2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC) | 978-1-6654-1403-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICPC52881.2021.00025

From Novice to Expert: Analysis of Token Level Effects in a Longitudinal Eye Tracking Study

Naser Al Madi
Department of Computer Science
Colby College
Waterville, Maine, USA
nsalmadi@colby.edu

Cole S. Peterson, Bonita Sharif
Department of Computer Science and Engineering
University of Nebraska–Lincoln
Lincoln, Nebraska, USA
cole.scott.peterson@huskers.unl.edu, bsharif@unl.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, Ohio, USA
jmaletic@kent.edu

***Abstract*—Program comprehension is a vital skill in software development. This work investigates program comprehension by examining the eye movement of novice programmers as they gain programming experience over the duration of a Java course. Their eye movement behavior is compared to the eye movement of expert programmers. Eye movement studies of natural text show that word frequency and length influence eye movement duration and act as indicators of reading skill. The study uses an existing longitudinal eye tracking dataset with 20 novice and experienced readers of source code. The work investigates the acquisition of the effects of token frequency and token length in source code reading as an indication of program reading skill. The results show evidence of the frequency and length effects in reading source code and the acquisition of these effects by novices. These results are then leveraged in a machine learning model demonstrating how eye movement can be used to estimate programming proficiency and classify novices from experts with 72% accuracy.**

***Index Terms*—eye tracking, source code, natural text, expertise, empirical study, token effects**

I. Introduction

Software development involves diverse activities such as program composition, comprehension, debugging, modification, and learning [1, 2]. However, the overwhelming majority of software development time is spent reading source code in a process known formally as program comprehension [3]. Studies have found that programmers spend more than 50 percent of their time on activities that reflect searching for information [3–8]. Program comprehension is defined as the process of understanding how a software system or part of it works [3]. Program comprehension is a vital skill for understanding the intended behavior of source code which can be written by the same person or someone else. Without understanding existing source code, debugging, modifying, reusing, and maintaining software is impossible [9]. From an economic standpoint, software maintenance cost is the biggest factor in creating software systems [10, 11].

The process of reading source code shares many aspects with reading natural text, yet the cognitive processes involved in reading source code are much less understood and studied. A contributing factor is the recency of programming languages and the cumulative evidence that shows reading source code differs fundamentally in purpose, syntax, semantics, and viewing strategy from natural text [12–15].

The differences between source code and natural text prohibit the extension of the results of natural text reading research to source code, leaving a cloud of uncertainty on the commonalities and differences between natural text and source code reading. At the same time, understanding and modeling the cognitive processes involved in reading source code grants a way to increase the efficiency and productivity in learning to program and also helps understand the software development process in general.

In this paper, we focus on the frequency and length effects on eye movement in reading source code in relation to programming experience. A *fixation* is the stabilization of the eye over a particular area of interest. The period of time our eyes stay focused over the area is the *fixation duration*. Visual features of text are extracted only during fixations [16]. The frequency effect refers to the difference in fixation duration for high-frequency and low-frequency words (in our case tokens) in which low-frequency words are fixated longer than high-frequency words [16]. The frequency of a word (in natural language) is measured by counting the number of times a word appears in books and articles. We measure the frequency of tokens by examining open source systems. Word length effect refers to the difference in reading time in relation to word (token) length, as longer words, in number of characters, take longer fixation durations [16, 17] to be read and understood.

The frequency and length effects are well studied and established in natural language reading [18–20], yet the presence of the frequency and length effects in reading source code, especially in relation to skill level, is still unclear. Through an existing dataset of a longitudinal eye tracking study of novices and experts with source code and natural text [12] we address the following research questions:

- RQ1: Does token frequency influence the eye movement of experienced programmers in reading source code?
- RQ2: Does token frequency influence the eye movement of novices during source code reading?
- RQ3: When do novices acquire the frequency effect in reading source code?
- RQ4: Does token length influence eye movement in reading source code?
- RQ5: Can we distinguish novices from experienced programmers based solely on eye movement?

978-1-6654-1403-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ICPC52881.2021.00025

Studying the frequency and length effects in source code leads the way in understanding the most suitable coding styles and best practices that enhance programmer productivity and comprehension. Our focus on classifying novices from experts based on their eye movement is motivated by our long term vision of an IDE that adapts its interface and features to match the skill level of the user. Additionally, the work can form the basis to automatically assess programmer skill level.

Our results support the presence of the frequency and length effects on the eye movement of novice and experienced readers of source code. In addition, we show that the frequency effect is acquired gradually by novice programmers during an introductory Java course. Finally, we demonstrate how token level analyses of eye movement during reading source code is used to classify novices from experts with 72% accuracy.

The contributions from this paper are as follows:

- Provide evidence to support the presence of the frequency and length effects in reading source code.
- Show the gradual acquisition of the frequency effect by novices.
- Demonstrate how eye movement over source code is used to estimate program reading skill level to classify novices from experts.

II. Background and Related Work

Eye tracking is one of the key tools in software engineering that provides evidence on attention and the cognitive processes of programmers [21]. This popularity is evident by surveying 31 papers in the field in 2015 [22] and 63 papers in 2018 [21]. The use of eye tracking in software engineering can be categorized into five groups: Program comprehension, debugging, model comprehension, collaborative programming, and traceability [21, 22]. Two patterns can be identified in previous research - First, the collective agreement that reading source code differs fundamentally from reading natural text [12, 23, 24]. Second, that there are differences in eye movement of novices and experts [25–27].

A. Eye Movement on Natural Text and Source Code

The first eye tracking study in software engineering by Crosby et al. [23] found that subjects need a significant number of additional fixations to comprehend algorithms in comparison to natural text. In addition, Busjahn et al. [24] found that source code received higher fixation durations and more regressions (jumps back) consistently on a statistically significant level. Moreover, in a later study, Busjahn et al. [12] focus on the scanning strategies in reading source code and natural text. They investigate the linearity of reading natural text and source code. Natural text is read from top to bottom and from left to right, some skipping and regressions do occur but eye movement patterns match to a great extent the linear nature of the text. The study found that programmers look at source code less linearly and to some extent their eye movement is coupled with the control flow of the program.

At the token level, Busjahn et al. [14] present an investigation of token level effects in source code reading where the eye movement of 15 programmers is recorded with a focus on attention distribution over code elements. The paper uses dwell time as a proxy for attention over lexical elements and explored the frequency effect in Java keywords. The study found no effect for frequency on eye movement in reading source code, unlike natural text. This result deserves attention as the frequency effect describes a reduction in the duration of time needed to recognize frequent visual stimuli. Therefore, the frequency effect is a result of neurons forming more synapses as a result of frequent activation. At least theoretically, a frequency effect could exist in source code.

The work by Busjahn et al. [14] found no evidence of the frequency effect in reading Java keywords. We extend this investigation to include all types of Java tokens, such as identifiers, separators, operators, and literals. In addition, we conduct our examination with novices and experienced Java programmers, and we conduct our frequency calculations with two normalization techniques (division and matching).

B. Eye Movement Differences between Novices and Experts

A significant pattern in the study of eye movement in software engineering is the difference between novice and expert readers. This is a shared phenomena between natural text and source code [28]. In an early study, Crosby et al. [25] focused on the novice vs. expert viewing strategies of *beacons* – the surface features of computer programs that serve as keys to facilitate program comprehension. The study found that novices do not benefit from comprehension aids, while experts take advantage of the visual aids in the program to support comprehension. In addition, Bednarik et al. [26] focused on visual attention of novice and experienced programmers in a debugging task with multiple representations of the code. The study found distinct viewing patterns between novices and experts, where novices frequently shift attention between the code and the visual representation of the code while experts focus more on the code and the output.

In a study focusing on task difficulty assessment in software development, Fritz et al. [27] used eye tracking, electroencephalography, and electrodermal activity to determine the perceived task difficulty. The study uses NASA Task Load Index (NASA-TLX) [29] as an account of task difficulty, the tasks used C# code followed by multiple choice questions about the code. The results of the study indicate that eye movement could be used for difficulty assessment and skill level should be considered as a factor for perceived difficulty.

C. Frequency and Length Effects in Eye Movement on Natural Text

Many studies on eye movement in reading natural text have appeared over the years [16, 18, 30, 31]. The frequency effect refers to the difference in reaction times for high-frequency and low-frequency words in which low-frequency words are fixated longer than high-frequency words [16]. The frequency of a word is measured through counting the number of times a word appears in books, articles, and various sources. Word frequency for natural languages has been organized in lexical

databases such as the CELEX database [32], and it is measured in words per-million. High-frequency words appear thousands of times in a million words, an example of that is the word "the" which appears approximately 65,000 times-per-million [32]. Word length effect refers to the difference in reading time in relation to word length, as longer words in number of characters take longer fixation durations [16, 17]. Word length effect comes from visual acuity as processing more characters that are distributed further from the center of the fixation takes more time than a shorter word [16].

One of the early studies on the frequency and length effects on natural text produced the effects in a number of different experimental situations [33]. Another study on the lexical complexity and fixation times found strong effect of word frequency where low frequency words received longer fixation duration than high-frequency words [18]. In addition, focusing on the differences between younger (novices) and older (experts) readers a more recent study found that the frequency effect is larger with older readers than younger ones [31]. Another study focused on the frequency effect in relation to average and skilled readers found that the magnitude of the frequency effect is influenced by reading skill [34]. These results form the basis to investigate the frequency effect as an indicator of programming skill in source code reading.

III. Source Code Frequency and Length

In the context of our research questions, we need to know the frequency of every code token to inspect the difference between low- and high-frequency tokens on eye movement duration. In this section we present our methodology for calculating token frequency and length for source code tokens in order to measure the influence of token frequency and length on fixation duration.

Frequency information for natural text is obtained from lexical databases, but no such resources exist for source code. Therefore, we use 10 Java repositories to calculate the frequency of each token. We use repositories that are reported to provide a language model with low perplexity [35], and fulfilled the guides [36] of selecting meaningful repositories sets. The repositories represent approximately 3.9 million lines of code resulting in approximately 14.3 million tokens, after removing comments from code. Only Java files were processed, code in other languages is excluded. Table I shows the details of the selected Java repositories and the number of tokens in each repository.

The frequency of a token is estimated by counting the number of times the token appeared in all repositories over the number of all tokens in all repositories normalized by one million:

$$frequency(token) = \frac{count(token)}{count(vocabulary)/(1,000,000)}$$

Vocabulary is the set of all tokens that appear in the 10 repositories. Token frequency is normalized by one million to provide a reproducible estimation of how common a word or a token is in a specific language.

TABLE I
Java repositories used to calculate token frequencies.

Repository	Files	Lines	Tokens
Ant	1,314	304,957	1,053,481
Batik	1,651	353,516	1,185,185
Cassandra	2,673	586,451	2,055,723
Eclipse	154	25,914	77,699
Log4J	309	60,078	208,578
Lucene	8,467	1,874,373	6,900,196
Maven2	378	60,775	206,887
Maven3	834	113,384	388,503
Xalan-J	958	348,769	1,355,646
Xerces2	833	261,312	958,345
Total	17,571	3,979,251	14,390,243

Token length is calculated as a direct measure of the number of characters in a token. When studying the frequency effect, it is common practice to normalize eye movement duration by word/token length, as this provides a more reliable measure of only the frequency effect and removes any length effect that influences eye movement duration [14, 18].

IV. Experimental Dataset and Metrics

In this section we describe the preexisting dataset we used in this study, which was first presented in [12]. In addition, we present the eye movement metrics, filtering, and transformations that we perform in our analyses. A complete replication package is available at https://osf.io/t5je9/.

A. Participants and Apparatus

We now summarize relevant aspects of the longitudinal experiment presented in [12]. The 20 participants consisted of 14 people with no previous programming experience and six experienced professional software developers. Novices were college students with the exception of two subjects and enrolled in a Java course consisting of six learning modules. Each learning module took several weeks in duration and an eye movement recording session with code and natural language text followed each module. The six experienced programmers had 5 to 28 years of programming experience and they were recorded only once. An SMI RED-m remote eye tracker with a sample rate of 120 Hz is used in the experiment. The screen resolution was set to 1680x1050 for novices (done in a lab) and 1280x1024 for experts (outside the lab). The font used was Arial with size no less than 13 pt and no greater than 16 pt and varied for each lesson making sure that everything was clear and calibration was successfully done.

B. Experiment Design and Hypothesis

As reported in [12], the experiment material consists of programs that are 4 to 28 lines long, subjects are asked to read three programs after they completed each module. Two of the programs are in pseudocode and programs progress in complexity throughout the six learning modules. An experiment block diagram showing this is given in Figure 1.

Our methodology consists of examining expert and novice eye movement over source code to determine the influence

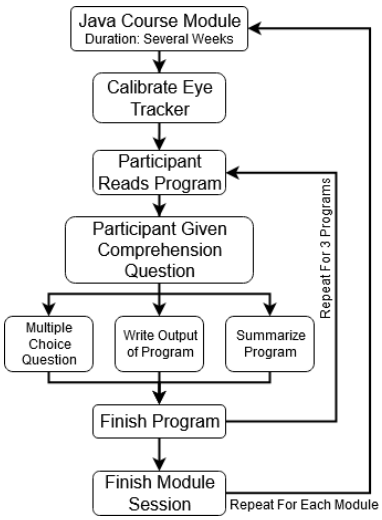

Fig. 1. Experimental Design Workflow

of the frequency effect on their eye movement. The six Java learning modules contain source code tokens with diverse lengths and frequencies. Our *hypothesis* is that novices have not been exposed to source code tokens initially, therefore their eye movement will not show an influence of token frequency on eye movement duration. In contrast, experts who have been exposed to source code tokens extensively will show an influence of the frequency effect and display shortened fixation durations on high-frequency tokens. If this is true, then we expect novices to acquire the frequency and length effects after a certain amount of exposure to programming in Java, which we measure in learning modules.

C. Tokenization and Filtering

According to Schütze et al. [37], tokenization is the process of splitting a character sequence into pieces called tokens. Tokenization is language and application specific, and in the context of eye movement over source code, a token includes any source code element (i.e. identifier, keyword, separator, operator, and literal) that is surrounded by spaces (including the beginning and ending of each line). Figure 2 shows collection of tokens highlighted in different colors, notice that some tokens are separators or operators that can be one character long.

Since our study focuses on token level effects, each token of the source code stimuli is mapped to an area of interest (AOI). An example of the token level AOIs used in this analysis is seen in Figure 2 (this mapping is done using EyeCode[1]). Keywords and symbols are individual tokens. For identifiers, we do not split them apart or split on dots. For example, ”text.substring” is one token, not two separate tokens.

After this mapping is complete, the first and last tokens of each line are removed as fixation duration is irregular and prolonged at the beginning and ending of a line of code

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

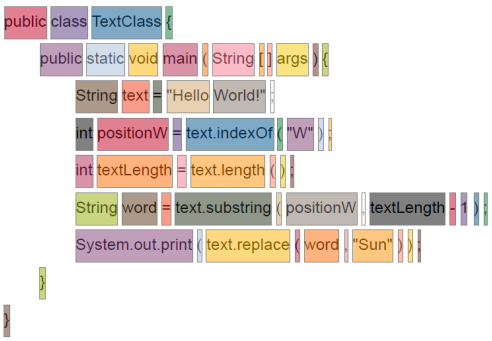

Fig. 2. A sample program with token level AOIs shown.

(sentences in natural text) [38]. In addition, fixations with duration less than 90 ms and greater than 500 ms are removed as very short fixations represent eye tracker error or inter-word regressions caused by motor error. Fixations longer than 500 ms often occur at the end of a sentence and are not representative of the influence of linguistic factors such as frequency and length on eye movement [38].

D. Eye Movement Metrics

We measure the influence of the frequency and length effects on eight eye-movement metrics that are calculated for each token. Four duration metrics and four probability metrics as defined below [38, 39].

Let F be the set of all fixations on token n before moving to token n+1. F_i is the fixation at index i where i = {1, 2, ..., k}. S refers to the number of subjects in the experiment, and R is all re-fixations on token n after moving to token n+1 and making a regression. The duration metrics are:

1) First-Fixation Duration (FFD): The duration of the first fixation on a word/token.

$$\text{FFD} = \left\{ \frac{\sum_1^S F1}{S} \quad , k > 0 \right\}$$

2) Single-Fixation Duration (SFD): The duration of the fixation when only one fixation was made on the word/token.

$$\text{SFD} = \left\{ \frac{\sum_1^S F1}{S} \quad , k = 1 \right\}$$

3) Gaze Duration (GD): The sum of all fixations on word/token n before moving to word/token n+1.

$$\text{GD} = \left\{ \frac{\sum_1^S F_1 + F_2 + ... + F_k}{S} \quad , k > 0 \right\}$$

4) Total Time (TT): The sum of all fixations on word/token n (including regressions).

$$\text{TT} = \left\{ \frac{\sum_1^S F_1 + F_2 + ... + F_k + R_1 + R_2 + ... + R_m}{S} \quad , k > 0 \right\}$$

Assume that S_F is the number of subjects who make at least one fixation on the token, S_1 is the number of subjects who make only one fixation, S_2 is the number of subjects who

make 2 fixations or more, and S_0 is the number of subjects who skip the token. The probability metrics for that token are:

1) fixation probability (PrF): The probability of a word/token in getting a fixation.

$$\text{PrF} = \frac{S_F}{S}$$

2) probability of making exactly one fixation (Pr1): The probability of a word/token in getting exactly one fixation.

$$\text{Pr1} = \frac{S_1}{S}$$

3) probability of making two or more fixations (Pr2): The probability of a word/token in getting two or more fixation.

$$\text{Pr2} = \frac{S_2}{S}$$

4) probability of skipping (PrS): The probability of a word/token in getting no fixation (getting skipped).

$$\text{PrS} = \frac{S_0}{S}$$

V. Experimental Results

A. RQ1: Does token frequency influence the eye movement of experienced programmers in reading source code?

In answering this question we attempt to find evidence for the frequency effect in source code reading with experts. A small previous study by Busjahn et al. [14] reported a negative result on the presence of the frequency effect in reading source code. For that reason, we replicate the previous study with a few changes and borrow the methodology from key research that presents the frequency effect in natural text [19, 20]. The frequency effect appears as a reduction in fixation duration on high-frequency words in comparison to low frequency words with the same length. The normalization by length is done by dividing eye movement duration by the number of characters (as in [14]), or by selecting high and low frequency target words of the same length (as in text [19, 20]). To answer our question thoroughly we do both division and matching methods of normalization and verify our results statistically.

The influence of frequency on eye movement is acquired by experience and exposure to the language in question, therefore we focus our attention on experienced programmers to investigate the presence of the frequency effect in reading source code. We consider low-frequency words with frequency below 100/million (e.g., height, method), and high-frequency words with frequency above 10,000/million (e.g., public, return).

Starting with normalizing by division, we divide eye movement duration over a token (i.e. Single Fixation Duration, First Fixation Duration, Gaze Duration, and Total Time) by the number of characters in that token. For example, if the Total Time is 400 ms on a token of 4 characters, the normalized total time (n_Total Time) is 100 ms. Excluding tokens consisting of exactly one letter space in length, such as ")", we end up with 234 low and high frequency target tokens. Single character tokens are filtered as natural language text studies have showed single character are often skipped or processed through the parafovea (peripheral vision) without a direct fixation [17, 38].

TABLE II
Mean eye movement durations (in milliseconds) normalized by **division** for low-frequency and high-frequency code tokens. p shows the probability that low-frequency and high-frequency belong to the same population.

Metric	Low	High	p
n_Single Fixation Duration	53	52	0.11
n_First Fixation Duration	49	55	0.75
n_Gaze Duration	79	81	0.60
n_Total Time	133	114	0.34

Table II shows mean eye movement metrics (normalized by division) for low/high-frequency code tokens. We observe that low-frequency tokens take longer fixation durations than high-frequency tokens in the total time metric only. Yet when this difference in duration is tested statistically through Wilcoxon signed-rank test the difference between high and low frequency tokens is not statistically significant.

Wilcoxon signed-rank test is used here instead of paired *t* test since the data residuals are not normally distributed, which is one of the requirements of the paired *t* test. Wilcoxon signed-rank test is the non-parametric version of the paired *t* test, and use it here since we have two within subject categories (high frequency and low frequency).

This result replicates the results of the previous study by Busjahn et al. [14] which found no statistically significant difference in eye movement over low- and high-frequency source code tokens when normalized by length through division.

TABLE III
Mean eye movement durations (in milliseconds) normalized by **matching** for low-frequency (Low) and high-frequency (High) code tokens. p shows the probability that low-frequency and high-frequency belong to the same population. Bold when p <0.05.

Metric	Low	High	p
n_Single Fixation Duration	220	225	0.11
n_First Fixation Duration	224	228	0.34
n_Gaze Duration	411	345	0.24
n_Total Time	937	425	**0.02**

We use another normalization technique, namely matching, to thoroughly test the presence of the frequency effect in reading source code. The target tokens are matched with 6 characters in length, with 42 high-frequency tokens and 36 low-frequency tokens. The resulting 78 target tokens are more than double the number of target words reported in [19, 20]. Table III shows mean eye movement durations for length-matched low- and high-frequency tokens.

Both Gaze Duration (GD) and Total Time (TT) appear to show shorter fixation duration for high-frequency tokens, yet statistical testing through Wilcoxon signed-rank test showed that only Total Time was statistically significant. The effect size (d = .92) was found to be large according to a Cohen's d test. Table III shows that the difference between low- and

high-frequency tokens in Total Time is statistically significant (with $p < 0.05$), providing evidence to support the presence of the frequency effect in reading source code. In fact, the mean Total Time difference between low and high frequency tokens is more than double.

To summarize, our results suggest that there is a difference between low- and high- frequency tokens in the eye movement of experienced programmers. This difference is statistically significant only in the Total Time metric when eye movement is normalized by matching tokens of equal length. It is important to note that the lack of statistical significance in other metrics does not indicate a negative result.

B. RQ2: Does token frequency influence the eye movement of novices during source code reading?

In this section, we focus on the eye movement of novices during a Java course consisting of six learning modules, where eye movement over source code was recorded after each module. Our hypothesis is that linguistic effects such as the frequency effect are an indication of skilled reading, and that the frequency effect appears as a result of repeated exposure to source code. Therefore, we search for evidence of the frequency effect in all modules, and in the next section we focus on each module individually.

To verify that the frequency effect is not observed in the eye movement of novices we apply identical filtering as the previous question, where low-frequency words have a frequency below 100/million, and high-frequency words have a frequency above 10,000/million. Similar to the previous section, single character tokens are filtered as they are highly likely to be processed in the parafovea (peripheral vision). In addition, we focus on novice data in all six learning modules, and we normalize eye movement duration by length using division and matching, similar to the previous question.

TABLE IV
MEAN NOVICE EYE MOVEMENT DURATIONS (IN MILLISECONDS) NORMALIZED BY **DIVISION** FOR LOW-FREQUENCY AND HIGH-FREQUENCY CODE TOKENS. p SHOWS THE PROBABILITY THAT LOW-FREQUENCY AND HIGH-FREQUENCY BELONG TO THE SAME POPULATION. BOLD WHEN $p < 0.05$.

Metric	Low	High	p	Effect Size
Single Fixation Duration	68	50	**0.001**	0.512
First Fixation Duration	60	48	**0.003**	0.364
Gaze Duration	100	71	**0.005**	0.466
Total Time	178	119	**0.0009**	0.368

Processing 241 high-frequency tokens and 481 low-frequency tokens in all six modules yielded a statistically significant advantage for high-frequency tokens in terms of shorter eye movement durations. Table IV shows mean novice eye movement durations in the all six learning modules normalized (by division), and every metric shows shorter mean duration for high-frequency tokens in comparison to low-frequency tokens. When this difference is tested statistically through Wilcoxon signed-rank test, the difference was statistically significant with $p < 0.05$. Statistical testing of individual modules and accompanying post-hoc tests will be presented in the next section. Wilcoxon signed-rank test is used here again since the data residuals are not normally distributed, and we have two within-subject categories (high frequency and low frequency).

TABLE V
MEAN NOVICE EYE MOVEMENT DURATIONS (IN MILLISECONDS) NORMALIZED BY **MATCHING** FOR LOW-FREQUENCY AND HIGH-FREQUENCY CODE TOKENS. p SHOWS THE PROBABILITY THAT LOW-FREQUENCY AND HIGH-FREQUENCY BELONG TO THE SAME POPULATION. BOLD WHEN $p < 0.05$.

Metric	Low	High	p	Effect Size
Single Fixation Duration	264	232	0.465	
First Fixation Duration	235	230	0.753	
Gaze Duration	480	352	**0.046**	0.653
Total Time	1178	595	**0.046**	0.934

We now present the results of normalization by matching. Despite the small sample size for tokens of equal length, in all six modules 186 high-frequency tokens and 42 low-frequency tokens where matched with length equal to six characters. This token length is chosen based on the most common token length to aid in finding the largest sample of high- and low-frequency tokens of equal lengths. Nonetheless, in some instances no matches were found resulting in a small sample size.

Table V presents data from all six learning modules, where novice eye movement is compared over low- and high-frequency tokens normalized by matching. The table shows that novice eye movement over low frequency tokens have longer durations than high frequency tokens in every metric. Again, Wilcoxon signed-rank test is used, and the difference is statistically significant in the Gaze Duration and Total Time metrics.

To summarize, we found evidence of the frequency effect in the eye movement of novices when normalized by division and matching, the difference between high- and low- frequency tokens is statistically significant. This result suggests that novices acquire the frequency effect during a Java course.

C. RQ3: When do novices acquire the frequency effect in reading source code?

To answer this question we calculate the influence of the frequency effects on eye movement of novices longitudinally over the duration of a Java course (six learning modules). Since the frequency effect appears as a reduced fixation duration over high-frequency tokens in comparison to low-frequency tokens, we sample tokens from each of the six modules into high-frequency (>10,000/million) and low-frequency (<100/million) to compare normalized mean duration. We exclude tokens that are single characters. All duration in this section are normalized by division, since matching tokens of equal lengths reduces the size of the sample data.

Table VI is a comparison of eye movement durations over high/low frequency code tokens in each learning module. This module level view allows for an examination of the frequency effect as reduced eye movement duration over high-frequency tokens in comparison to low-frequency tokens from the same module. One observation in the last module is that in every eye

TABLE VI
NOVICE MEAN EYE MOVEMENT DURATION OVER LOW-FREQUENCY AND HIGH-FREQUENCY TOKENS IN EACH LEARNING MODULE (DURATIONS NORMALIZED BY LENGTH THROUGH DIVISION).

learning module	metric	high-frequency	low-frequency
1	SFD	83	61
1	FFD	77	61
1	GD	92	113
1	TT	96	195
2	SFD	63	56
2	FFD	58	60
2	GD	72	76
2	TT	103	102
3	SFD	42	74
3	FFD	46	71
3	GD	71	102
3	TT	129	155
4	SFD	60	49
4	FFD	41	47
4	GD	53	91
4	TT	103	221
5	SFD	53	69
5	FFD	53	50
5	GD	81	84
5	TT	115	164
6	SFD	44	73
6	FFD	44	75
6	GD	54	103
6	TT	87	217

movement metric, high-frequency tokens had lower average durations than low-frequency tokens. This advantage to high-frequency tokens is not consistent among the first five modules.

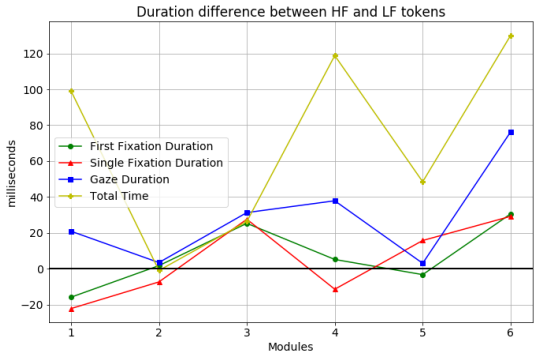

Fig. 3. Novice low-frequency mean duration minus mean high-frequency durations, showing the average advantage of high-frequency tokens in each learning module.

We now focus on the difference in duration between high- and low-frequency tokens. Figure 3 presents the mean low-frequency durations minus mean high-frequency durations for each learning module. It shows how much longer low-frequency tokens take to process, when compared to high-frequency tokens in the same learning module. The bigger the value in milliseconds, the greater the benefit from the frequency effect, and if the difference is zero it means that there is no difference between high- and low-frequency tokens. Figure 3 shows a positive trend between modules 1 and modules 6, suggesting an increase in the frequency effect over time. Most notably, in module 1 novices show an inverted frequency effect where low-frequency tokens took less time than high-frequency tokens, and module 6 shows an advantage for high-frequency tokens in all metrics.

A non-parametric Friedman test of differences among repeated measures is used and renders a Chi-square value of 12.42 which is significant p = 0.029. Friedman test is the appropriate test here as we have repeated measures and non-parametric data. Post-hoc pairwise comparisons using Dunn's test indicated that module 1 scores are observed to be significantly different from those of module 6 (p = 0.040, Cohen's d = 0.88). The same test indicates that module 2 scores are observed to be significantly different from those of module 6 (p = 0.009, Cohen's d = 1.98). No other differences are statistically significant.

The statistical tests suggest that there is a significant difference between the frequency effect measured in module 1 and module 6. To examine this difference further, Wilcoxon test is used to determine the eye movement metrics that show a significant difference. Wilcoxon signed-rank test indicates that Gaze Duration and Total Time are statistically significant with ($p = 0.027$) and ($p = 0.046$) respectively. Single Fixation Duration and First Fixation Duration are not significant.

To summarize, the results indicate statistically significant differences between the first two modules and module 6 in the affect of token frequency on Gaze Duration and Total Time.

D. RQ4: Does token length influence eye movement in reading source code?

Token length is a direct measure of the number of characters in a token. Word length effect refers to the difference in reading time in relation to word length, as longer words in number of characters take longer fixation durations [16, 17]. Word length effect comes from visual acuity as processing more characters that are distributed further from the center of the fixation takes more time than a shorter word [16].

Starting with experts, Figure 4 shows mean expert eye movement durations over tokens ranging in length from two to 10 characters (i.e., letters, dots, underscores). The figure shows that expert Single Fixation Duration and First Fixation Duration appear to be not influenced by token length. Yet, Gaze duration and Total Time appear to increase with the increase in characters. A non-parametric Friedman test of differences among Total Time repeated measures is done and renders a Chi-square value of 12.78 that is not significant p = 0.11 (p <0.05).

For novices, Figure 5 shows mean eye movement durations over tokens ranging in length from two to 10 characters. The figure shows that expert Single Fixation Duration and First Fixation Duration appear to be not influenced by token length. Yet, Gaze duration and Total Time appear to increase with the increase in letter-spaces, in a similar manner to experts.

A non-parametric Friedman test of differences among novice Total Time repeated measures used and renders a

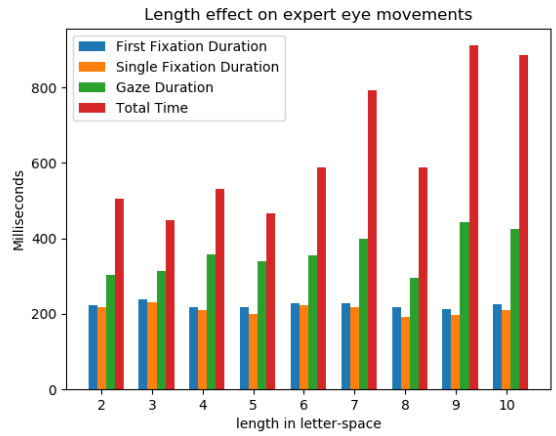

Fig. 4. Expert mean eye movement durations per-token-length in letter-spaces.

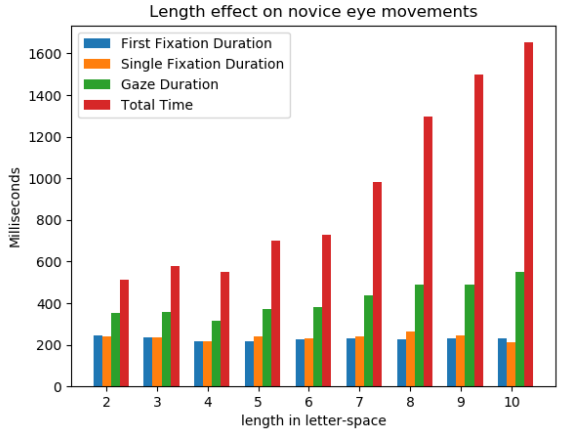

Fig. 5. Novice mean eye movement durations per-token-length in characters.

Chi-square value of 16.53 which is significant $p = 0.035$. Post-hoc pairwise comparisons using Dunn's test indicated that eye movement durations over tokens with three letter spaces in length are observed to be significantly different from those of length ten ($p = 0.045$, Cohen's $d = 1.36$). No other differences are statistically significant. Bonferroni correction is used with post-hoc Dunn's test to avoid Type-1 errors. Bonferroni correction compensates for the increased probability of detecting a false positive by accounting for number of pairwise tests in the significance threshold.

To summarize, our results suggest that token length influences eye movement duration in reading source code. This influence is seen in longer eye movement duration over tokens with more characters. This difference was only statistically significant in the eye movement of novice programmers.

E. RQ5: Can we distinguish novices from experts based solely on eye movement?

To answer this question we compare the eye movement of novices to experts based on our 8 metrics and validate the comparison through statistical testing. Table VII gives the mean eye movement metrics for novices and experts normalized (by division with prefix "n_") and probability metrics. Starting with duration metrics it appears that novices mean duration is either equal to experts (i.e. n_First Fixation Duration) or longer than experts (i.e. n_Single Fixation Duration, n_Gaze Duration, and n_Total Time). It is important to note that the durations are normalized by division, which means that a durations represent processing time per-character. For example the difference between novices and experts in n_Single Fixation Duration of 8 milliseconds will translate to a 40 millisecond difference in a 5 character token. When these differences are verified statistically using Kruskal-Wallis statistical test, n_Single Fixation Duration and n_Total Time are significantly different between novices and experts.

TABLE VII
MEAN EYE MOVEMENT METRICS FOR NOVICES AND EXPERTS NORMALIZED (BY DIVISION WITH PREFIX "N_") AND PROBABILITY METRICS. BOLD WHEN GROUPS ARE DIFFERENT ON A STATISTICALLY SIGNIFICANT LEVEL ($p < 0.05$).

Metric	Novice	Expert	p	Effect Size
n_Single Fixation Duration	48	41	**0.022**	0.179
n_First Fixation Duration	64	64	0.992	
n_Gaze Duration	97	92	0.107	
n_Total Time	159	138	**7e-05**	0.166
Fixation Probability	0.78	0.71	**0.008**	0.286
Prob. of One Fixation	0.28	0.32	0.531	
Prob. of Two Fixations	0.49	0.39	**6e-05**	0.308
Prob. of Skipping	0.21	0.28	**0.008**	-0.286

Moving to probability metrics, Table VII shows a comparison of mean probabilities between novices and experts. The values indicate that novices are more likely to make two or more fixations on the same token (Probability of Two Fixations - Pr2), and also experts are more likely to skip a token (Probability of Skipping). When these results are examined statistically through Kruskal-Wallis test the results show significant difference between novices and experts in Fixation Probability, Probability of Two Fixations and Probability of Skipping (Fixation Probability and Probability of Skipping complement each other as probability of making a fixation and a probability of skipping). Hence, we find statistically significant differences in eye movement between novices and experts in two duration metrics and two probability metrics.

The differences between the eye movement of novices and experts in specific metrics motivates the use of machine learning models to classify eye movements over tokens as coming from experts or novices. Such a model will validate our findings and form the bases to utilize the results of our research for practical tasks.

We construct a machine learning model to classify eye movement over tokens of code as coming from either novices or experienced programmers. We used Random Forest model

[40] for this task for a number of reasons. First, based on our previously presented statistical results we can make a distinction between novices and experts based on eye movement duration and probability of making more than one fixation. This leads us to use decision trees as a suitable classification model for the problem as each metric can represent a branch in the decision tree. For example if a token has a n_Total Time of 160 milliseconds and and Probability of Two Fixations (Pr2) of 0.5 then it is more likely that the tokens is coming from a novice. Second, Random Forests represent a class of ensemble learning that utilizes a combination of machine learning models to increase the classification accuracy. This is evident by the fact that Random Forests overcome the tendency of decision trees to over-fit training data [40]. Third, our data set consists of 1403 words with an asymmetrical distribution between expert and novice data (40/60) and Random Forest models work well with this type of distribution. Lastly, Random Forests can rank feature importance and this is very relevant while explaining the precise differences between the eye movement of novices and experts.

TABLE VIII
RANDOM FOREST MODEL PARAMETERS

Parameter	**Value**
n_estimators	100
criterion	gini
max_depth	None (unlimited)
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0
max_features	auto: max_features=sqrt(n_features)
max_leaf_nodes	None (unlimited)
min_impurity_decrease	0
min_impurity_split	1e-7
bootstrap	True
oob_score	False
random_state	None
warm_start	False
class_weight	None

We use the same data from the experiment described earlier with the following features: n_Single Fixation Duration, n_First Fixation Duration, n_Gaze Duration, n_Total Time, Probability of One Fixation (Pr1), Probability of Two Fixations (Pr2), Fixation Probability, Probability of Skipping (PrS) (Frequency and length features are removed to base the classification solely on eye movement). We use min-max normalization for the data as most ML algorithms are sensitive to features that are not on the same scale and we split the data using stratified sampling into a 70% training set and 30% testing set. A Random Forest model [40] is used with 10-fold cross validation as implemented in [41]. The model parameters are presented in Table VIII.

Running the model with 10-fold cross validation results in a mean accuracy score of 72.3%. Although this is a good result, accuracy alone is not very meaningful in a classification task; therefore precision, recall, and F1-score are used to provide a better understanding of the performance of the model. The precision score of our model is 77%, recall score is 75.4%, and F1-score is 76.2%. These results show that the model can classify experts and novices based on their eye movement. Furthermore, looking at feature importance presented in Table IX we see that the most important feature in classifying novices from experts is n_First Fixation Duration followed by Probability of Skipping, n_Total Time, n_Single Fixation Duration, and n_Gaze Duration. This ranking is informative of how the eye movement of novices and experts differ.

TABLE IX
RANDOM FOREST MODEL FEATURE IMPORTANCE RANKING AFTER TRAINING.

Feature	Importance
n_First Fixation Duration	0.124
Probability of Skipping	0.122
n_Total Time	0.112
n_Single Fixation Duration	0.108
n_Gaze Duration	0.102

VI. THREATS TO VALIDITY

Internal validity: With regards to RQ3, as mentioned in [12] some novice participants dropped out of of the experiment after a few weeks resulting in fewer samples in the last modules. In addition, experts are recorded through a single eye tracking session unlike novices who are recorded 6 times. This is caused by the difficulty of recruiting experts and professional programmers. This results in comparing the six novice modules to a single recording of experts, which might create an experience effect with novices that is not possible for experts. The machine learning classification in RQ5 is intended as a proof of concept demonstration to highlight the potential use of our research. We have taken many steps such as cross validation and stratified sampling to ensure the generalizability of the model. Nonetheless, significant steps remain incomplete before a practical real-time IDE extension is able to distinguish the level of expertise of a user based on their eye movement.

Construct validity: Busjahn et al. [12] notes that not all novice participants completed all modules as some dropped out. This does not affect RQ1 as we focus on expert eye movement exclusively. At the same time, RQ2 and RQ4 focus on novices, but we pool tokens based on their frequency and length across the six modules into groups that we can test statistically. As for RQ3, we used tokens aggregated in each module into low- and high-frequency tokens, and normalization by division was used to collect the largest valid sample possible. With regards to the experimental steps in RQ1 and RQ2, trying to quantify the frequency effect exclusively is extremely difficult. Here, we normalize eye movement duration by length to focus only on the frequency effect. We used two normalization techniques to cover aspects of variation in the normalization technique that might influence the results. We focused on the frequency and length effects in this paper as they appear to be the most influential language factors on eye movement in natural text reading [16]. Nonetheless, other factors such as token predictability and the syntactic structures can influence eye movement [42, 43].

External validity: The tasks used in earlier modules were relatively simple but got progressively more realistic. The ex-

perts were a representative sample from industry and novices were true novices, making these results applicable to real life settings.

Conclusion validity: We use appropriate inferential statistical tests since our data was not normally distributed and had a skew larger than 3 standard deviations.

VII. Discussion

When comparing our results to the previous study that reported no frequency effect in reading source code [14], we recognize a few differences. In this paper, we include key words and identifiers, and we apply proven filtering and normalization techniques that are reported in key previous research [19, 20, 38]. At the same time, our results on length effect matched the previous study.

The results from RQ1 and RQ2 provide evidence to support the presence of the frequency effect in reading source code. This result does not agree with the previous study by Busjahn et al. [14] which found no influence of frequency on eye movement in reading source code. However, there are a few major differences in the way we approach the question: First, we apply two methods of normalization and filtering techniques that are drawn from foundational research on the frequency effect in natural language text [19, 20]. In addition, we include four duration metrics in our analysis, namely First Fixation Duration, Single Fixation Duration, Gaze Duration, and Total Time. Furthermore, we considered the frequency effect on all code tokens that are greater than a single character in length, while the previous study focused on keywords. Lastly, our statistical analysis shows statistically significant differences between low- and high-frequency tokens in both normalization techniques.

The results of RQ3 provide a deeper analysis of the acquisition of the frequency effect by novices, where the results indicate a statistically significant difference between the first two modules and the last module in Gaze Duration and Total Time. These results suggest that indeed the frequency effect can be acquired through experience similar to how it is acquired in reading natural text. These results call for a future study to record novice eye movement beyond six learning modules to see the long-term effects of exposure to programming and how novice eye movement changes through the transition to experts. These results impact how educators would develop instructional materials for novices.

With regards to RQ4, our results match those presented by Busjahn et al. [14], which indicates the presence of the length effect in the eye movement of novice and expert code readers. This result calls for further investigation of the learning modules in which the effect appears. One hypothesis is that the length effect appears from the first module, since the length effect is a result of visual processing and acuity where larger stimulus requires more time to process. On the other hand, another hypothesis that the length effect is an acquired skill similar to the frequency effect.

Reflecting on the frequency and length effects from an eye movement control perspective, we can ask: Why are there frequency and length effects in the first place? And why do they appear in reading source code? The most prominent natural text eye movement control model, namely E-Z reader [38], may provide some insight in answering these questions. The model describes the length effect as a result of visual processing where a longer word requires more time to process as it involves encoding a more complex visual stimulus. As for the frequency effect, which influences lexical access where a stimuli that is more frequent is retrieved faster. This is possibly a result of the brain forming more synapses between neurons that are frequently activated together.

The results of RQ1-RQ4 leave us with implications to educators and practitioners. The presence of the frequency effect in reading source code means that exposure to consistent coding style facilitates reading and ultimately comprehension. When programming language elements are presented consistently their frequency increases, and thus become more predictable to the reader. Simultaneously, the presence of word length effect shows that longer identifier names take longer to process. Therefore, shorter identifier names are potentially easier and faster to read (without sacrificing a meaningful name). In the context of coding best practices, our results seem to uncover the cognitive bases for good coding style.

Regarding RQ5, the use of token level analyses of source code reading opens the door for interesting discussions on the influence of coding style and structure on source code reading and comprehension. Our intention behind the classification between novices and experts based on their eye movement is to demonstrate a potential practical use for our work. We envision an Integrated Development Environment (IDE) that is able to detect the expertise of a programmer and adjust its environment and recommendations accordingly. Perhaps such a system is able to introduce learners to IDE tools gradually in a way that matches their developing skill level.

VIII. Conclusions and Future Work

The paper analyzes an existing longitudinal data set of source code eye movements of novice and experienced Java programmers. The results present evidence in support of the presence of the frequency and length effects in reading source code. In addition, we present evidence of the acquisition of the frequency and length effects by novices over the duration of a Java course. Finally, we demonstrate how eye movement during reading source code can be used to classify novices and experts using fixation duration derived features. The results presented in this paper motivate further investigation into the linguistic factors, such as predictability and token type, that influence eye movement in reading source code. This could be useful in building a model of eye movement control that may be able to predict when and where the eyes move across a line of code. Eye movement models can give us a baseline for typical source code reading behaviour, which can be compared to novices or experts to determine their programming skill level. In addition, this research leads us to question the ways in which programming students leverage their natural language ability in reading source code.

References

[1] B. Shneiderman and R. Mayer, "Syntactic/semantic interactions in programmer behavior: A model and experimental results," *International Journal of Computer & Information Sciences*, vol. 8, no. 3, pp. 219–238, 1979.

[2] A. Von Mayrhauser and A. M. Vans, "Program comprehension during software maintenance and evolution," *Computer*, vol. 28, no. 8, pp. 44–55, 1995.

[3] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke, "On the comprehension of program comprehension," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 23, no. 4, pp. 1–37, 2014.

[4] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 2007, pp. 344–353.

[5] G. C. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the elipse ide?" *IEEE software*, vol. 23, no. 4, pp. 76–83, 2006.

[6] A. Von Mayrhauser, A. M. Vans, and A. E. Howe, "Program understanding behaviour during enhancement of large-scale software," *Journal of Software Maintenance: Research and Practice*, vol. 9, no. 5, pp. 299–327, 1997.

[7] T. A. Standish, "An essay on software reuse," *IEEE Transactions on Software Engineering*, no. 5, pp. 494–497, 1984.

[8] R. Tiarks, "What maintenance programmers really do: An observational study," in *Workshop on Software Reengineering*. Citeseer, 2011, pp. 36–37.

[9] A. Von Mayrhauser and A. M. Vans, "From program comprehension to tool requirements for an industrial environment," in *[1993] IEEE Second Workshop on Program Comprehension*. IEEE, 1993, pp. 78–86.

[10] B. Barry *et al.*, "Software engineering economics," *New York*, vol. 197, 1981.

[11] J. Siegmund and J. Schumann, "Confounding parameters on program comprehension: a literature survey," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1159–1192, 2015.

[12] T. Busjahn, R. Bednarik, A. Begel, M. Crosby, J. H. Paterson, C. Schulte, B. Sharif, and S. Tamm, "Eye movements in code reading: Relaxing the linear order," in *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, 2015, pp. 255–265.

[13] B. Liblit, A. Begel, and E. Sweetser, "Cognitive perspectives on the role of naming in computer programs." in *PPIG*. Citeseer, 2006, p. 11.

[14] T. Busjahn, R. Bednarik, and C. Schulte, "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*. ACM, 2014, pp. 335–338.

[15] C. Schulte, T. Clear, A. Taherkhani, T. Busjahn, and J. H. Paterson, "An introduction to program comprehension for computer science educators," in *Proceedings of the 2010 ITiCSE working group reports*. ACM, 2010, pp. 65–86.

[16] K. Rayner, "Eye movements in reading and information processing: 20 years of research." *Psychological bulletin*, vol. 124, no. 3, p. 372, 1998.

[17] R. Kliegl, E. Grabner, M. Rolfs, and R. Engbert, "Length, frequency, and predictability effects of words on eye movements in reading," *European journal of cognitive psychology*, vol. 16, no. 1-2, pp. 262–284, 2004.

[18] K. Rayner and S. A. Duffy, "Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity," *Memory & cognition*, vol. 14, no. 3, pp. 191–201, 1986.

[19] G. E. Raney and K. Rayner, "Word frequency effects and eye movements during two readings of a text." *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, vol. 49, no. 2, p. 151, 1995.

[20] K. Rayner and G. E. Raney, "Eye movement control in reading and visual search: Effects of word frequency," *Psychonomic Bulletin & Review*, vol. 3, no. 2, pp. 245–248, 1996.

[21] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "A survey on the usage of eye-tracking in computer programming," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, p. 5, 2018.

[22] 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*, vol. 67, pp. 79–107, 2015.

[23] M. E. Crosby and J. Stelovsky, "How do we read algorithms? a case study," *Computer*, vol. 23, no. 1, pp. 25–35, 1990.

[24] T. Busjahn, C. Schulte, and A. Busjahn, "Analysis of code reading to gain more insight in program comprehension," in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, 2011, pp. 1–9.

[25] M. E. Crosby, J. Scholtz, and S. Wiedenbeck, "The roles beacons play in comprehension for novice and expert programmers." in *PPIG*, 2002, p. 5.

[26] R. Bednarik, "Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations," *International Journal of Human-Computer Studies*, vol. 70, no. 2, pp. 143–155, 2012.

[27] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger, "Using psycho-physiological measures to assess task difficulty in software development," in *Proceedings of the 36th international conference on software engineering*. ACM, 2014, pp. 402–413.

[28] A. Kennedy and W. S. Murray, "The components of reading time: Eye movement patterns of good and poor readers," in *Eye Movements from Physiology to Cognition*. Elsevier, 1987, pp. 509–520.

[29] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical

research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[30] K. Rayner, J. Ashby, A. Pollatsek, and E. D. Reichle, "The effects of frequency and predictability on eye fixations in reading: Implications for the ez reader model." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 30, no. 4, p. 720, 2004.

[31] K. Rayner, E. D. Reichle, M. J. Stroud, C. C. Williams, and A. Pollatsek, "The effect of word frequency, word predictability, and font difficulty on the eye movements of young and older readers." *Psychology and aging*, vol. 21, no. 3, p. 448, 2006.

[32] R. P. Baayen, R H. and L. Gulikers. Linguistic data consortium. [Online]. Available: https://catalog.ldc.upenn.edu/LDC96L14

[33] D. A. Balota and J. I. Chumbley, "Are lexical decisions a good measure of lexical access? the role of word frequency in the neglected decision stage." *Journal of Experimental Psychology: Human perception and performance*, vol. 10, no. 3, p. 340, 1984.

[34] J. Ashby, K. Rayner, and C. Clifton, "Eye movements of highly skilled and average readers: Differential effects of frequency and predictability," *The Quarterly Journal of Experimental Psychology Section A*, vol. 58, no. 6, pp. 1065–1086, 2005.

[35] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 837–847.

[36] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, 2017.

[37] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

[38] E. D. Reichle, K. Rayner, and A. Pollatsek, "The ez reader model of eye-movement control in reading: Comparisons to other models," *Behavioral and brain sciences*, vol. 26, no. 4, pp. 445–476, 2003.

[39] N. Al Madi, "Modeling eye movement for the assessment of programming proficiency," Ph.D. dissertation, Kent State University, 2020.

[40] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[42] S. Frisson, K. Rayner, and M. J. Pickering, "Effects of contextual predictability and transitional probability on eye movements during reading." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 31, no. 5, p. 862, 2005.

[43] N. S. Al Madi and J. I. Khan, "Modeling part-of-speech and semantic significance effects on semantic construction during reading," in *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*. IEEE, 2019, pp. 162–165.