Studying Developer Reading Behavior on Stack Overflow during API Summarization Tasks

Jonathan A. Saddler
University of Nebraska - Lincoln
Lincoln, Nebraska USA 68588
jsaddle@cse.unl.edu

Cole S. Peterson
University of Nebraska - Lincoln
Lincoln, Nebraska USA 68588
cpeterso@cse.unl.edu

Sanjana Sama
Youngstown State University
Youngstown, Ohio USA 44555
ssama@student.ysu.edu

Shruthi Nagaraj, Olga Baysal
Carleton University
Ottawa, Ontario, Canada
{olga.baysal,shruthi.nagaraj}@carleton.ca

Latifa Guerrouj
École de technologie supérieure (ETS)
Montréal, Quebec, Canada
Latifa.Guerrouj@etsmtl.ca

Bonita Sharif
University of Nebraska - Lincoln
Lincoln, Nebraska USA 68588
bsharif@unl.edu

***Abstract*—Stack Overflow is commonly used by software developers to help solve problems they face while working on software tasks such as fixing bugs or building new features. Recent research has explored how the content of Stack Overflow posts affects attraction and how the reputation of users attracts more visitors. However, there is very little evidence on the effect that visual attractors and content quantity have on directing gaze toward parts of a post, and which parts hold the attention of a user longer. Moreover, little is known about how these attractors help developers (students and professionals) answer comprehension questions. This paper presents an eye tracking study on thirty developers constrained to reading only Stack Overflow posts while summarizing four open source methods or classes. Results indicate that on average paragraphs and code snippets were fixated upon most often and longest. When ranking pages by number of appearance of code blocks and paragraphs, we found that while the presence of more code blocks did not affect number of fixations, the presence of increasing numbers of plain text paragraphs significantly drove down the fixations on comments. SO posts that were looked at only by students had longer fixation times on code elements within the first ten fixations. We found that 16 developer summaries contained 5 or more meaningful terms from SO posts they viewed. We discuss how our observations of reading behavior could benefit how users structure their posts.**

***Index Terms*—API summarization, Stack Overflow, eye tracking, controlled experiment, reading behavior**

I. Introduction

When building software systems, developers need to understand code first before they make any changes. During maintenance, developers are often required to understand the code that others have written. While code comprehension is part of a developer's daily activities, the process can be time consuming due to the cognitive effort involved. Previous research [1, 2] has suggested that code summarization — the task of writing short, natural language descriptions of source code — can help developers quickly understand the source code and its various elements such as its classes and methods. A majority of summarization techniques are built around source code [2] and only a few on using informal documentation for the summarization of code elements [3].

However, developers often depend on other sources of informal documentation such as Stack Overflow (SO)[1], a popular Q&A forum for software developers to find solutions to development problems they face. Often, the source code (third party API usage) is not available to developers, leading them to seek help on SO posts. Stack Overflow's official website statistics suggest that 50 million developers and engineers visit its website every month.

The goal of this paper is to better understand how developers leverage Stack Overflow discussions for code comprehension and summarization tasks. To accomplish our research goal, we use eye tracking technology to collect eye gaze data from developers to learn how they read SO posts, In particular, we look to see what specific elements they read and use from posted questions, answers, and comments in their code comprehension and summarization tasks. We seek to answer the following research questions:

- RQ1: What elements do developers read on Stack Overflow pages when summarizing API elements?
- RQ2: What are the differences between how professional developers and non-professional (student) developers read Stack Overflow posts when summarizing API elements?
- RQ3: Do developer summaries reflect what was read on Stack Overflow?
- RQ4: How do specific attributes of the Stack Overflow posts (i.e., code block count, paragraph count) impact gaze behavior?

The first research question (RQ1) can help us understand how developers read SO pages and where they focus their attention during code summarization tasks. For RQ2, we compare two different groups of participants to understand whether there is a difference in the reading behavior of professional vs. non-professional developers during code summarization tasks. The findings of these research questions can help create guidelines for those writing SO posts to improve how they structure them. RQ3 offers insights into how relevant the participant's

[1]https://stackoverflow.com/

summaries are to the content of the Stack Overflow pages. The goal of RQ3 is to determine if the SO posts are used in the written summaries. RQ4 provides evidence on how specific attributes of SO content affect eye gaze and reading behavior.

To answer our research questions, we conduct a controlled experiment with 30 software developers, both professional and non-professionals (i.e., industry professionals and students in academia), to understand how they leverage SO discussions for code comprehension and summarization tasks.While they performed these tasks, we tracked elements they looked at on SO pages, the duration of their gaze on these elements, and what pages they navigated to while searching for answers to each question.

The paper offers the following contributions:

- A controlled experiment with 30 software developers to understand how they summarize API elements by leveraging Stack Overflow discussions.
- Use of eye tracking technology to understand developer behaviors, strategies and specific areas of focus when they read Stack Overflow pages.
- Evidence on what elements of informal documentation on Stack Overflow developers use to write summaries.

The paper is organized as follows. We present related work on code summarization and eye tracking in program comprehension in Section II. In Section III, we present the experimental design followed by the results in Section IV. Section V discusses the results and implications of those results. Threats to validity are presented in Section VI. We conclude with a summary of our findings and future work in Section VII.

II. Related Work

A. Code Summarization and Stack Overflow Studies

There have been several works in the area of code summarization, however, they mainly focus on the source code and its textual information when summarizing code [3–5]. For example, Moreno et al. [3] suggested a summarization approach based on the idea of Java source-code stereotypes. They engineered a set of algorithms to traverse code for facts about method structure in Java class source files, what variables are returned, how often they get returned, and how often all methods in a class share similar functions.

Recently, Guerrouj et al. [6] investigated the use of Stack Overflow for code summarization. They considered as context the information which surrounds the classes or methods trapped in Stack Overflow discussions. Treude and Robillard [7] proposed an approach to automatically augment API documentation with insights from Stack Overflow. The above two works are closest to the work presented in this paper.

From the human evaluation aspect, Ford et al. [8] learned that women users posted more often when they knew the other responder was of the same gender. Calefato et al. [9] found that longer question body lengths, and high uppercase-to-lowercase character ratio in the text, can be a deterrent to having a question get an answer marked acceptable by the original poster. Moreover, a far-more-persistent, independent variable such as the reputation score of the original poster had the strongest effect on whether a question was answered and promoted.

B. Eye Tracking Studies

A large body of research studies [10] in the field of program comprehension has leveraged eye tracking to have a better understanding on how programmers understand programs, and have given insights and detailed information about the cognitive processes of program comprehension strategies, including for example the effects of the way code is colored when rendered on screen [11]. Additionally, some researchers have used eye trackers to contrast movement and navigation through code [12–14].

Abid et al. [15] replicated the study by Rodeghero at al. [16] for code summarization tasks on large Java open source systems and found that developers tend to look at calls the most compared to method signatures (as previous reported in smaller snippets). This indicates that developers behave differently when tasked with realistic code compared to smaller snippets.

Turner et al. [17] investigated the effects of debugging across two different programming languages, Python and C++. Uwano et al. [18] found a *Scan* gaze pattern when developers read code with the goal of finding a defect. Saddler et al. [19] analyze reading behavior and comprehension of C++ source code by conducting both line and keyword level analysis on the transitions students make between C++ source code blocks for comprehension tasks.

One eye tracking study that is most relevant to ours is by Peterson et. al. [20] , which studied information seeking behavior of developers using Stack Overflow. They found developers focus on code elements the most and show a transition matrix of the most common regions on a page a developer searches through. In other work [21], the authors focused on explaining how developers view source code visually via radial transition graphs - this study did not use Stack Overflow.

We share with the above-mentioned research the idea that code summarization is a complex task and that automating it can help guide developers during their work. Unlike previous works, we focus on examining Stack Overflow and its parts (*e.g.*, comment regions on a Stack Overflow page, scores, code snippets, etc.) that can be vital to enhance code summarization. Our work differs in that we conduct an eye tracking study to understand reading behavior which has not been done in prior work. Unlike previous works that have mainly leveraged code and/or its comments to summarize code elements, we have investigated how developers summarize code elements discussed in informal documentation, *i.e.*, Stack Overflow.

III. Experimental Design

We now outline the experimental design including the tasks, participants, and instrumentation of the study.

A. Study Overview

In this study, each participant was asked to provide a summary of four API elements, two methods and two classes. Table I presents the details of API elements from four open source Java projects: Eclipse, JMeter, Tomcat and NetBeans. Each API element corresponds to one summarization task and was well researched to ensure that developers actually discuss these APIs on Stack Overflow.

Each task (i.e., API element to be summarized) included a link to a Stack Overflow search page mentioning a specific API element as the search query. Participants could select posts in this search query and search additional queries. The tasks (2 classes and 2 methods) were randomly assigned to participants to ensure that each API element was equally distributed. During the study, participants had access to a browser, a text file where the questions and tasks were presented, and a form to report API summaries. They were not shown the codebase itself.

Selection of API elements. When selecting APIs we wanted to make sure they were not too self explanatory nor too complex for comprehension. The selected API elements have three levels of comprehension complexity or difficulty. Some API elements were meant to be easy to understand, while others might require more time and information for participants to thoroughly understand and summarize. Each task sequence provided to a participant was designed to include API elements of various difficulty levels, (no one sequence has all four easy API elements or all four difficult to comprehend API elements.) Once the APIs were finalized, we made eight main sequences to use in the experiment as part of a larger study. Each sequence included a combination of easy, moderate and difficult API elements. Each had a total of four different API elements, and one of these was given to each participant for the purpose of this study, containing a URL to a Stack Overflow discussion. The assignment of these sequences to participants was completely randomized.

B. Areas of Interest (AOI) on a Stack Overflow Page

Stack Overflow pages can be broken down into regions. We label these regions based on their usefulness in aiding a user toward a solution. Figure 1 presents a sample Stack Overflow post with a label for different regions. Every post includes a *title* and *question*. The *title* is typically a brief summary of the question being asked, while the *question* is a textual description that provides more details on the problem the developer is facing.

The question given on each page can be followed by zero or more *answers* where users leave textual responses proposing a solution to the original poster. Stack Overflow users can also respond to the original question via *question comments* posted below the question to gather more information from the poster or clarify the question. Users may also leave *answer comments* which appear directly below a specific answer that was previously posted. *Tags* are short labels that appear next to questions that help identify topics of the question. *Vote counts* appear on every page next to the question, answers, and some comments and highlight whether other users find the content useful or appropriate.

We consider the seven above-mentioned regions of Stack Overflow pages to be the targeted areas of interest in our analysis. Other regions of the post such as advertisements, account navigation, page navigation, and search bars are excluded from our analysis.

C. Dependent Variables

The dependent variables used in our analysis are as follows:

- **Fixation Count** - the number of times a participant fixated upon an area of interest (AOI).
- **Fixation Duration** - the duration of all fixations of a participant on an area of interest.
- **Summary Accuracy** - the accuracy of the participant's summaries when compared to a summary from an oracle.
- **Summary Relevance** - the sum of words from a participant's summary that match words found in a Stack Overflow page
- **Vertical Early Percentage** - percentage of fixations made to an AOI that is vertically earlier in the page than the previous fixation.
- **Vertical Later Percentage** - percentage of fixations made to an AOI that is vertically later in the page than the previous fixation.
- **Regression Rate** - percentage of transitions back to an AOI after the participant transitions out of it and revisits it again.

The *summary accuracy* metric was calculated using several human annotators' ratings against an oracle. The *summary relevance* was automatically calculated using several ad-hoc scripts. The rest of the metrics were based on eye movement data collected while participants read the posts.

By tracing each webpage visited, we were able to later collect the text of certain page features (questions / answers / comments) using an ad-hoc parser, and later count how often fixations reached these. We downloaded pages via the Stack Exchange printing service[2], using its ability to render text verbatim from any Stack Overflow page we gave it to help simplify the DOM for quick and easy parsing [22].

D. Oracle Summaries

The summaries produced by the participants were verified against an oracle of summaries generated by human annotators. All eight API elements were summarized in the oracle. Based on the information available on the web (both formal and informal) and after analyzing API usage in source code, three independent human annotators (among the list of co-authors on this paper) formed an oracle of summaries. The three final sets of summaries from each annotator were further discussed and combined to form one final oracle of API summaries. The oracle summaries contain comprehensive summaries for each API element and are used to rate the quality and correctness of the summaries produced by the participants. To answer fully

[2]http://www.stackprinter.com/

TABLE I: Summarization tasks and characteristics of API elements.

Task ID	Project	Version	API element	Fully qualified name	#LOC	#SO posts
T1	Eclipse	4.2	method	`org.eclipse.core.databinding.binding.dispose`	18	42
T2	Eclipse	4.2	class	`org.eclipse.swt.SWTError`	34	56
T3	JMeter	3.2	method	`org.apache.jmeter.Jmeter.convertSubTree`	51	1
T4	JMeter	3.2	class	`org.apache.jmeter.samplers.SampleResult`	721	196
T5	Tomcat	7	method	`org.apache.catalina.realm.JDBCRealm.getRoles`	42	2
T6	Tomcat	7	class	`org.apache.catalina.valves.ValveBase`	133	39
T7	NetBeans	7.4	method	`org.netbeans.api.progress.ProgressUtils.runOffEventDispatchThread`	14	1
T8	NetBeans	7.4	class	`org.openide.nodes.CHildFactory.Detachable`	70	1

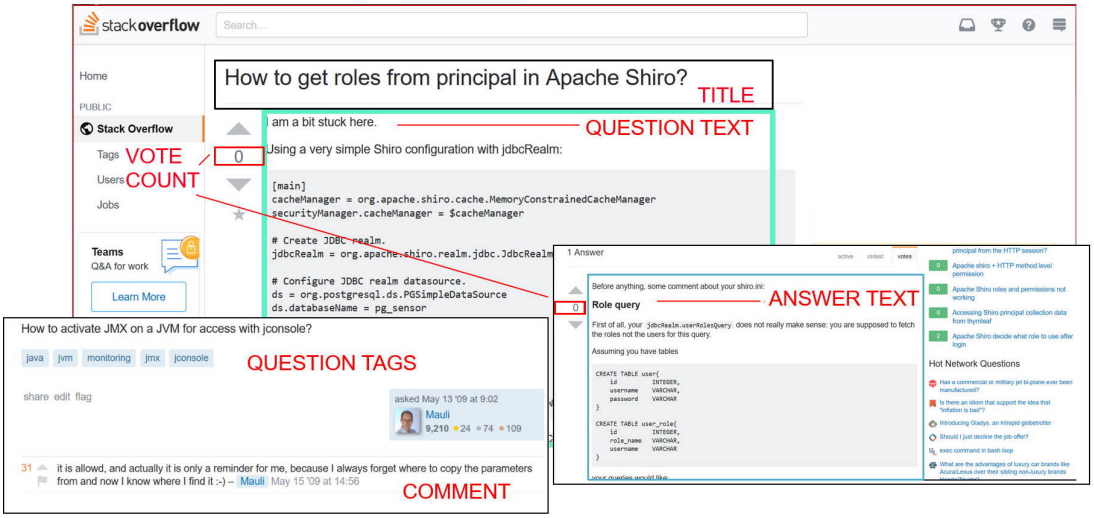

Fig. 1: Areas of interest (AOIs) on a SO page included in our analysis.

or partially correct, the participant had to fully or partially describe all elements stated in that API's oracle summary. The annotators were graduate research students pursuing PhD and Master's degrees in the field of computer science and had prior experience with Java APIs.

As an example, the oracle summary for the SWTError class was "A class representing an error internal to the SWT module. It contains a integer code and a throwable object. Indicates that an internal error occurred in SWT, displaying the error code and a description of the problem". Omitting some but not all information from the oracle will reduce a participant's score. One of the fully correct answers given by a participant was "The class is an exception class for SWT in eclipse, the exceptions are used for widgets and dialogs (for example, you can not have two file selection dialogs open at once), or otherwise have too many widgets open on a single display". A partially correct answer was "org.eclipse.swt.SWTError is a display method used for displaying widgets to your screen. Each widget seems to be a get function".

E. Eye Tracking Environment and Apparatus

We used the Tobii X60 eye tracker to collect eye tracking data on Stack Overflow pages within the web browser in Eclipse. We used iTrace [23], an eye tracking infrastructure that supports collection of fine-grained gaze data in Integrated Development Environments (IDEs) and browsers such as Chrome. The plugin has been effectively used for determining feasibility of deriving traceability links between code and bug reports using gaze [24] and for large realistic program comprehension [14] studies. iTrace is able to automatically map the eye gazes to semantically meaningful elements on Stack Overflow (and source code). We ran the Olsson fixation filter [25] on raw gazes to generate fixations of gazes that were 60 milliseconds or longer and were dispersed by less than 35px.

F. Participants

Thirty participants, drawn from a pool of two university campuses, participated in our study (26 males and 4 females, aged 18–35). Participants were recruited via email once research ethics protocol approval was obtained. All reported they had completed a Bachelor's degree program or higher, while 53% reported they were enrolled in or completed a Master's degree program or higher. 17 participants self identified as only holding an occupation of student, leaving 13 more participants who self-identified as affiliated with industry or as an academic faculty, and who also did not identify as a student.

When ranking their experience with the Java programming language on a scale from poor to excellent (coded 1-5, 1 - "Poor", 5 - "Excellent"), 70% rated themselves 2 and lower, while 30% (9) rated themselves 3 or higher. We asked about open source software they had been exposed to and found 21

of them were already familiar with the interface of Eclipse IDE, and 15 familiar with NetBeans. None of the participants developed code for the subject systems.

G. Study Instrumentation

On the day of the study, the participants came into the research lab and first filled out a pre-questionnaire that collected their demographic data. Next, each participant was asked to summarize in their own words (without any restriction on the length) two classes and two methods from our set of 4 open source projects. The API elements were arranged in eight randomly assigned task sequences. The comprehension context was to explain the purpose and usage of the API elements, i.e., methods and classes. Since summarization requires a fair amount of mental activity, we used four API elements to limit participants' fatigue effect. This study was conducted as part of a broader study having each participant use varied context resources to solve summarization problems. A subset of thirty responses from this larger study was set aside for use in the analysis we present in this paper.

The participants had access to a *web browser*, the text file where all questions and tasks were presented, and a response file where they typed summaries of the class/method. All of these elements were present within the Eclipse IDE. Participants were provided with a link to SO posts mentioning the API element. Tasks were randomly assigned to participants while making sure that each API element had an equal distribution among participants.

Eye tracker calibration was done at the start of each summarization task. A separate eye tracking session was made for each summarization task. At the end of the study, participants filled out a post-questionnaire that helped us gain insights and feedback about the collected data. The study took one hour, on average, for each participant. A remuneration of a 20 USD gift card was offered to every participant as compensation for completing the study.

H. Verifiability

A complete replication package on tasks and results is provided at http://seresl.unl.edu/SANER2020.

IV. Experimental Results

We analyzed 30 eye-movement data sessions, along with 28 API summaries across 29 pages of Stack Overflow content. Summaries from two participants were discarded due to grading errors.

A. RQ1: What elements do developers read on Stack Overflow pages when summarizing API elements?

Most fixations were on question code and question main text, and with our precise tracking, we could trace 29% of all fixations to question paragraphs, and 16% to those in answers. Figures 2 and 3 help show participants spent nearly 45% of their session reading plain text paragraphs, and nearly 50% of their fixations on code blocks within a page, but less of their time fixating (40%) on that code.

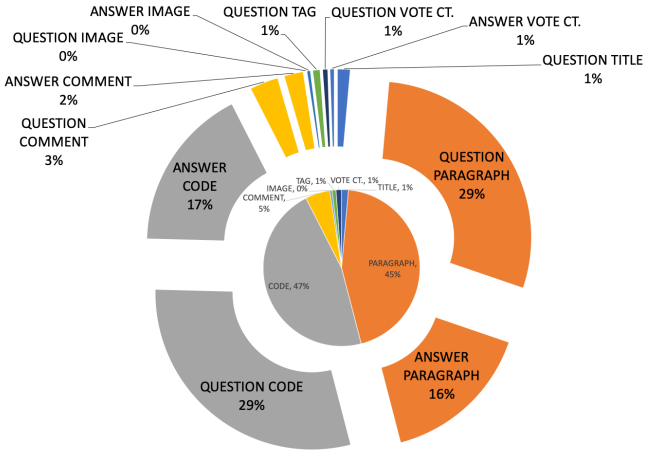

Fig. 2: Fixation counts by region.

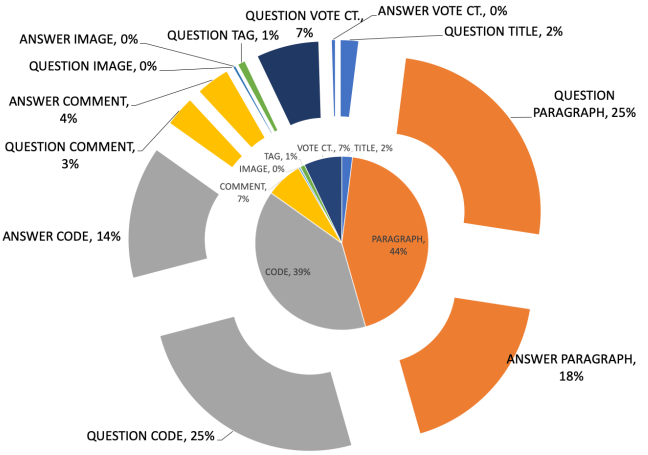

Fig. 3: Fixation duration by region.

We also see in Figure 3 that the ratio of fixation duration spent on question paragraphs as opposed to answer paragraphs is close to 3:5 (3.6 seconds on question paragraphs to every 5 seconds on answer paragraphs), while *question* code gets almost twice as many visits as *answer* code, a helpful insight from our fine grained analysis. 12 of 30 participants were tasked with answering a question related to either Eclipse or NetBeans. Eclipse-familiar participants spent 20 seconds less time (23s v. 42s) on average looking at paragraphs, but interestingly, spent less time fixating on code (17s v. 21s).

Looking at navigation between and inside AOIs, we see that most transitions were made to the same AOI with an average vertical early percentage (transitions made to a AOI vertically earlier in the SO page) of 18.44% and an average vertical later percentage of 22.04%. We also found that participants had a regression rate (transitions to a different AOI before coming back to an AOI that the participant had already visited) of 45.54% on average. However, not all AOI types had the same regression behavior. AOI's other than code, paragraphs, and

TABLE II: Mean fixation count and mean fixation duration on AOIs (PAR=paragraph, COD=code block., ALL=all regions)

	Fixation Count			Fixation Duration		
	PAR	COD	ALL	PAR	COD	ALL
Professionals	40.7	32.0	82.8	29.0s	19.0s	64.5s
Students	28.6	37.9	70.7	20.2s	23.6s	47.8s
Eclipse: Not Fam. Asked Eclipse*	39.5	45.0	98.0	42.0s	16.9s	72.7s
Eclipse: Fam. Asked Eclipse	34.8	28.8	78.5	23.0s	21.0s	59.0s
NetBeans: Not Fam. Asked NetBeans*	27.0	40.5	70.5	18.9s	21.0s	90.7s
NetBeans: Fam Asked NetBeans	20.2	8.3	34.3	14.3s	14.6s	33.1s
Does not use SO	19.7	11.0	33.3	13.3s	13.7s	28.6s
Uses SO	35.4	38.0	80.7	25.2s	22.5s	58.0s

*Asked for Summary of Eclipse/Netbeans Method/Class.
Note that we were only interested in IDE familiarity.

TABLE III: Top fixation duration regions and task accuracy for students and professionals.

Fixation Duration	Student	Professional
Code Block	9 (64%)	3 (37%)
Para. Block	5 (36%)	3 (37%)
Comment Block	-	2 (25%)
Incorrect	5 (31%)	3 (25%)
Some Correct	11 (69%)	9 (75%)

imagery had small regression rates. For instance, titles had a regression rate of 7.14% which was lower than we expected.

RQ1 Findings: Developers spent most of their time reading the code portion of questions over comments, vote tallies, and titles when they were looking at the question part of the Stack Overflow page. When they were looking at the answer parts of the Stack Overflow page, participants were found to spend most of their time on answer text over comments and vote tallies. The most regressions were to question-region code.

B. RQ2: What are the differences between how professional developers and non-professional (student) developers read Stack Overflow posts when summarizing API elements?

In Table II, we show gaze fixation duration comparison of students and professionals on code regions across all questions. Professionals in our population visited code regions 18% more on average than students. It is also worth noting that, even though both professionals' and students' average fixation count on comments is less than 10 fixations per session (See Table III), professionals visited nearly 3 times as many comments on average than students did in any given session, and 2 of 8 professionals fixated on comments longer than on any other region (including code and paragraphs). This might indicate comments held special meaning to professionals.

Next, we investigate what pages (See Table IV) on SO the participants looked at and whether there was overlap between participants. On average, participants looked at approximately 2 SO pages per session. Professionals looked at an average of 2.31 SO pages during their tasks whereas students looked at 1.88 pages on average. In total, there were 29 unique SO pages that were visited during this study. Of these, 16 were

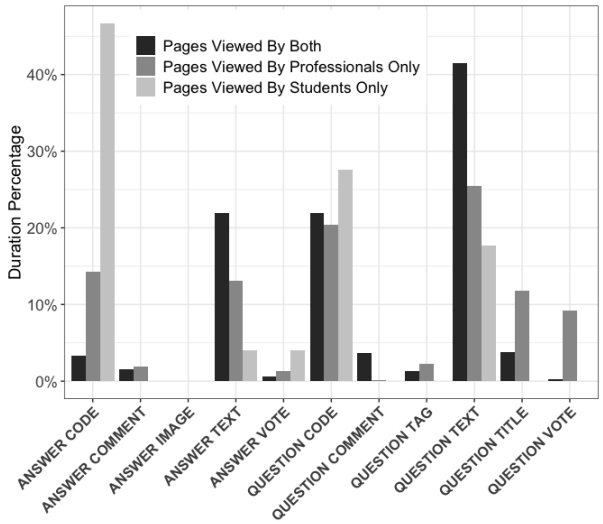

Fig. 4: Fixation duration distribution of the first 10 fixations on a SO page by areas of interest (AOIs) and experience.

viewed by at least one professional and one student, 8 viewed only by professionals, and 5 only by students.

We wanted to see how developers first glanced at these SO pages so we analyzed the first ten fixations a participant made on a page. The ten fixations threshold that was chosen is arbitrary, but we feel it is a good compromise between providing more context on the first fixations alone and on the general gaze distribution of the entire session. Looking at Figure 4, we can see the distribution of the first ten fixations using their duration for pages that were looked at by both students and professions, pages that were looked at only by professionals, and pages that were looked at only by students.

The first major difference we see between pages read by only students and pages read by only professionals is that student only pages had a higher percentage of fixation duration on *Answer Code* elements. We also see that *Answer Text* on pages read by only students has a smaller percentage of fixations than on pages read by only professionals. This pattern of student-only read pages containing more fixations and more fixation time on code elements and less on the text elements is seen again in the *Question Code* and *Question Text* elements. While this doesn't show that students read the code in a SO post more at the beginning of reading the post than professionals, it does hint at a difference in SO searching behavior between students and professionals.

Another notable difference in the gaze distribution between pages viewed by students only and pages viewed by professionals only is that very little time on student only pages was spent fixating on the *Question Title* or *Question Vote* elements and the professional only pages had more of the beginning fixations on these elements. This seems to indicate that most of the pages found by students only were not being looked at for these elements. One explanation is that professionals looked at the title and vote count of the posts at the beginning of

reading it to evaluate it or provide more context to the question, whereas students went directly to the content inside the post.

RQ2 Findings: On average, professionals fixate on more AOIs than students. In addition, they fixate on 18% more code regions than students. Pages that were looked at only by students had more fixation time on code elements and pages that were looked at only by professionals had more fixation time on text elements, the question title, and the question vote count.

C. RQ3: Do developer summaries reflect what was read on Stack Overflow?

We compared the summaries developers submitted with what content they found on a Stack Overflow page in answers, code snippets, and comments. Nearly 23,000 words unique to each page from SO pages visited were extracted from questions, answers, comments, and titles. We processed the words from each page, splitting words connected by periods and underscores, discarding this and other punctuation – including (, ' : ; " " [ ] ( )) – adding all SO words as tokens to a SO word-token dictionary, keyed to *only* retrieve words in a participant's log of SO pages they themselves visited. We then determine "summary relevance" by counting how often words from developer summaries appear in the text of Stack Overflow pages, following a lemmatization step via the following process:

1) We broke each word from the summaries into a word.
2) We used the premier NLTK tagger [26] packaged with NLTK 3.4.4, the averaged perceptron tagger, to tag each summary word with its proper part of speech. We did not re-train the packaged model.
3) We used the WordNet lemmatizer (part of NLTK) to shorten each tagged summary word to its root, removing any words matching parts of speech that were not nouns, adjectives, and verbs, adding the rest to a dictionary.
4) We matched elements from participant's SO-word-token entries with this lemmatized content from their summary.

A word can be lemmatized in different ways, and lemmatizers typically require complete sentences to work properly. Rather than train the averaged perceptron model for peak accuracy in terms of tagging the right part of speech, we first look for any way that a long word (like runner, or running, or runs) can be reduced (to say run), by lemmatizing each in the list of words in all three intended forms – noun, adjective, and verb – using the WordNet Lemmatizer. We then determined whether one of the three roots appears in the summary, by attempting to match each against the lemmas of complete sentences provided in a participant's summary.

If text from a participant's summary appears 5 or more times in SO word tokens from their session, the summary is considered to contain content relevant to SO. Otherwise, we consider the summary to be non-relevant to the discussion posted online. Minor conversions were necessary to make sure our parser could get proper part-of-speech information (such as first mapping all words to the Universal POS Tagset [27]).

It would be helpful to relate the results of whether accuracy was reflected in how well their summary matched words in the question. We present the outcome of attempts by a few participants to answer questions in Table V. Though participants had the opportunity to navigate as many pages as they wanted to in an untimed session, some participants only navigated up to a single page. The thoroughness of the response to the summary works to the participant's advantage. Fewer words included in the summary that are relevant, as in the case of a few shown in our example, can hurt their chances of getting the summary correct.

RQ3 Findings. With our threshold cutoff of 5 matching terms, we were able to discover that 53% of the time, (for 16 participants) summaries matched what was found in the content they viewed on Stack Overflow.

D. RQ4: How do specific attributes of the Stack Overflow posts (i.e., code block count, paragraph count) impact gaze behavior?

Our experiment was able to track the regions that participants visited the most and looked at the longest. The two most visited blocks for every participant were code blocks and plain text paragraph blocks. Next, we wanted to determine if the number of appearances of a feature such as a hyperlink, block count, or paragraph of text affects how long a participant spends looking at code blocks or paragraph blocks. If the number of paragraphs is above or below a threshold, it is reasonable to suspect this may affect how long a participant will spend look at the entire page if on a time budget.

We first divided questions into specific groups, by ordering them by feature count by dividing the counts of the AOI categories into quartiles. One of our initial ideas for groupings was "word count" (strings of contiguous letters) on the page. The word counts of questions ranged from 74 to 1,605 words (across titles, questions, answers, and comments). After creating quartiles, we created three groups to keep comparisons simple, one group G1 with counts less than the 1^{st} quartile, a second G2 with counts between the 1^{st} and 3^{rd} quartile, and the rest in the final group G3.

We also created similar quartile groups by creating a "head count" in each of appearances of our seven main page regions plus 5 Stack Overflow feature quirks that are used to emphasize text: *hyperlinks, bold words, blockquotes, plain text paragraphs,* and *code blocks*. We created some of these through manual analysis by looking for, counting, and recording appearances of these on every page. Looking at this data this way made it possible for us to narrow consideration to just two independent variables – appearances of paragraphs, and appearances of code blocks – focusing on whether these numbers have any impact on gaze behavior.

1) Comparing the Groups: We ran an Analysis of Variance test to detect whether differences in word counts in code blocks or question/answer paragraph answer text affect whether fixations would rise or fall on either code, comments, paragraphs, or titles. In the ANOVA, we looked at the total amount of fixations a participant made on one of these two regions whenever they visited this page in a session. We call this case a "*visit*.". Across 29 unique pages, participants were able to visit overlapping

TABLE IV: Questions visited by participants and their attributes.

PAR=plain text paragraphs, COD=code blocks, WC=Word count, all regions

num	SO-ID	Short name	Visited by	WC	PAR	COD
1	24414549	"DisableGUINetbeans"	P08, P13	1605	5	0
2	40970794	"JavaJMXDisabled"	P01, P11, P15	1335	7	6
3	2379688	"TestingJavaClassesJMeter"	P04	699	7	6
4	31539964	"DynamicallyRemoveNode"	P05, P16, P24	513	6	5
5	16940470	"JMeterMultipleSampleResult"	P21, P29	476	9	8
6	37306300	"CanTomcat7BeConfigured"	P14	434	13	11
7	19438438	"ReadRequestStreamMultiple"	P02, P19	372	7	0
8	23011284	"InvocationTargetExceptionSWTError"	P03, P25	355	6	12
9	27487610	"VisualStudio"	P12, P30	338	5	0
10	34333812	"CustomSSOValveBaseJBoss"	P28	330	4	2
11	17869755	"FindingTheIssueOfNoClass"	P14, P18, P28	328	5	6
12	46256355	"DataExtractionJMeter"	P21	306	6	4
13	12313939	"SessionTransactionHibernate"	P30	306	19	1
14	24803004	"RedirectPostRequest"	P19	281	7	3
15	24358707	"ExtendsValveBase"	P02, P14, P18, P28	268	6	5
16	1138450	"ImplementTomcatRealm"	P06, P23	261	13	11
17	6977864	"DisposingDataContext"	P07, P12, P22	227	7	2
18	46032100	"RolesPrincipalShiro"	P06, P10, P23	219	12	12
19	46367242	"HowToGetCookieManager"	P29	217	7	2
20	40259589	"SWTErrorNotImplementedEclipse"	P17	214	9	23
21	34236820	"VerifyHeaderBeforeRequest"	P14	206	10	3
22	6014767	"AutomaticallyUploadAfter"	P07, P12	162	4	0
23	19878257	"HowCanIGetRequestResponse"	P04	147	18	7
24	43594344	"JMeterCustomJavaSampler"	P21, P30	137	6	4
25	11959334	"NoClassDefFoundErrorSWTError"	P03, P17, P25	131	14	3
26	14768131	"SWTErrorNotImplementedMultiple"	P17, P25	127	6	7
27	856881	"HowToActivateJMX"	P11	96	38	41
28	23569777	"CantMakeCustomValveWork"	P14, P28	87	13	1
29	44160451	"JMeterAssertion"	P09	74	9	3

TABLE V: Selected participants, their answers, and words used from Stack Overflow

Part.	API Asked	C/R	Shortnames of All Visited Questions	Lemmatizer Matches from Summary	Summary Snippet
P03	Eclipse SWTError Class	C R	NoClassDefFoundErrorSWTError InvocationTargetExceptionSWTError	class, exception, SWT, eclipse, used, widgets, file, open, many	exception class for SWT in eclipse, exceptions are used for widgets and dialogs
P15	JMeter convertSubTree Method	C R	DisableGUINetbeans	Subtree, method, removes, disabled, elements, target	removes disabled elements and replaces with target subtree
P05	Netbeans Detachable Class	R	DynamicallyRemoveNode	Detachable, add, child, nodes, root	allows to add child nodes under the root nodes

Legend: C - Fully correct answer, R - Answer meets relevance threshold.

TABLE VI: Accuracy by OSS project in question

	OSS Project			
Accuracy	Eclipse	JMeter	Tomcat	NetBeans
Incorrect	0	2 (20%)	3 (42.9%)	3 (50%)
Some Correct	5 (100%)	8 (80%)	4 (57.1%)	3 (50%)

TABLE VII: Accuracy by API scope in question

	API Type	
Accuracy	Method	Class
Incorrect	5 (31.25%)	3 (25.0%)
Partially Correct	11 (68.75%)	9 (75.0%)

TABLE VIII: Effect of paragraph count on SO page fixations (COD=code block, TTL=title, COM=comment)

Group Pair	Mean Fix Count	min N_1,N_2	Mean Sq Error	HSD min dif	p
COD G1-G2	10.50/23.29	4	436.487	17.60	.483
COD G1-G3	10.50/18.45	4	436.487	17.60	.792
COD G2-G3	23.29/18.45	11	436.487	17.60	.783
TTL G1-G2	1.23/.37	13	.657	.61	.004
TTL G1-G3	1.23/.37	11	.657	.61	.007
TTL G2-G3	.37/.18	11	.657	.61	.780
COM G1-G2	5.90/1.46	10	10.226	2.72	.001
COM G1-G3	5.90/1.36	10	10.226	2.72	.006
COM G2-G3	1.46/1.36	11	10.226	2.72	.996

sets of these, as they were free to explore. When all is taken into account, 62 total *visits* were available to experiment with, along with 62 recordings of fixations to paragraphs, to code, and to the other main 7 regions of our analysis. This data is presented as part of our replication package.

The two ANOVAs focused on the three quartile-based groups for paragraphs and code blocks. After getting the means of each group in each category, we found an appropriate test for answering the question would be comparing the means using the Tukey Honest Significant Difference (HSD) minimum mean difference test [28]. This mean difference test uses results from an Anova F test to specify a unique value by which means must differ in order for the difference to count as significant. We present these comparisons and the Tukey HSD values below and in Tables VIII and IX.

TABLE IX: Effect of code block count on SO page fixations (COD=code block, TTL=title, COM=comment)

Group Pair	Mean Fix Ct	min N_1,N_2	Mean Sq Error	HSD min dif	p
COD G1-G2	14.22/20.19	9	422.442	17.72	.725
COD G1-G3	14.22/30.60	9	422.442	17.72	.203
COD G2-G3	20.19/30.60	10	422.442	17.72	.353
TTL G1-G2	.44/.32	9	.583	.64	.907
TTL G1-G3	.44/.40	9	.583	.64	.991
TTL G2-G3	.32/.40	10	.583	.64	.958
COM G1-G2	1.40/1.32	8	6.342	2.50	.983
COM G1-G3	1.32/1.14	7	6.342	2.50	.960
COM G2-G3	1.14/1.14	7	6.342	2.50	.986

2) How Adding More Paragraphs Affects Gaze: Paragraph totals we found in pages helped us divide pages into groups with less than 5 paragraphs (low), 6 to 12 paragraphs (medium) and 13 or more paragraphs (high). On average, the fixation counts on medium and high paragraph page comments were significantly less than for low ($p_{low-med} = .001, p_{low-high} = .006$). The minimum amount these means have to differ according to Tukey's HSD test is by 2.72 fixations to be significant.

3) How Adding More Code Affects Gaze: Quartiles on our counts of code blocks in our participants visited pages helped us divide pages into groups that had fewer than 2 code blocks (low), 3 to 9 code blocks (medium) and 10 or more code blocks (high). Considering only pages containing at least 1 block, increase in code blocks on a page does not make a significant change in mean fixation count (ANOVA $F_{code}(2, 47) = 1.602$, $p = .212$). Fixations on comments are not affected by number of code blocks, (ANOVA $F_{comment}(2, 47) = .038$, $p = .963$).

We need to point out that behavior of participants made it difficult to compare certain groups in our analysis. Nearly 70% of visits to pages (42/62) presented us with data where participants had fixated on titles for less than 60ms, and nearly 43% of visits (21/49 visits) resulted in comments not being fixated upon at all when they were present. These statistics helped explain why some group-wide distributions in our data were skewed toward 0. The skewness of inputs to our data sets ranged from -.222 to 2.8, from minor to moderate skewing. We found applying a square-root transformation to 9 of 18 of our groups was not enough to bring skewness for these down to less than an acceptable 0.8. We explain this as a threat to validity.

RQ4 Findings. Higher paragraph counts can have a negative impact on the amount of fixations that reach other parts of the page that contain text relevant to the question, in particular comments. Paragraphs might draw more interest from participants new to browsing pages on these APIs for the first time because other features appearing below them take up less space. Since comments take up less space, they might seem less relevant to the main material. This content might be perceived by newcomers as extra and unrelated to what will help answer the question.

V. Discussion

We conducted a controlled exploratory study where we tracked the gaze of 30 participants for a total of 77 minutes of session time while they looked at Stack Overflow pages, and localized 27.5 minutes of fixation data to areas of interest related to this study. On average, each participant spent 55.06 seconds browsing regions of interest in a session, (34.97 seconds for a question, 20.09 seconds for an answer).

Professional and Student Gaze Outcomes. We found that professionals looked at more AOIs than students and looked at code elements 18% more on average than students. There was no difference in the number of pages professionals and students in this sample read per session as they both read 2 pages per session on average. We found that SO pages that were viewed only by students had more fixation time on code elements within the first 10 fixations on the page. One possible explanation for this difference is that students may search for SO posts that contain more examples of code than professionals, or it may be that students are more drawn to the code inside of a SO post.

53% (16 of 30) of the participants passed our 5-word threshold test for whether the submitted summary was relevant to the questions that participants viewed, indicating a majority of participants relied on StackOverflow to inform their understanding. We did not learn where within these posts participants collected their text from most often, though we did find more plain text paragraphs that were present in the questions and answers, the fewer fixations other regions received on a page.

At least 50% of each sub-population of either Eclipse, JMeter, Tomcat or NetBeans respondees scored at least a partially correct answer to our comprehension questions. (See Table VI). 75% of these asked questions about classes from these API's scored partially correct or better, and 68.75% method-level API questions scored partially correct or better. (See Table VII).

We now discuss some implications of this work. Stack Overflow is a great resource for developers to learn more about programming and these results can help to improve the quality of questions and answers alike. We found that developers read the text and code of Stack Overflow posts. Comments, tags, and votes are not looked at nearly as often. If there is an important correction to an answer, the poster should consider editing the original answer in order to ensure that the correction is seen by the most people. Likewise, if there is an important clarification to a question, the poster should consider editing the question rather than adding the clarification as a comment. In addition, we found that students did not look at the title on the page within the first few fixations on the page and that all participants rarely fixate on the title after they had viewed it once. A poster might want to consider including the title's context inside the text of the question since developers will be more likely to read and navigate back to question text than the title.

VI. Threats to Validity

With respect to *internal validity*, the quality of summaries from participants could be affected by how much page content

a participant was able to view in their session. There are many factors that make guessing the size of the summary based on the content viewed difficult, and we could not control for this, given that we gave participants freedom to navigate Stack Overflow on their own. We believe the freedom we allowed participants to navigate throughout the site was necessary to mimic a realistic situation and allow them to fully and completely answer our comprehension questions without much foreknowledge of the API's. In RQ4, we downloaded pages visited by participants and recorded paragraph count, code region count, as well as blockquote, hyperlink, and bold region count, to see how they affected gaze. However, not all pages participants visited contained these elements. To ensure proper comparison between groups in our data, we removed from page-to-page comparisons, any page that contained 0 code blocks if we were comparing fixations on code blocks, or pages containing 0 comments if we were comparing fixations on comments.

With respect to *external validity*, the selection of the API elements and page attributes we chose to analyze out of the many available could affect the generalizability of our work. To keep the study realistic, we had the participants use Stack Overflow in a browser and used open source API elements (and not some toy application). For attributes analyzed, some were chosen because they appeared in other research (code and bold words), while others because they were the easiest to extract from HTML. We did not focus on ads or account information on SO pages, though they could be prominent attractors of gaze on Stack Overflow. We leave this as future work.

In terms of *construct validity*, we used a count of 5 terms to measure a binary representation of whether content from user's summaries was relevant. Plus, the threshold of 10 fixations used to compare the fixation distributions on SO pages read by only students, only professionals, or both is arbitrary, but this was done to help illustrate initial behavior. In addition, we chose to use lemmatization, when a potentially more accurate alternative for matching words could have been using thesaurus synonyms to match words. However, in this paper we present only the final step in our approach to this problem. Before settling on lemmatization, we tried much simpler methods such as regex matching on words and matching on word stems instead of whole words, but these led to a less robust matching system in the end, and would match words that were not parts of speech indicative of understanding the question. Implementing reducing all words to lemmas was an alternative that led to less bias to a particular grammatical choice made by the participant. We invite future work in this field to explore whether adding thesaurus results would take our relevance analysis a step further.

With respect to *conclusion validity*, during this study, to ensure ANOVA would give reliable results, we inspected whether our fixation count histograms followed normal distributions. We found two interesting things that prevented us from getting optimal ANOVA results - very few participants fixate on titles or comments in our pages, making their fixations harder to compare to other areas. This presented us with a choice to make to either apply stronger transformations to every single element in the data set, thereby "cleaning the data first", or to present the data as is. We chose the latter which may have had us miss some important mean differences because of the skew in the ANOVA inputs.

VII. Conclusions and Future Work

We studied how 30 students and professionals read Stack Overflow posts while they summarized API elements to answer API comprehension questions without direct access to the source code. After finding that plain text paragraphs along with code examples attract the greatest total fixation duration and fixation counts, we showed how gaze behavior of those familiar with SO or familiar with an API in their question differed greatly from those unfamiliar. Those familiar with Stack Overflow fixate much less often on code, paragraphs, and the page overall, and their gazes spent less time per page. Those familiar with an API they were asked to summarize spent about 25% less time on average fixating on code or paragraphs they came across, and spent nearly half the time fixating on the page overall as those unfamiliar with the API.

Though we did not see professionals and students differ much in terms of their accuracy or the direction of their transitions among blocks on pages or pages themselves, we saw differences in gaze behavior that depended on page content. Higher word counts in plain text paragraphs led to no change in fixations on paragraph content or code content, but rather significantly fewer fixations on comments. Plain text paragraphs are found everywhere throughout Stack Overflow questions, but we were able to unearth how the internal structure of a question may draw gaze activity to certain parts of a page.

As part of future work, we plan on re-conducting this study by providing Stack Overflow posts as well as the source code of the system. The more we enable access to more of the web, the more we can mimic a real developer workspace, and the more our research can allow us to see what resources developers voluntary pick while browsing the web to understand code, and how and why they choose to transition between them. In line with this concept, future work also includes opening up the participant's freedom to browse other relevant parts of the web to see how various resources of online information will improve their learning experience.

References

[1] G. Sridhara, E. Hill, L. Pollock, and K. Vijay-Shankar, "Towards automatically generating summary comments for java methods," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, New York, NY, USA, 2010, pp. 43–52.

[2] R. P. L. Buse and W. R. Weimer, "Automatically documenting program changes," in *Proceedings of the International Conference on Automated Software Engineering*, 2010, pp. 33–42.

[3] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Proceedings of the 2013 21st International Conference on Program Comprehension*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 23–32.

[4] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 22Nd International Conference on Program Comprehension*, ser. ICPC 2014. New York, NY, USA: ACM, 2014, pp. 279–290. [Online]. Available: http://doi.acm.org/10.1145/2597008.2597149

[5] R. P. Buse and W. R. Weimer, "Automatically documenting program changes," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1858996.1859005

[6] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 639–642.

[7] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 392–403.

[8] D. Ford, A. Harkins, and C. Parnin, "Someone like me: How does peer parity influence participation of women on stack overflow?" in *IEEE Symp. VL/HCC 2017, Raleigh, NC, USA*, 2017, pp. 239–243. [Online]. Available: https://doi.org/10.1109/VLHCC.2017.8103473

[9] F. Calefato, F. Lanubile, and N. Novielli, "How to ask for technical help? evidence-based guidelines for writing questions on stack overflow," *IST Journal*, vol. 94, pp. 186–207, 2018. [Online]. Available: https://doi.org/10.1016/j.infsof.2017.10.009

[10] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "A survey on the usage of eye-tracking in computer programming," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 5:1–5:58, Jan. 2018.

[11] T. Beelders and J.-P. du Plessis, "The influence of syntax highlighting on scanning and reading behaviour for source code," in *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, ser. SAICSIT '16. New York, NY, USA: ACM, 2016, pp. 5:1–5:10. [Online]. Available: http://doi.acm.org/10.1145/2987491.2987536

[12] P. Rodeghero and C. McMillan, "An empirical study on the patterns of eye movement during summarization tasks," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, vol. 00, Oct. 2015, pp. 1–10.

[13] S. Raina, L. Bernard, B. Taylor, and S. Kaza, "Using eye-tracking to investigate content skipping: A study on learning modules in cybersecurity," in *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, Sep. 2016, pp. 261–266.

[14] N. J. Abid, J. I. Maletic, and B. Sharif, "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*, ser. ETRA '19. New York, NY, USA: ACM, 2019, pp. 13:1–13:9. [Online]. Available: http://doi.acm.org/10.1145/3314111.3319834

[15] N. J. Abid, B. Sharif, N. Dragan, H. Alrasheed, and J. I. Maletic, "Developer reading behavior while summarizing java methods: Size and context matters," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 384–395. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00052

[16] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. K. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, 2014, pp. 390–401. [Online]. Available: https://doi.org/10.1145/2568225.2568247

[17] R. Turner, M. Falcone, B. Sharif, and A. Lazar, "An eye-tracking study assessing the comprehension of c++ and python source code." in *Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA 2014*. New York: ACM, 2014, pp. 231–234.

[18] H. Uwano, M. Nakamura, A. Monden, and K. ichi Matsumoto, "Analyzing individual performance of source code review using reviewers' eye movement," in *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications*, ser. ETRA '06. New York, NY, USA: ACM, 2006, pp. 133–140.

[19] J. Saddler, C. Peterson, P. Peachock, and B. Sharif, "Reading behavior and comprehension of c++ source code – a classroom study," in *Proceedings of the 21st International Conference on Human Computer Interaction*, ser. HCII 2019. Orlando, FL, USA: Springer, July 2019.

[20] C. S. Peterson, J. A. Saddler, N. M. Halavick, and B. Sharif, "A gaze-based exploratory study on the information seeking behavior of developers on stack overflow." in *CHI Extended Abstracts*. New York, NY, USA: ACM, 2019.

[21] C. S. Peterson, J. Saddler, T. Blascheck, and B. Sharif, "Visually analyzing students' gaze on c++ code snippets," in *Proceedings of the 6th International Workshop on Eye Movements in Programming*, ser. EMIP '19. Montreal, Quebec, Canada: IEEE, May 2019.

[22] "Stackprinter." [Online]. Available: http://www.stackprinter.com/

[23] D. T. Guarnera, C. A. Bryant, A. Mishra, J. I. Maletic, and B. Sharif, "itrace: eye tracking infrastructure for development environments," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, ETRA 2018, Warsaw, Poland, June 14-17, 2018*, 2018, pp. 105:1–105:3. [Online]. Available: https://doi.org/10.1145/3204493.3208343

[24] B. Sharif, J. Meinken, T. Shaffer, and H. H. Kagdi, "Eye movements in software traceability link recovery," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1063–1102, 2017. [Online]. Available: https://doi.org/10.1007/s10664-016-9486-9

[25] P. Olsson, "Real-time and offline filters for eye tracking," p. 42, 2007.

[26] S. Bird, E. Loper, and E. Klein, *Laura Lion and Gabrielle Giraffe and Carl Capybara*. London, England: O'Reilly Media Inc., 2009.

[27] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," *CoRR*, vol. abs/1104.2086, April 2011. [Online]. Available: http://arxiv.org/abs/1104.2086

[28] S. E. Maxwell and H. D. Delaney, *Designing experiments and analyzing data: A model comparison perspective*. Belmont, CA: Wadsworth Publishing Company, 1990.