. Novices also tend to have a significant number of fixations and higher gaze time compared to experts when they comprehend code. Moreover, a correlation study found that programming experience is still a powerful indicator when explaining expertise in this eye-tracking dataset among other expertise variables.

CCS Concepts: • **Human-centered computing → Empirical studies in HCI**; • **Software and its engineering → General programming languages**;

Additional Key Words and Phrases: Reading analysis, Eye movements and cognition, Pupil dynamics, Eye tracking, Source code reading, Expertise, Cognitive effort

ACM Reference Format:
Salwa D. Aljehane, Bonita Sharif, and Jonathan I. Maletic. 2023. Studying Developer Eye Movements to Measure Cognitive Workload and Visual Effort for Expertise Assessment. *Proc. ACM Hum.-Comput. Interact.* 7, ETRA, Article 166 (May 2023), 18 pages. https://doi.org/10.1145/3591135

1 INTRODUCTION

Different programming expertise levels require a different mental workload to solve a comprehension task [45]. Program comprehension is a cognitive process that allows developers to use their knowledge along with a mental model to acquire information from the code and draw their conclusions [23]. There are also a number of studies that show reading and comprehending source code is very different from reading and understanding natural language prose [13, 22]. Thus, studying how experts and novices differ in reading code is important since the results from studies on reading natural language texts do not apply well. Over the years, researchers have made many attempts to measure programming expertise and try to evaluate programmers' activities with respect to their expertise [6, 23]. Additionally, studies attempt to understand developers' behaviors during programming activities corresponding to their expertise level, such as in source code

Authors' addresses: Salwa D. Aljehane, University of Tabuk, Department of Computer Science, Tabuk, Kingdom of Saudi Arabia, saljehani@ut.edu.sa; Bonita Sharif, University of Nebraska - Lincoln, School of Computing, Lincoln, Nebraska, USA, 68588, bsharif@unl.edu; Jonathan I. Maletic, Kent State University, Department of Computer Science, Kent, Ohio, USA, 44240, jmaletic@kent.edu.

2573-0142/2023/5-ART166 $15.00
https://doi.org/10.1145/3591135

comprehension [12, 24, 62], maintenance [67], and debugging [5, 66]. Previous studies measure the expertise in fairly straightforward ways, including questionnaires [24], years of programming experience, education level [1, 13, 41, 50], and how programmers self-evaluate their expertise [23]. Other studies use eye-tracking technologies to study the effect of expertise levels on comprehension during code reading [15], reviewing [65], summarization [51], and differences between the expertise and professional status of software developers in class diagram comprehension [61].

In this paper we are particularly interested in assessing the expertise level of software developers in the workplace along with students studying and learning to program. This is a difficult problem for a number of reasons. Factors such as new programming languages [59] (each suited for specific applications/domains), the need of being up to date with current (and ever-changing) technologies, and the availability and expansion of learning resources all play a part in developer expertise. There is a need to redefine expertise beyond using professional status, education level, or years of programming experience. Moreover, with improvements in the field of software engineering, there needs to be a consistent evolution in developer expertise assessment. One example of this is adopting more holistic methods for developer expertise evaluation in a realistic development environment while conducting programming activities (e.g., eye movement tracking of code reading). We do not claim that eye movements are the only measure. However, using eye movements in addition to other measures should provide valuable insight to improve processes and tools.

The objective of this study is to provide a reliable approach to characterize developers' expertise level as expert/novice via studying their eye movement data. Our intention is that results from such studies will provide a means to develop automated approaches to assessing expertise level based on eye movements. This study also aims to make use of the eye movement metrics to uncover distinct patterns that can differentiate between developers in a program comprehension task related to their expertise. We see this as the first step in further refining measures leading up to expertise prediction. Here we are specifically interested in pupillometry, the measurement of fluctuations in pupil diameter in response to a given stimulus (i.e., reading source code). In terms of analyzing pupillometry data before comparing subjects, we need to define pupil peak values. One can then identify the changes in the pupil sizes relative to the baseline for each developer up to each threshold. According to the Beatty reviews in 1982 [8], Hakerem and Sutton provide one of the first attempts of pupillometric analyses at the visual threshold [30]. We adopt this approach, which is presented and used previously in multiple studies to analyze pupil dilation [21, 25, 42].

Pupil size is influenced by cognitive load and tends to dilate up to 0.5 mm above its relative baseline value [8, 9, 60]. Many early studies have shown the effects of cognitive workload effect on pupil size [31, 32, 47]. Moreover, pupil dilation is used in many studies as a measure to study mental workload [7, 43], to explore the relationship between cognitive ability and the pupil baseline [64] and to combine pupil dilation with other physiological measurements to assess cognitive load [33].

This work uses the changes in the pupillary response of expert/novice developers as an indicator of the underlying cognitive efforts the subject performs. In order to provide a valid dilation analysis and to perform a fair comparison between developers, this study includes developers who solve tasks in the same language (Java). The results show that less experienced developers have statistically higher average fixations of dilated pupils than skilled developers. This suggests that novices apply more attentional focus and mental effort to solve the comprehension task compared to expert developers. The goal of the paper is to explore how years of programming experience influence the visual behavior of developers during comprehension. To achieve this goal, this paper seeks to address three research questions:

- RQ1: What is the best representative measurement for estimating expertise that shows a best connection with eye-tracking metrics?

- RQ2: Using eye movement measurements: fixation counts, total fixation duration, lines of code read, saccade length and saccade duration, to what degree do experts differ from novices?
- RQ3: To determine the differences between the cognitive load effort of experts and novices, can a developer's pupillary response contribute to assessing expertise?

This paper makes the following fundamental scientific contributions: 1) This is the first study to assess the differences between software developer cognitive workload using a pupillary response analysis in the context of expertise; 2) We study the relationship between multiple types of developer expertise metrics and their eye movement metrics; 3) The work provides evidence that the changes in the eye movement metrics are influenced by the developer's expertise level (novices/experts); and 4) We identify multiple eye movement-based metrics extracted from the largest publicly available eye tracking dataset (EMIP) [11] of 207 developers. This is the largest publicly available eye tracking data set of software comprehension tasks collected by a multi-institutional team, available to date. The results of this study support previous work on this topic however, give far greater credence as we examine unstudied measures and utilize a much larger data set. This provides much stronger evidence that a model for (accurate) automated expertise assessment can be constructed.

2 BACKGROUND AND RELATED WORK

We now present related work in pupil dilation and its connection to cognitive load followed by related work comparing experts and novices in program comprehension (a subfield of software engineering).

2.1 Pupil Dilation and Cognitive Load

In several studies researchers have used eye movement metrics to gain deep insights and to understand the ongoing cognitive process while solving programming problems. For these studies, pupil size is the most commonly used metric to assess the mental workload [8, 32, 37]. In one of the early research studies, which was released in 1982, Beatty conducted an empirical study reviewing multiple datasets from different domains [8]. They conclude that pupil dilation is a valid indicator of the cognitive workload that was applied when performing a mental process. In comparison, Klingner proposed a new method to study the quick cognitive load changes by aligning the gaze events with pupil dilation [42]. This method allowed for an analysis of participants' pupillary responses in a short time when understanding the visual tasks. This alignment has also successfully captured the variation that happens in pupil diameter sizes while subjects are changing between multiple types of tasks. Fritz et al. performed an empirical experiment to capture the cognitive process while working with difficult/easy tasks [25] and use multiple types of psycho-physiological measurements. They find that pupil dilation and blink rates are two valuable metrics that respond relative to working with difficult tasks.

Further research has found the same connection and concluded that there are many advantages of using pupil response as a cognitive workload measurement during interactive tasks [37], driving [48], in real-time during web browsing [39], and also using pupil dilation with blinking rate to study the performed mental efforts [34, 35, 63]. Additionally, we believe that studying the changes of developers' pupillary response while solving comprehension problems with considering developers' expertise level would provide a better understanding of expert/novice differences in cognitive processing. Thus, our study mainly relies on the analysis of individual pupil size changes as an interpretation of the mental workload that occurs during comprehension tasks. Kontogiorgos and Manikas propose a theoretical idea about how expertise level could influence the programmers' mental effort [44]. They hypothesize that novice developers have a higher cognitive load compared to experts while performing programming tasks. Their work intends to identify programmers' cognitive activity by measuring the pupillary response. Our work presented here confirms and empirically supports Kontogiorgos' initial hypothesis by providing an in-depth analysis of the pupil dilation changes in the novice group and comparing it to experts. However, the findings of this paper draw upon following different methodology than Kontogiorgos attempts to examine the connection between expertise level (expert/novices) and the performed cognitive effort; testing pupil dilation at multiple maximum points instead of taking one average of dilation of all fixations. Also, we control the expertise variable using the years of programming experience rather than using a combination of independent variables (age, years of experience, and education) as in Kontogiorgos' study.

2.2 Comparing Experts and Novices in Program Comprehension

In the debugging domain, Bednarik [10] compared the debugging strategies of expert and novice developers who alternate between three display methods: the source code, the code visualization, and the output. They found that experts tend to use the source code more, while novices focus on the code visualization and use repetitive fixations. Aljehane at el. showed differences between experts and novices in reading the source code elements [3]. Considering all the identified source code parts in the study, experts showed the ability to comprehend and finish the task using fewer source code elements than novices. The study's results also showed that the differences are significant when reading keywords, method signatures, identifiers, and variable declarations. Moreover, novices have significantly higher gaze visits when looking at more details such as names and operators in if-then-else statements. In terms of providing analysis at the token level, recent work by Madi et al. [46] focused on studying the impacts of token frequency and length in developers' reading strategies. They demonstrated these impacts by analyzing eight eye-related measures for the studied source code token. They found statistical evidence against the null hypothesis when comparing low and high frequency tokens in gaze time and total duration measurements, especially in the novices group. They also found that the length of the source code element has a significant effect on the duration needed to read over tokens for novice programmers.

Abid et al. conducted an eye-tracking study to explain the developers' cognitive model followed while reading programs during a summarization task [1]. The result showed no significant differences in the reading model between experts and novices, and programmers mostly used the bottom-up approach while reading the code. Nevertheless, novices record more gaze time than experts when performing the bottom-up reading approach. A more detailed analysis was done on the same collected dataset [2] that replicates and overcomes the limitation in the prior eye-tracking study conducted by Rodeghero et al. [51]. In this study, Abid et al. [2] compared the developers gaze behaviors collected in a more realistic environment using the iTrace eye tracking infrastructure [53][29]. They showed that on summarization tasks, developers tend to read more on the method body than just the signature. Also, while the method size increased, experts showed a significantly higher gaze when revisiting the method body than when revisiting its signature. Abid et al. found that for both developer groups, the call terms are the most focused location for programmers in reading the code during summarization activity, followed by control flow terms then method signatures.

None of the above studies assess developers' cognitive load, using pupillometry data, with regards to expertise. This work uses the developers' years of programming experience to evaluate their expertise and also features the first attempt to use pupillary response to evaluate the differences between experts' and novices' cognitive workloads during comprehension activities.

3 STUDY DESIGN

This section introduces the dataset being analyzed, eye-tracking metrics that we use to assess expertise during comprehension activities, the participants' descriptions relative to their expertise evaluation and finally, the process of cleaning and transforming the dataset.

3.1 Dataset

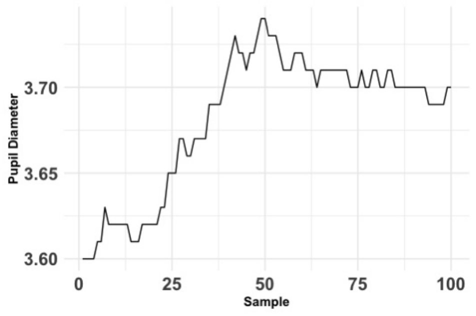
This analysis uses the EMIP dataset known formally as the Distributed Collection of Eye Movement Data in Programming [11]. This dataset is part of the EMIP international workshop, which started in 2013. Due to the lack of availability of a large eye movement dataset, multiple labs from different countries contributed to an eye-tracking experiment to enrich the research field with a large gaze dataset. The same settings were used at each location since the laptop and tracker were shipped to each researcher in order for them to conduct the study in the same setup. They use the SMI RED 250 remote tracker to record the data which was shipped to the participants with all the instructions about running the experiment. To date, the published EMIP dataset has the eye gaze data of 216 participants who conducted the program comprehension experiment with two visual stimuli which are available in three different programming languages. The two programming tasks are called Rectangle and Vehicle, each of which includes a class that matches the name of the task. Subjects were asked to choose a programming language between Java, Python, or Scala with which to start the experiment. After they finish reading the code, participants are asked to solve a comprehension question (multiple choice selection, free-form answer on what is the output, and free-form answer on summarizing the code) to assess their understanding of the code. The majority of the subjects use Java (207 participants), with five data points in Python, and four in Scala. In this analysis, we use the eye-tracking data for participants who use the Java stimulus only.

3.2 Measures

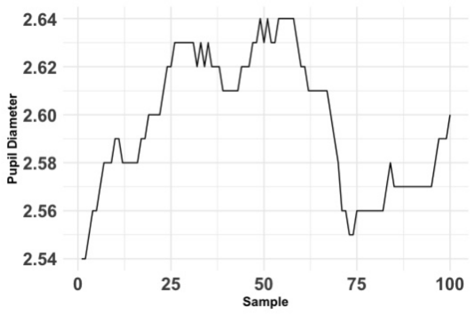
Researchers examine and employ a wide range of eye-tracking measurements in previous software engineering eye-tracking studies [56–58]. In our study, we use ten eye-related metrics to analyze the data and assess the participants' visual attention efforts. These eye-related measurements fall into three categories: fixation metrics, saccade metrics, and pupil size metrics. All measures are available at https://osf.io/rsqdx/ [4].

3.2.1 Fixation Metrics. A fixation [20, 49] happens when the eye is focused and stabilized on one location within the source code elements for a short time. The following fixations-related metrics are included in this study:

- `Number of Fixations`: The total number of fixations that the subject used over the programming tasks (Rectangle.java and Vehicle.java). This metric is calculated for both tasks that the participant used after cleaning the data and removing all invalid eye records. Many of the previous eye-tracking studies use fixation metrics to detect and measure either the search efficiency [26, 38] or to find the most focused areas of interest [18, 65]. Overall, a higher number of fixations indicates a lower efficiency (more visual effort to complete task) to search for relevant information in an area of interest (AOI) or stimuli.
- `Line Coverage`: This metric represents the percentage of lines the participant looked at from all the program lines. Eye gazes are mapped to the corresponding source code line number by using the information provided with the EMIP dataset [11]. The stimulus information included with the dataset shows the lines coordinates for each program. The area of interest (AOI) model proposed by Deitelhoff et al., is used in the mapping process between the eye records and the code lines; this allows the researchers to capturing more AOIs (i.e., lines of code) that are used in comprehending the code [19]. This AOI model approach utilizes the gap



(a)



(b)

Fig. 1. (a) The timeline of the pupil diameter(mm) for novices (N=62) at the beginning of reading during the first trial (Rectangle.java), where the baseline averaged to 3.61 mm from the first 15 samples. (b) The timeline of the pupil diameter(mm) for expert (E1) at the beginning of reading during the first trial (Vehicle.java), where the baseline averaged to 2.57 mm from the first 15 samples.

between the code lines due to the possibility of having fixations that fall in that space. Using this approach improves the number of fixations mapped to the correct line number. However, our intent when calculating the line coverage metric is to capture more AOI transitions in the mapping process, and this requirement is satisfied and proven by Deitelhoff et al. [19].

- `Sum of Fixation Duration`: This represents the sum of all intervals between two fixations. Thus, we calculate the fixation duration that subjects use at each eye record by taking the difference between the timestamp at one fixation (F_i) and the subsequent fixation (F_{i+1}). However, fixation duration is used to get insight into developer performance and their overall visual attention that require them to work on the task [10, 61]. A longer fixation duration means that the participants spend more time understanding and analyzing the task [17] due to code complexity [14] or defect difficulty [16], and more cognitive effort is needed to explore the given tasks [55]. Although fixation counts and gaze time represent developers' visual efforts to perform programming tasks, they are not correlated [57].

3.2.2 Saccade Metrics. A saccade happens between two consecutive fixations. It is a form of navigation behavior. No processing happens during saccades.

- `Saccade Length`:This study defines the saccade length as the sum of the Euclidean distance between consecutive fixations divided by the number of saccades [56].
- `Saccade Duration`:In this study, saccade duration represents the sum of the duration of all saccades divided by the total number of saccades.

3.2.3 Pupil Size Metrics. When the pupil gets dilated, it allows for more light in the eye. However, it happens in different conditions such as in low light situations or as a result of changes in the cognitive efforts of the person [21, 27, 28]. Thus, pupil dilation can provide a sensitive index to reflect the cognitive challenges when working on a difficult task [25, 40, 43]. Therefore, instead of using the exact pupil diameter this study's analysis uses the pupil dilations to measure the differences in memory load between the two groups (experts/novices) [42] as follows.

- `Pupil dilation`: This represents the increase in the pupil's diameter size measured in mm. Since pupil size can increase up to 0.5mm as a response to an increase in cognitive load, here, we consider four maximum values in our study's analysis (0.1, 0.2, 0.3, 0.4) to capture small changes in pupil size increasing over the baseline.

We exclude the changes of 0.5mm in the pupil size analysis because only a few developers have recorded dilation up to this point, especially with experts (only 13 experts). We use a baseline interval of 60ms (15 samples) from starting the task (reading the code) and take into account the trial with which the subject starts. Furthermore, we detect the baseline as the average of the developer's pupil size in the first 60 *ms* after starting fixations on the target (code). For each developer, the relative pupil dilations are obtained by subtracting the baseline value from each pupil diameter value in the trial. Then, we calculate the percentage of the fixations when a subject's pupil gets dilated above their baseline in each tested point. Figure 1 shows the timeline of pupil diameter (100 samples) for both a novice (N62) and an expert (E1) while reading the first trial.

3.3 Expertise Grouping

There are 207 participants we consider in this study. Their years of programming experience ranged between zero and 56. Based on the fact that programming skill increases with the years of experience in the field [67], and based on the findings of the first research question, we classify experts and novices within the experimental group by using their years of programming experience. The median value of the programming experience information provided by the participants (4 years) is used. Excluding those with none (zero) experience, which makes 16 participants out of the total, there are 101 experts with 4+ years of experience, and 90 novices with 0.5 to 4 years of programming experience. Most of the subjects who participated in the experiment fall within 1 to 10 programming years of experience.

3.4 Data Cleaning and Transformation

First, we extract the records related to each stimulus program (Rectangle.java and Vehicle.java, with 18 - 22 LOC) from the raw data for every subject and save the result in csv files. We then check the data for validity, including removing all records when the eye pupil dilation is zero. To validate the comparison and make it easier to compare between participants' eye measurements, this analysis uses a common scale for all the data without changing the range between the values. So, we utilize the min-max normalization to transform all the data records to the same scale by subtracting the minimum from each value, then dividing the result by the difference between the max and the min value (max-min scaling). Thus, the minimum value is transformed to zero, and the maximum value is transformed to 1. This scaling was applied to the fixation metrics and saccade metrics. However, for pupil metrics, there is no need to transform the data records to the same scale. For each developer, the relative pupil dilations are obtained by subtracting the baseline value from each pupil diameter value in the trial. Then, we calculate the percentage of the fixations when the pupil gets dilated above their baseline at each tested point (0.1, 0.2, 0.3, 0.4). The process of outlier removal for pupil dilation data is presented in Section 4.3.

4 RESULTS AND DISCUSSION

We combine the Vehicle and Rectangle program results from the dataset for all 207 participants (novices and experts) who read Java source code, creating a total of 414 data points. To compare the studied eye-tracking measurements of the experts and novices and to find any significant differences between them, we now seek to answer our research questions.

4.1 RQ1: Correlation Between Expertise and Eye Tracking Metrics

The answer to this question we show the correlation between expertise and eye-related metrics. To overcome any errors by dividing the participants into two different groups of expertise we

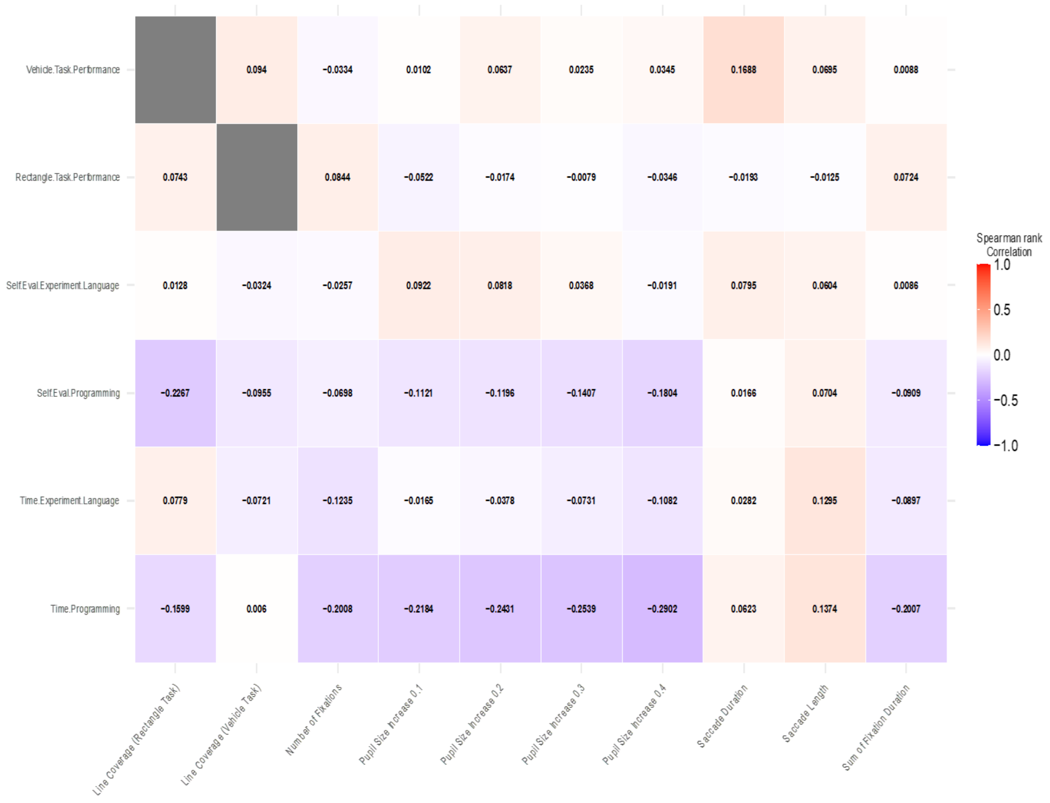


Fig. 2. Correlation matrix of the correlation coefficient result between eye-movement measurements and expertise

ignore the variance between their years of programming. This in turn will allow for finding the best metrics to use when explaining the differences in levels of expertise by using eye-related metrics.

We define a set of independent metrics that have been utilized in previous studies to assess and measure developer expertise. All metadata related to subjects is provided with the EMIP dataset in a separate file [11]. We use the following variables from the metadata to represent expertise:

- `Time Programming`: This metric is measured as years of practicing coding. In this analyzed dataset (EMIP), the distribution of the subjects' years of programming range between 0 and 56.
- `Time Programming in Experiment language`: This is measured as years of programming in the experiment language Java. Relating to the analyzed dataset, 207 participants chose to use Java programs to conduct the study. The subjects' years of experience range between 0 to 30 years of using the Java programming language.
- `Self-Evaluation in programming`: This variable shows the developers' self-evaluations of their programming expertise. Subjects chose between none, low, medium, or high. The metric receives one of the following values: 0 if the subject chooses none for programming expertise, 1 for low, 2 for medium, and 3 for a developer with high expertise.

- `Self-Evaluation in experiment language`: This metric aims to show how developers evaluate their expertise in the Java programming language. It gets a value of 0, 1, 2, or 3 based on participants' entries (none, low, medium, or high).
- `Rectangle task performance and Vehicle task performance`: This binary metrics take a value of 1 if the developer solved the task correctly and a value of 0 if the developer does not choose the correct answer.

To investigate RQ1, we determine the Spearman rank correlation between the extracted eye movement parameters and the expertise metrics. Because of the highly skewed distribution of the data, we adopt the non-parametric Spearman rank correlation, which does not assume either normality distribution or linearity association between values, and is robust to outliers [52]. For each pair of metrics, we compute the Spearman rank correlation coefficient to evaluate the strength of the relationship between eye measurements and expertise variables. Figure 2 contains a color-coded correlation matrix that shows the correlation results between expertise and gaze behavior measurements. We observe that time programming is the expertise variable that correlates with eye-related metrics with the highest correlation coefficient. Typically, a correlation coefficient with the amount of time programming shows small to medium results $| r_s | > 0.1$ and $| r_s | \leq 0.3$ in all eye-related metrics except for saccade duration and line coverage (Vehicle task).

We obtain a negative correlation between the fixation-related metrics, including fixations count and total duration and the time of programming representing expertise level. Also, this correlation is statistically significant with $p < 0.0001$. It is the same with pupil dilation parameters, as they are negatively correlated with years of programming and statistically significant with $p < 0.0001$. However, the time of programming correlates positively with saccade metrics, and it is statistically significant with saccade length, and not with saccade duration. We have also noticed that the correlation with pupil dilation increases with the pupil diameter. The metric of pupil size increased up to 0.4 mm; this has the highest correlation results compared to other pupil-related metrics with time programming ($r_s \approx 0.3$). However, the years of programming experience, in general, appears to be the most accurate metric to use with eye-tracking parameters to estimate expertise in this study. The same result is derived in the original dataset paper [11]. However, adding this paper's correlation analysis further strengthens the case for choosing the years of programming experience to represent expertise statistically.

Discussion: Comparing correlation results across all expertise variables with eye movements parameters shows that using the years of programming experience is the best variable choice to explain overall expertise in this analysis. However, this connection's results reveal insights about the importance of focusing on teaching the programming logic regardless of the language's syntax for beginner programmers. Almost all of the parameters that captured the participants' gaze reading behaviors correlated with programming time higher than that of the other expertise metrics. Thus, we can conclude that expertise represented by the years of programming experience can play an important role on the source code reading behaviors in this experiment. However, this result does not exclude the fact that other expertise metrics could also have an impact on code comprehension, which reflects on developers reading strategies. Nevertheless, in this experiment measuring expertise by the time of programming was the best fit to interpret eye parameters successfully. For this reason, replication experiments with different settings are important to generalize the results, such as assigning participants in multiple groups with different programming languages for each.

RQ1 Finding: Overall, the correlations of developers' years of programming experience indicate a small to a moderate relationship with each eye-tracking metric (except saccade duration and line coverage in the Vehicle task) which performs the best association across approximately all

Table 1. Eye tracking measures for experts vs. novices showing average of data values along with st. dev. (SD)

Eye Movement Measure	Level of expertise	Average of actual data	SD
Number of Fixations	Experts	12500.58	7516.89
	Novices	14656.87	7046.74
Sum of Fixation Duration	Experts	56474.88 *ms*	30941.87
	Novices	65341.34 *ms*	30347.74
Saccade Length	Experts	4.53°	4.58
	Novices	4.17°	5.22
Saccade Duration	Experts	5.9 *ms*	6.65
	Novices	5.1 *ms*	7.03

expertise metrics. In addition, we provide empirical evidence that the more time spent practicing programming regardless of the language that a developer is particularly interested in, the more noticeable increase in the probability of reading programming code efficiently, along with decreasing the cognitive workload and improving the program comprehension process. Because the existence of a correlation does not necessarily mean causation, further investigation is needed to examine the influence of expertise in a particular language or domain that would affect the developer performance in other programming languages.

4.2 RQ2: Metrics Between Experts and Novices

Novices tend to have noticeably higher reading parameters with fixation-related metrics, whereas experts record a higher average on their saccade-related metrics than that of the novice developers. However, the line coverage measurement varies with the task performed. Table 1 shows the average of the actual data in fixation and saccade metrics. To determine the significance, we test these differences between experts and novices using the non-parametric Mann-Whitney test[36] based on the normalized values. We apply the non-parametric Mann-Whitney test because the data is not normally distributed. From Table 2, the results show that the difference between experts and novices is statistically significant in fixation counts, total fixation duration, saccade length, and line coverage in the Rectangle program. In the Vehicle trial, experts cover more lines in the program than novices, but the difference is not significant. Thus, less skilled subjects exhibit more fixations on the code with a longer gaze time duration and a shorter saccade length; which means they perform short skips over the program lines more often than skilled subjects.

Number of Fixations: We find statistical evidence that novices use a higher amount of fixations when comprehending the task. On a scaled average, experts use 0.29 fixations to comprehend the source code, while novices need 0.07 more fixations to read the code. In the actual reading shown in Table 1, experts averaged 12500.58 gaze visits in reading both trials, while novices have an average of 14656.87 visits. Based on the computed probability and Z presented in Table 2, we show that the difference between the two groups in the fixations is significant. The results of the non-parametric Mann-Whitney test for the number of gaze code visit is (Z = -3.748, p = 0.00018) with a medium effect size of d = 0.312. This result indicates that expertise influences the total number of fixations required to understand the programs. This reading behavior was previously presented in the main paper of the EMIP dataset[11], but here we shed light on the degree of these differences between experts and novices when testing the significance of this result.

Total Fixation Duration: We find evidence that novices have a statistically significant higher percentage of gaze time views of the source code than that of experts. Using actual duration in

Table 2. Mann-Whitney test results of experts and novices in fixation-related and saccade-related metrics

Eye Movement Feature	U	p	Z	Cohen's d
Number of Fixations	14142	0.00018	-3.748	0.312
Sum of Fixation Duration	14190	0.00021	-3.703	0.309
Saccade Length	20808	0.01474	-2.439	0.077
Saccade Duration	19286	0.3046	-1.027	0.089
Line Coverage (Rectangle)	3791.5	0.04353	-2.019	0.35
Line Coverage (Vehicle)	4914.5	0.3266	-0.981	0.159

Table 1, experts averaged 56474.88 *ms* of their gaze time viewing the programs while novices spent 65341.34 *ms* of the duration reading and understanding the programs. We determined the significance based on the computed Z and p values (as shown in Table 2). The results show that the average novices' fixation time is significantly higher than experts: U = 14190, p = 0.00021, Z = -3.703 with a medium effect size of $d \approx 0.31$.

This indicates that expert developers are more efficient in using their prior knowledge, thus comprehending the task faster than novices. This observation aligns with the descriptive statistics visualized in the EMIP dataset paper on the influence of expertise [11]. However, note that none of the analysis done in this paper has been done in the dataset paper. The findings of this study emerge from statistical analysis to provide further explanation about expertise effects on fixation duration.

Saccade Length: It is observed that experts tend to make a longer saccade while reading the programs than novices. The results of the average saccade length for all the source code show that experts had an average length of 4.53 deg. while the novices' average saccade length is 4.17 deg. When testing this difference with the Mann-Whitney test, we find that the average experts' saccade is significantly higher than that of novices', with p = 0.01474, Z = -2.439, U = 20808, and a small Cohen's effect size d = 0.07. This result was also observed previously by findings in Busjahn et al. [13]: that experts are better when deciding where to look in the code and can therefore make larger skips to find important parts to understand the program.

Saccade Duration: Related to the saccade duration, we find in this analysis that the saccade duration for the experts lasted between 4 *ms* and 50.75 *ms* for each saccade, whereas the novices spent from 4 *ms* to 83.816 *ms* in each saccade. On average, experts recorded longer saccade durations than the novices by using on average 6 *ms* on saccades per task, while novices took an average of 5.1 *ms*. However, this difference is not significant between experts and novices for saccade duration (p = 0.3046, Z = -1.027) with a small effect size $d \approx 0.1$ (Table 2).

Discussion: Although the result does not show any significant difference between experts and novices in saccade duration, experts used a higher average of saccade duration than novices, thus showing that experts tend to be more efficient/non-linear when reading the source code. We can conclude that although experts used a significantly lower fixation count than the novices, they spent enough time at each fixation to acquire the information they need to comprehend the task. Thus, experts in this experiment are better at reading the task with less fixation and time but with more focus.

Line Coverage: Reading the Rectangle task shows that novices recorded a higher average of line coverage than that of the experts. From mapping the fixation counts to the line number coordinates using the no vertical margins method [19], we find that novices read 20% of the Rectangle class lines, while experts covered about 17% of the code lines. This difference between experts and novices in their code reading coverage is significant with a medium effect size (p = 0.04353, Z = -2.019, d = 0.35). However, with the second task the results show that the average of reading code lines in the expert group is higher than that of the novice group, with 19% for experts and 17% for novices. Nevertheless, this result is not significant for this task (p = 0.3266, Z = -0.981, d = 0.159).

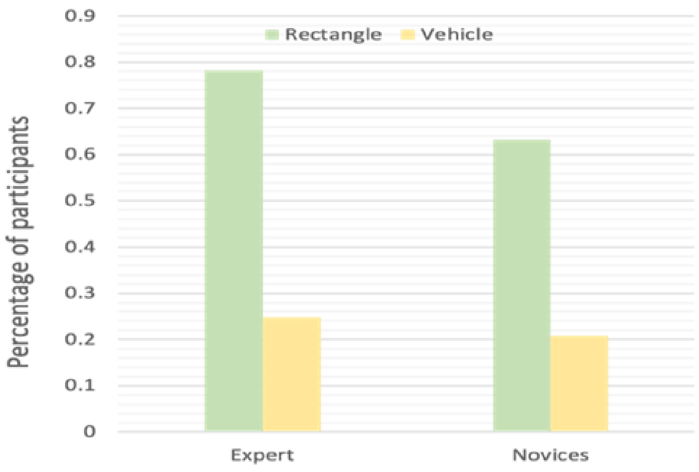


Fig. 3. Participant correctness level for Java code in Rectangle and Vehicle stimuli

Discussion: With studying reading behavior of developers, researchers have determined that experts and novices read code differently when it comes to code word coverage [13], or even in more fine-grained level reading source code elements [3][46]. Thus, this study's result in analyzing reading behavior at line level for the Rectangle task matches those found in previous findings. However, the developers' performance of the Vehicle task may explain the different reading strategies by developers. In the Vehicle task, only 47 developers solved the task and comprehended it correctly, while 160 did not answer the comprehension questions correctly. Figure 3 shows the participants' correctness results for those who chose Java for each task with expertise consideration. According to the authors of the EMIP dataset experiment, subjects understood the main idea of the program but missed the tricky part of the effect on the Vehicle speed by passing a negative number to the acceleration method [11].

RQ2 Finding: To summarize, we find strong statistical evidence concerning the usefulness of eye tracking measurements in the context of assessing expertise, confirming the findings of previous studies. Novice developers have significantly higher fixations with longer durations to comprehend the code than expert developers. Moreover, we observe that experts tend to make longer saccade lengths when moving between the code elements and longer saccade durations than novices.

4.3 RQ3: Cognitive Load and Pupillary Response

Addressing this research question aims to detect evidence about expertise impact on the cognitive process of subjects. Therefore, for each developer, we study their pupillary responses to the reading and understanding process of the two visual stimuli, after which we provide an interpretation of the differences between the experts/novices groups.

We apply outlier filtering on the dilation data, as it is likely to have errors caused by blinking [60]. We calculate the boxplot over the fixations with dilation up to 0.1mm. The outliers include data of 23 participants for both Rectangle and Vehicle tasks. Six of 23 had no dilation i.e., the differences

Table 3. Mann-Whitney test results of experts and novices in pupil-related metrics

Pupil Dilation	U	p	Z	Cohen's d
Pupil Size Increase 0.1	12533	0.00934	- 2.599	0.519
Pupil Size Increase 0.2	11500	0.000104	- 3.882	0.52
Pupil Size Increase 0.3	11813	0.000204	- 3.714	0.447
Pupil Size Increase 0.4	11988	0.000049	- 4.059	0.37

between their pupil dilation and baseline was zero. This is likely due to their wearing of glasses that can affect tracking. After the removal of these data points, all analysis is done with the data after the boxplot outlier removal was complete. After removing all outliers (35 data points) we are left with 379 data points included in the analysis out of all 414 data points.

We compared the distributions of the increase in pupil size above the baseline; for all the tested peak values. Novices show a significantly higher average of fixations with dilated pupils than experts. For a pupil size increase by 0.1*mm*, the results show that novices have a higher average of fixations with dilated pupils than the experts. Experts have 7.73% of their fixations where their pupil sizes tend to be dilated up to 0.1*mm*, while novices recorded an increase in the pupillary response with 12.17% of the total fixations. The same trend is discovered when comparing the average of fixations with dilated pupils up to 0.2*mm*. We notice that novices have pupil dilations more often (7.9% of their fixations) than those of experts (2.3% of their fixations).

In the case of the differences at 0.3*mm*, the analysis shows that less than 1% of experts' fixations have an increase in pupil diameter size up to 0.3*mm*, whereas novices have an average of 3.2% of these same fixations. The results are similar with the differences at 0.4*mm*; we find that on average, novices' fixations with a pupil dilation up to 0.4*mm* is 1.2% of their total gaze distribution, compared to experts who recorded less than 1% of fixations (0.34%) with their pupil dilated up to the same value.

Testing these differences between experts and novices in the pupillary response using the Mann-Whitney test shows that the difference is statistically significant in all tested maximum values of pupil dilation above the chosen baseline. The analysis shows that the significance increased with the pupil size from 0.1*mm* to 0.4*mm*, but the tested effect size decreased (Table 3). As shown in Table 3, based on the calculated p and Z values, we reject the null-hypothesis that stated there is no difference between experts and novices in their pupillary response, which represents the cognitive workload. When data of both trials are combined, the results of the Mann Whitney test for the increase in pupil diameter up to 0.1*mm* and 0.2*mm* is ($U = 12533$, $p = 0.00934$, $Z = -2.599$ with a large Cohen's d effect size of $d = 0.519$) and ($U = 11500$, $p < 0.001$, $Z = -3.882$ with a large Cohen's d effect size of $d = 0.52$), respectively. Similarly, we found evidence that the difference between experts and novices in their pupil dilation up to 0.3 and 0.4*mm* is statistically significant with a medium effect size ($U = 11813$, $p < 0.001$, $Z = -3.714$, Cohen's $d = 0.447$, for pupil size increasing up to 0.3*mm*) and ($U = 11988$, $p < 0.0001$, $Z = -4.059$, Cohen's $d = 0.37$, for pupil size increasing up to 0.4*mm*).

Discussion: We found evidence that pupil size corresponds with expertise level. Novices show a significantly higher average of fixations with an increase in their pupil sizes relative to the baseline. We can therefore conclude that less expert developers perform a higher workload effort and focused attention while processing the comprehension tasks which support the hypothesis that is reported by Kontogiorgos and Manikas [44]. This may suggest that novices see the task as difficult which explains the increase in their pupil size. The task difficulty is also assessed by Fritz et al. [25] and

Hess et al. [31]. Since age might affect the pupil diameter, we ran a quick test and found that pupil dilation tends to go up in response to age (positive correlation), however, for this data set, this very weak correlation (r_s | < 0.1) shows that age has not much effect on pupil dilation.

RQ3 Finding: This study aims to characterize changes in ocular responses, such as pupil dilations. Within all tested pupil diameter values, the analysis identifies significant differences between experts and novices in the pupillary response; these range between medium and large effect sizes. This result would suggest that since the pupillary response is a promising measure of mental effort, it can be used successfully (along with other eye-tracking metrics) to find developer differences in terms of expertise levels.

4.4 Study Implications

These results can be applied to introduce an automatic approach i.e., build a prediction model for expertise with the incorporation of eye movement parameters. We also believe that this result can open the door for new strategies for testing and ranking programmers' skills. It is debatable how to accurately determine the expertise level of developers. Moreover, studies have shown that expertise is more about cognitive skills and making the right decision that reflects on the expert's performance [54]. Thus, the incorporation of eye-tracking technology can add a more reliable way to assess expertise depending on how the developer traverses through the task. In such a situation, identifying expertise can be a valuable aid to provide appropriate help for students who demonstrate less understanding indicated by, for example, an increase in pupil size, more fixations on some area of the code and longer gaze time, or reading the task line by line. These implications are directly applicable to education and practice.

5 THREATS TO VALIDITY

Construct Validity: We note that there is some limitation related to deciding the pupil diameter baseline value for participants. Since the study is conducted by extracting the metrics from a previously published dataset, it is likely that there are other factors that could have possibly affected the pupil diameter, such as maintaining the light brightness in the experiment setting. Relatedly, to reduce the effect on calculation accuracy and to overcome this limitation, we sample and average participants' pupil diameter in the first 15 samples of the starting trial to estimate the pupil baseline value rather than using a specific value. We have noticed that the dilation starts after this baseline interval 60 *ms*, as shown in Figure 1. The risk of having a bias in pupil dilation may arise due to the issue of having different settings. Thus, we calculate the baseline after they start to read the code and then study changes in the pupil size thorough the task. This step would assure that the cognitive load is the only changing variable between the baseline and finishing the task. Fixation counts are inherently dependent on the time on task especially if the time is different for each participant as in our study. We mitigate this risk by also using fixation durations (based indirectly on time on task overall), which we show differs between experts and novices as seen in Table 1. An alternate way to account for different task durations would be to use the fixations per second metric for counts.

Internal Validity: As developers read the code for comprehension purposes and as our goal is to assess expertise in a realistic environment, we removed developers with no years of programming experience from the analysis. Including these may have strongly affected the time and the number of gaze visits due to the completely lack of programming experience. In addition, it can be argued that reading the code for comprehension purposes may have affected the way that the participants read the programs. As such, they dedicated more fixations, gaze time, and mental effort to fully understand the code. However, we could eliminate these factor effects because the subjects did not know before starting the experiment what type of comprehension question they would be given.

External Validity: We believe that these results are generalizable to similar programming tasks and small code snippets.

Conclusion Validity: All appropriate statistical assumptions were tested before standard inferential statistics were run on the data.

6 CONCLUSIONS AND FUTURE WORK

The paper aims to develop a basic scientific understanding of how we can use eye movement metrics, collected during comprehension tasks, to find a distinctive pattern that assesses developers' expertise levels (expert/novice). Our findings, compared to the results of earlier studies that use eye-tracking measures to characterize expertise, add strong evidence that cognitive load analysis can be an effective measurement of expertise. Novice programmers exhibit more fixations and longer durations compared to experts. In applying the analysis of pupil responses between experts and novices, our results show the degree of dilation from the baseline in both groups. Based on these observations, we conclude that eye-movement data contains valuable insights about programmers' skills and levels of expertise.

Our overarching objective in conducting this study is to ultimately develop a model for the automatic classification of coding expertise based on eye-tracking measures. Such a model, if accurate, has very practical applications in both educational and workplace settings. The model can potentially be used for the assessment of student learning outcomes in programming courses or as a method for unbiased placement of students in computer science programming course sequences. The latter is a difficult problem to do accurately in practice. In the workplace, the model has the potential to assess an individual's skills and recommend appropriate training to improve expertise.

7 ACKNOWLEDGEMENT

This work has been funded by the US National Science Foundation under Grant Numbers CNS 17-30181, CNS 18-55753, and CCF 18-55756. Any opinions, findings, and conclusions expressed herein are the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Nahla J. Abid, Jonathan I. Maletic, and Bonita Sharif. 2019. Using Developer Eye Movements to Externalize the Mental Model Used in Code Summarization Tasks. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications (ETRA '19)*. ACM, New York, NY, USA, 13:1–13:9. https://doi.org/10.1145/3314111.3319834 event-place: Denver, Colorado.

[2] N. J. Abid, B. Sharif, N. Dragan, H. Alrasheed, and J. I. Maletic. 2019. Developer Reading Behavior While Summarizing Java Methods: Size and Context Matters. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 384–395. https://doi.org/10.1109/ICSE.2019.00052 ISSN: 1558-1225.

[3] Salwa Aljehane, Bonita Sharif, and Jonathan Maletic. 2021. Determining Differences in Reading Behavior Between Experts and Novices by Investigating Eye Movement on Source Code Constructs During a Bug Fixing Task. In *ACM Symposium on Eye Tracking Research and Applications*. Number 30. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3448018.3457424

[4] Salwa Aljehani and Bonita Sharif. 2023. Studying Developer Eye Movements to Measure Cognitive Workload and Visual Effort for Expertise Assessment - Dataset. https://doi.org/10.17605/OSF.IO/RSQDX

[5] Basma S. Alqadi and Jonathan I. Maletic. 2017. An Empirical Study of Debugging Patterns Among Novices Programmers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle Washington USA, 15–20. https://doi.org/10.1145/3017680.3017761

[6] E. Arisholm, H. Gallis, T. Dyba, and D. I. K. Sjoberg. 2007. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Transactions on Software Engineering* 33, 2 (Feb. 2007), 65–86. https://doi.org/10.1109/TSE.2007.17 Conference Name: IEEE Transactions on Software Engineering.

[7] Brian P. Bailey and Shamsi T. Iqbal. 2008. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *ACM Transactions on Computer-Human Interaction* 14, 4 (Jan. 2008), 1–28. https://doi.org/10.1145/1314683.1314689

[8] Jackson Beatty. 1982. Task-evoked pupillary responses, processing load, and the structure of processing resources. *Psychological Bulletin* 91, 2 (1982), 276–292. https://doi.org/10.1037/0033-2909.91.2.276 Place: US Publisher: American Psychological Association.

[9] Jackson Beatty and Brennis Lucero-Wagoner. 2000. The pupillary system. In *Handbook of psychophysiology, 2nd ed.* Cambridge University Press, New York, NY, US, 142–162.

[10] Roman Bednarik. 2012. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies* 70, 2 (Feb. 2012), 143–155. https://doi.org/10.1016/j.ijhcs.2011.09.003

[11] Roman Bednarik, Teresa Busjahn, Agostino Gibaldi, Alireza Ahadi, Maria Bielikova, Martha Crosby, Kai Essig, Fabian Fagerholm, Ahmad Jbara, Raymond Lister, Pavel Orlov, James Paterson, Bonita Sharif, Teemu Sirkiä, Jan Stelovsky, Jozef Tvarozek, Hana Vrzakova, and Ian van der Linde. 2020. EMIP: The eye movements in programming dataset. *Science of Computer Programming* 198 (Oct. 2020), 102520. https://doi.org/10.1016/j.scico.2020.102520

[12] Roman Bednarik and Markku Tukiainen. 2006. An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the 2006 symposium on Eye tracking research & applications (ETRA '06).* Association for Computing Machinery, New York, NY, USA, 125–132. https://doi.org/10.1145/1117309.1117356

[13] T. Busjahn, R. Bednarik, A. Begel, M. Crosby, J. H. Paterson, C. Schulte, B. Sharif, and S. Tamm. 2015. Eye Movements in Code Reading: Relaxing the Linear Order. In *2015 IEEE 23rd International Conference on Program Comprehension.* 255–265. https://doi.org/10.1109/ICPC.2015.36

[14] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. 2014. What influences dwell time during source code reading? analysis of element type and frequency as factors. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14).* Association for Computing Machinery, New York, NY, USA, 335–338. https://doi.org/10.1145/2578153.2578211

[15] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of code reading to gain more insight in program comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research - Koli Calling '11.* ACM Press, Koli, Finland, 1. https://doi.org/10.1145/2094131.2094133

[16] Nergiz Ercil Cagiltay, Gul Tokdemir, Ozkan Kilic, and Damla Topalli. 2013. Performing and analyzing non-formal inspections of entity relationship diagram (ERD). *Journal of Systems and Software* 86, 8 (Aug. 2013), 2184–2195. https://doi.org/10.1016/j.jss.2013.03.106

[17] Gerardo Cepeda Porras and Yann-Gaël Guéhéneuc. 2010. An empirical study on the efficiency of different design pattern representations in UML class diagrams. *Empirical Software Engineering* 15, 5 (Oct. 2010), 493–522. https://doi.org/10.1007/s10664-009-9125-9

[18] M. E. Crosby and J. Stelovsky. 1990. How do we read algorithms? A case study. *Computer* 23, 1 (Jan. 1990), 25–35. https://doi.org/10.1109/2.48797

[19] Fabian Deitelhoff, Andreas Harrer, and Andrea Kienle. 2019. The Influence of Different AOI Models in Source Code Comprehension Analysis. In *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP).* 10–17. https://doi.org/10.1109/EMIP.2019.00010

[20] Andrew T. Duchowski. 2007. *Eye tracking methodology: Theory and practice.* Springer-Verlag New York Inc.

[21] Maria K. Eckstein, Belén Guerra-Carrillo, Alison T. Miller Singley, and Silvia A. Bunge. 2017. Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development? *Developmental Cognitive Neuroscience* 25 (June 2017), 69–91. https://doi.org/10.1016/j.dcn.2016.11.001

[22] Madeline Endres, Zachary Karas, Xiaosu Hu, Ioulia Kovelman, and Westley Weimer. 2021. Relating Reading, Visualization, and Coding for New Programmers: A Neuroimaging Study. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE).* 600–612. https://doi.org/10.1109/ICSE43902.2021.00062

[23] J. Feigenspan, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg. 2012. Measuring programming experience. In *2012 20th IEEE International Conference on Program Comprehension (ICPC).* 73–82. https://doi.org/10.1109/ICPC.2012.6240511 ISSN: 1092-8138.

[24] J. Feigenspan, M. Schulze, M. Papendieck, C. Kastner, R. Dachselt, V. Koppen, and M. Frisch. 2011. Using background colors to support program comprehension in software product lines. In *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011).* IET, Durham, UK, 66–75. https://doi.org/10.1049/ic.2011.0008

[25] Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using Psychophysiological Measures to Assess Task Difficulty in Software Development. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014).* ACM, New York, NY, USA, 402–413. https://doi.org/10.1145/2568225.2568266 event-place: Hyderabad, India.

[26] Joseph H Goldberg and Xerxes P Kotval. 1999. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics* 24, 6 (Oct. 1999), 631–645. https://doi.org/10.1016/S0169-8141(98)00068-7

[27] Stephen D. Goldinger and Megan H. Papesh. 2012. Pupil Dilation Reflects the Creation and Retrieval of Memories. *Current Directions in Psychological Science* 21, 2 (April 2012), 90–95. https://doi.org/10.1177/0963721412436811 Publisher: SAGE Publications Inc.

[28] Eric Granholm and Stuart R. Steinhauer (Eds.). 2004. Pupillometric measures of cognitive and emotional processes. *International Journal of Psychophysiology* 52, 1 (2004), 1–6. https://doi.org/10.1016/j.ijpsycho.2003.12.001 Place: Netherlands Publisher: Elsevier Science.

[29] Drew T. Guarnera, Corey A. Bryant, Ashwin Mishra, Jonathan I. Maletic, and Bonita Sharif. 2018. iTrace: eye tracking infrastructure for development environments. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, Warsaw Poland, 1–3. https://doi.org/10.1145/3204493.3208343

[30] Gad Hakerem and Samuel Sutton. 1966. Pupillary Response at Visual Threshold. *Nature* 212, 5061 (Oct. 1966), 485–486. https://doi.org/10.1038/212485a0

[31] Eckhard H. Hess and James M. Polt. 1964. Pupil Size in Relation to Mental Activity during Simple Problem-Solving. *Science* 143, 3611 (March 1964), 1190–1192. https://doi.org/10.1126/science.143.3611.1190 Publisher: American Association for the Advancement of Science Section: Reports.

[32] Bert Hoeks and Willem J. M. Levelt. 1993. Pupillary dilation as a measure of attention: a quantitative system analysis. *Behavior Research Methods, Instruments, & Computers* 25, 1 (March 1993), 16–26. https://doi.org/10.3758/BF03204445

[33] Maarten A. Hogervorst, Anne-Marie Brouwer, and Jan B. F. van Erp. 2014. Combining and comparing EEG, peripheral physiology and eye-related measures for the assessment of mental workload. *Frontiers in Neuroscience* 8 (2014). https://doi.org/10.3389/fnins.2014.00322 Publisher: Frontiers.

[34] Morris K. Holland and Gerald Tarlow. 1972. Blinking and Mental Load. *Psychological Reports* 31, 1 (Aug. 1972), 119–127. https://doi.org/10.2466/pr0.1972.31.1.119 Publisher: SAGE Publications Inc.

[35] Morris K. Holland and Gerald Tarlow. 1975. Blinking and Thinking. *Perceptual and Motor Skills* 41, 2 (Oct. 1975), 403–406. https://doi.org/10.2466/pms.1975.41.2.403 Publisher: SAGE Publications Inc.

[36] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. 2013. *Nonparametric Statistical Methods*. John Wiley & Sons. Google-Books-ID: Y5s3AgAAQBAJ.

[37] Shamsi T. Iqbal, Xianjun Sam Zheng, and Brian P. Bailey. 2004. Task-evoked pupillary response to mental workload in human-computer interaction. In *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04*. ACM Press, Vienna, Austria, 1477. https://doi.org/10.1145/985921.986094

[38] Robert J. K. Jacob and Keith S. Karn. 2003. Commentary on Section 4 - Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. In *The Mind's Eye*, J. Hyönä, R. Radach, and H. Deubel (Eds.). North-Holland, Amsterdam, 573–605. https://doi.org/10.1016/B978-044451020-4/50031-1

[39] Angel Jimenez-Molina, Cristian Retamal, and Hernan Lira. 2018. Using Psychophysiological Sensors to Assess Mental Workload During Web Browsing. *Sensors* 18, 2 (Feb. 2018), 458. https://doi.org/10.3390/s18020458 Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[40] Daniel Kahneman and Jackson Beatty. 1966. Pupil Diameter and Load on Memory. *Science* 154, 3756 (1966), 1583–1585. https://www.jstor.org/stable/1720478 Publisher: American Association for the Advancement of Science.

[41] Katja Kevic, Braden M. Walters, Timothy R. Shaffer, Bonita Sharif, David C. Shepherd, and Thomas Fritz. 2015. Tracing Software Developers' Eyes and Interactions for Change Tasks. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 202–213. https://doi.org/10.1145/2786805.2786864 event-place: Bergamo, Italy.

[42] Jeff Klingner. 2010. Fixation-aligned pupillary response averaging. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications - ETRA '10*. ACM Press, Austin, Texas, 275. https://doi.org/10.1145/1743666.1743732

[43] Jeff Klingner, Barbara Tversky, and Pat Hanrahan. 2011. Effects of visual and verbal presentation on cognitive load in vigilance, memory, and arithmetic tasks. *Psychophysiology* 48, 3 (2011), 323–332. https://doi.org/10.1111/j.1469-8986.2010.01069.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-8986.2010.01069.x.

[44] Dimosthenis Kontogiorgos and Konstantinos Manikas. 2015. Towards identifying programming expertise with the use of physiological measures. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-194351

[45] Eduard Kuric and Mária Bieliková. 2014. Estimation of student's programming expertise. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/2652524.2652561

[46] Naser Al Madi, Cole S. Peterson, Bonita Sharif, and Jonathan I. Maletic. 2021. From Novice to Expert: Analysis of Token Level Effects in a Longitudinal Eye Tracking Study. IEEE Computer Society, 172–183. https://doi.org/10.1109/ICPC52881.2021.00025

[47] S. P. Marshall. 2002. The Index of Cognitive Activity: measuring cognitive workload. In *Proceedings of the IEEE 7th Conference on Human Factors and Power Plants*. 7–7. https://doi.org/10.1109/HFPP.2002.1042860

[48] Oskar Palinko, Andrew L. Kun, Alexander Shyrokov, and Peter Heeman. 2010. Estimating cognitive load using remote eye tracking in a driving simulator. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications -*

ETRA '10. ACM Press, Austin, Texas, 141. https://doi.org/10.1145/1743666.1743701

[49] K. Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin* 124, 3 (1998), 372.

[50] Filippo Ricca, Massimiliano Di Penta, Marco Torchiano, Paolo Tonella, and Mariano Ceccato. 2007. *The Role of Experience and Ability in Comprehension Tasks Supported by UML Stereotypes.* https://doi.org/10.1109/ICSE.2007.86 Pages: 384.

[51] Paige Rodeghero, Collin McMillan, Paul W. McBurney, Nigel Bosch, and Sidney D'Mello. 2014. Improving automated source code summarization via an eye-tracking study of programmers. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, Hyderabad, India, 390–401. https://doi.org/10.1145/2568225.2568247

[52] Patrick Schober, Christa Boer, and Lothar A. Schwarte. 2018. Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia* 126, 5 (May 2018), 1763–1768. https://doi.org/10.1213/ANE.0000000000002864

[53] Timothy R. Shaffer, Jenna L. Wise, Braden M. Walters, Sebastian C. Müller, Michael Falcone, and Bonita Sharif. 2015. iTrace: Enabling Eye Tracking on Software Artifacts Within the IDE to Support Software Engineering Tasks. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 954–957. https://doi.org/10.1145/2786805.2803188 event-place: Bergamo, Italy.

[54] James Shanteau. 1992. Competence in experts: The role of task characteristics. *Organizational Behavior and Human Decision Processes* 53, 2 (Nov. 1992), 252–266. https://doi.org/10.1016/0749-5978(92)90064-E

[55] Z. Sharafi, A. Marchetto, A. Susi, G. Antoniol, and Y. Guéhéneuc. 2013. An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In *2013 21st International Conference on Program Comprehension (ICPC)*. 33–42. https://doi.org/10.1109/ICPC.2013.6613831

[56] Z. Sharafi, T. Shaffer, B. Sharif, and Y. Guéhéneuc. 2015. Eye-Tracking Metrics in Software Engineering. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*. 96–103. https://doi.org/10.1109/APSEC.2015.53

[57] Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. 2020. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering* 25, 5 (Sept. 2020), 3128–3174. https://doi.org/10.1007/s10664-020-09829-4

[58] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. 2015. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology* 67 (Nov. 2015), 79–107. https://doi.org/10.1016/j.infsof.2015.06.008

[59] Nischal Shrestha, Colton Botta, Titus Barik, and Chris Parnin. 2020. Here We Go Again: Why Is It Difficult for Developers to Learn Another Programming Language?. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 691–701. ISSN: 1558-1225.

[60] Sylvain Sirois and Julie Brisson. 2014. Pupillometry. *WIREs Cognitive Science* 5, 6 (2014), 679–692. https://doi.org/10.1002/wcs.1323 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcs.1323.

[61] Zéphyrin Soh, Zohreh Sharafi, Bertrand Van den Plas, Gerardo Cepeda Porras, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2012. Professional status and expertise for UML class diagram comprehension: An empirical study. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*. 163–172. https://doi.org/10.1109/ICPC.2012.6240484 ISSN: 1092-8138.

[62] E. Soloway and K. Ehrlich. 1984. Empirical Studies of Programming Knowledge. *IEEE Transactions on Software Engineering* SE-10, 5 (Sept. 1984), 595–609. https://doi.org/10.1109/TSE.1984.5010283 Conference Name: IEEE Transactions on Software Engineering.

[63] C. W. Telford and N. Thompson. 1933. Some factors influencing voluntary and reflex eyelid responses. *Journal of Experimental Psychology* 16, 4 (1933), 524–539. https://doi.org/10.1037/h0071694 Place: US Publisher: Psychological Review Company.

[64] Jason S. Tsukahara, Tyler L. Harrison, and Randall W. Engle. 2016. The relationship between baseline pupil size and intelligence. *Cognitive Psychology* 91 (Dec. 2016), 109–123. https://doi.org/10.1016/j.cogpsych.2016.10.001

[65] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. 2006. Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 symposium on Eye tracking research & applications - ETRA '06*. ACM Press, San Diego, California, 133. https://doi.org/10.1145/1117309.1117357

[66] Iris Vessey. 1985. Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies* 23, 5 (Nov. 1985), 459–494. https://doi.org/10.1016/S0020-7373(85)80054-7

[67] Chak Shun Yu, Christoph Treude, and Maurício Aniche. 2019. Comprehending Test Code: An Empirical Study. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 501–512. https://doi.org/10.1109/ICSME.2019.00084 ISSN: 2576-3148.

Received November 2022; revised February 2023; accepted March 2023