How Developers Make Decisions When Choosing Issues and Reviewing Code: An Eye Tracking GitHub Study

Igor Wiese
Federal Technological University of Paraná (UTFPR)
Campo Maurao, Brazil
igor@utfpr.edu.br

Jasmine Boyer
University of Nebraska-Lincoln
Lincoln, Nebraska, USA
jasmineboy2@gmail.com

Ethan Rasgorshek
University of Nebraska-Lincoln
Lincoln, Nebraska, USA
erasgorshek2@huskers.unl.edu

Gustavo Pinto
Federal University of Pará (UFPA)
Belém, Brazil
gpinto@ufpa.br

Marco Aurelio Gerosa
Northern Arizona University
Flagstaff, Arizona, USA
Marco.Gerosa@nau.edu

Igor Steinmacher
Northern Arizona University
Flagstaff, Arizona, USA
Igor.Steinmacher@nau.edu

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

ABSTRACT

The paper presents a pilot eye-tracking study on how developers choose what issues to work on and how they perform code-reviewing tasks within the GitHub ecosystem. In this study, we recorded the eye movements of thirteen developers to understand what they look at on the GitHub interface to make decisions. They completed four tasks namely, ranking a list of open issues to work on, prioritizing pull requests, the likelihood of pull requests being accepted, and finally evaluating 25 diverse user profiles for pull request acceptance likelihood. Results suggest that the title, description, and labels are the most important information when developers choose the issue to work on and pull requests to review. The quality of the description and reproduction steps also influenced how the developer ranked an issue. The contribution heat map and repository language were relevant areas that attracted more attention when they looked at user profiles.

CCS CONCEPTS

• **Software and its engineering** → *Software maintenance tools*; • **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

eye tracking study, GitHub study, code review, social signals, open source systems

ETRA'25, May 26 – 29, 2025, Tokyo, Japan

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/3715669.3723108

ACM Reference Format:
Igor Wiese, Jasmine Boyer, Ethan Rasgorshek, Gustavo Pinto, Marco Aurelio Gerosa, Igor Steinmacher, and Bonita Sharif. 2025. How Developers Make Decisions When Choosing Issues and Reviewing Code: An Eye Tracking GitHub Study. In *2025 Symposium on Eye Tracking Research and Applications (ETRA '25), May 26–29, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3715669.3723108

1 INTRODUCTION

Many modern software development projects rely on *issue tracking systems* to organize tasks effectively. Issues may represent bug fixes, feature suggestions, enhancements, usage questions, or general project discussions. These projects frequently utilize distributed version control systems, allowing developers to maintain local copies of the project's reference repositories while working on various issues. Once developers complete work on an issue, they must submit (i.e., pull) their changes to the reference repository.

While some projects grant developers direct permission to pull changes, others require mandatory code reviews to ensure that contributions meet quality standards before merging. This review process often occurs through a pull request, where reviewers can assess changes, provide feedback, ask questions, and decide whether to accept the changes [Gousios et al. 2014]. Pull requests help maintain high software quality, reduce the risk of bugs, enforce coding conventions, and foster collaboration among development teams [Koehler Leman et al. 2020; Souza et al. 2022; Tsitoara 2020]. Consequently, the pull request model and code review have become integral components of modern software development practices across both technology sectors and open-source projects.[1]

Despite the benefits of using pull requests and issue trackers, they are a source of potential bottlenecks and challenges. From the newcomers' perspective, the literature shows that finding an issue to start is far from being an easy task, and may demotivate the newcomers and lead them to give up contributing [Steinmacher et al.

[1] https://github.com, https://gitlab.com, https://bitbucket.org

2015]. Looking at pull requests, the social dimension introduced in the pull-based approach may introduce bias related to the way the developers approach code reviews [Tsay et al. 2014]. Therefore, it becomes important to take a closer look at how developers choose issues to work, how they choose pull requests to review, and what they look at when deciding about the acceptance of a pull request.

This importance is evidenced by the high number of studies conducted on different aspects of issue tracker and pull requests over the past several years [Badampudi et al. 2019; Bertoncello et al. 2020; Rahman and Roy 2014; Souza et al. 2022]. Badampudi et al. [2019] mapped 873 papers related to code review to find several common topics about the process people have been researching. The study found that most papers analyze solutions, reviewer identification, and the impact/outcome of code reviews. [Rahman and Roy 2014] investigated 78,955 pull requests from over 78 different GitHub projects and found the number of developers on the project, number of forked projects, age of the project, technical issues, and applications domains all affect the success of pull requests. The above studies do not perform an in-depth analysis of the workflow involved in a pull request acceptance.

To bridge this gap, we present a preliminary eye tracking study to understand what developers look at in the process of making decisions on issue choice and code review. Unlike prior work, our study explores the entire workflow of deciding and prioritizing issues or pull requests in a realistic setting within the browser (not merely on snapshots that cannot be interacted with). In this context, our study aims to analyze how developers make decisions on modern social coding platforms based on their eye movement. To guide our research towards this goal, we designed the following research questions:

- RQ1: What pieces of information do developers use to rank issues and pull requests to work on?
- RQ2: What do developers look at when assessing a potential pull request?
- RQ3: What profile characteristics are more often assessed?

We answer these research questions by running an eye-tracking study where participants were required to perform four tasks related to ranking issues and pull requests, and evaluating the likelihood of accepting pull requests based on the pull request information (e.g., title, comments, code), meta-information (e.g., labels, dates) or the pull request author profiles. The results indicate that developers use more information than meta-information to choose issues and pull requests to work on – in terms of fixation time. Specifically, the most used element on issues was the title, and the most used elements on pull request reviews were comments and the code, as expected. When looking at the social aspect, we found that the username, technical aspects, and the reputation of pull-request authors play a role in the acceptance of pull-requests. Developers look at the contributor heat map, repositories they contributed to, and languages used on these repositories.

2 RELATED WORK

Huang et al. [2020] conducted a study of 37 participants including behavioral, eye-tracking, and medical imaging measurements. The results showed that men and women conduct code reviews differently. While men fixated more frequently, women spent significantly more time analyzing pull request messages and author pictures. The authors also reported a gender bias effect. When pull request author information changes, participants report seeing quality differences where none exist. Sharif et al. [2012] replicates the study from [Uwano et al. 2006] focusing on understanding how reviewers spot defects. The results indicate that the longer a reviewer spends in the initial scan, the quicker they find the defect. Conversely, if a reviewer does not spend sufficient time on the scan they are likely to find irrelevant lines thereby decreasing the performance of the review. Ford et al. used eye tracking glasses to see what parts of the pull request author's profile developers looked at intending to approve or reject the pull request [Ford et al. 2019]. They recruited 42 software engineers to perform two code reviews. The results showed that developers look at more than just the code being reviewed while performing pull requests. They also look at the author's profile, where all participants viewed at least one piece of information that could identify the demographic of the pull request author. These results indicate that the developer/author's profile (social signals) influences decisions on code contributions. Park and Sharif used eye tracking and sentiment analysis to see how existing tools compare against perceived developer sentiment in GitHub pull requests with emojis [Park and Sharif 2021]. The study also found that participants looked at pull request comments with emojis noticeably longer than comments without them. Begel et al. [Begel and Vrzakova 2018; Vrzakova et al. 2020] conducted a study with 35 software engineers performing 40 code reviews while capturing their gaze with an eye tracker. The authors reported how long it took to confirm defect suspicions for each type of defect and the fraction of time spent skimming the code vs. carefully reading it. Hijazi et al. [2022] explored the use of eye-tracking and other biometric sensors with AI to determine the quality of code reviews. They focused on how effective participants were at finding bugs by analyzing their cognitive states during code reviews. Eye-tracking software was used to determine which part of the code the reviewer was looking at so that they could match the biometric data to a code region. Overall, this approach could predict code review quality with an accuracy of 87.77%. Bertram et al. [2020] investigates how developers' trust perceptions are affected by whether code patches are labeled as human or machine-generated. They found that participants spent more time looking at class-level methods when reviewing human-generated patches, but focused more on unit tests when looking at machine-generated patches.

The above studies are all similar in that they use eye tracking to uncover behaviors about the code review process. Our study differs in the following ways. First, we use a realistic setting within the browser to collect eye-tracking data. This means that the developer was not restricted to viewing a static screenshot. Instead, they had the ability to scroll through and interact with the elements on the page. We did this by extending iTrace [Guarnera et al. 2018] for Google Chrome. The plugin iTrace-Chrome let us seamlessly track where the developer was looking at while they were on task. Second, the tasks we investigated were different. We were interested in understanding how issues and pull requests are ranked in addition to looking at profiles/social signals related to accepting pull requests (as done by [Ford et al. 2019]).

3 EXPERIMENTAL STUDY DESIGN

3.1 Privacy and Ethics

The data being used was collected and processed in line with relevant institutional review board policy are Institution A. We strongly believe eye tracking is a useful during the design stage, by recording opt-in participants who are fully aware that they are being eye-tracked to use their data solely for research purposes to improve GitHub issues and code reviewing. All data provided is de-identified for preserve confidentiality of participants.

3.2 Participants

We recruited a total of 13 participants. The recruited participants are a mix of undergraduate and graduate students from the University of Nebraska at Lincoln, averaging 25 years old. Two of the 13 participants identified as women and 11 as men. On average, the participants stated that they perform around four code reviews each month. All but one participant was unfamiliar with the programming language used in the project for the code review tasks (Java).

3.3 Tasks and Subject System

The participants were given four tasks to complete, which are presented below:

Task 1. The first task involved ranking a list of open issues based on the order in which the developer would choose to work on them. The developers were allowed to click through the issues in Chrome as they pleased and were encouraged to think aloud during the process. See Figure 1 for how the list of issues was shown to participants.

Task 2. Similar to the first task, we asked participants to analyze a list of 10 open pull requests and prioritize them in the order in which they would be more comfortable reviewing them.

Task 3. In the third task, we asked the participants to evaluate the likelihood of 10 pull requests being accepted or rejected in their current state.

Task 4. The last task asked participants to view 23 different developer profiles with varying genders, countries of origin, companies, and time contributing. Each profile was shown to the participant, and they were asked to evaluate whether a pull request authored by that developer would likely be accepted or rejected. No specific pull requests were shown to the participant.

A similar screen to Figure 1 but for pull requests was shown for Tasks 2 and 3. Task 4 was the user profile page on GitHub. The participants did not apply a think aloud method in any of the tasks. In this study, the participants performed these tasks using the JabRef project[2]. JabRef is a Java application that helps users manage large BibTex files. All the issues and pull requests used in the study are listed in the replication package, along with the participants' answers.

[2]https://github.com/JabRef/jabref/

3.4 iTrace-Chrome: Tracking your Eyes Implicitly Within the Chrome Browser

The iTrace [Guarnera et al. 2018] community eye tracking infrastructure connects to an eye tracker to capture the *(x, y)* coordinates of where a person is looking at the screen. The raw gaze is recorded in an XML file. The iTrace-Chrome plugin connects to the iTrace-Core application and receives the `x,y` coordinates as well as a unique timestamp. When using Chrome, our plugin can then be configured to track different HTML DOM elements that contain that `x,y` input and records these results into its own XML file. Our first task to implement the iTrace-Chrome plugin was to identify a unique DOM pattern for each AOI in our study. Based on this DOM pattern, we could identify and distinguish each element uniquely. Then, the core application could check for these patterns and collect an *(x, y)* coordinate. If the *(x, y)* coordinate were on a DOM element that matched one of the AOI, it was logged in the XML file. The iTrace-Chrome plugin appears right next to the search bar in the top right of the page. After the data was collected, we run fixation event detection on the data and see different areas of interest that participants were focusing on the page.

3.5 Area of Interest (AOI) elements on GitHub Pages

In this study, we used five GitHub pages: User Profile, Issue list, Issue details, Pull Request list, and Pull request details. We mapped each page defining the regions that the participants use to complete each of the tasks described in the previous section. We classified AOI elements into two groups: information and meta-information. Information is the content of something in a profile, issue or pull request. On the other hand, meta-information is processed data that describes the information or the object of interest. For instance, the "issue title", "pull request title", "lines of code added or removed", and "comments" were classified as information. The "issue status", "issue label", and "comment date" are examples of meta-information.

3.6 Study Variables

We use two dependent variables in our analysis:

(1) **Fixation Count**, which measures the number of times a participant fixated upon an AOI
(2) **Fixation Duration**, which measures the duration of all fixations of a participant on an AOI.

To compare the groups of AOI, we computed the fixation count and fixation duration and applied the Mann-Whitney-Wilcoxon Test [Wilks 2011] to check if information and meta-information AOIs come from distinct populations; thus, we can decide whether the population distributions are identical without assuming that they follow a normal distribution. We also used Cliff's Delta [Romano et al. 2006] to quantify the difference between these groups of observations beyond p-value interpretation. The effect size magnitude is assessed using the thresholds provided in Romano et al. [2006], i.e. $|d|<0.147$ "negligible", $|d|<0.33$ "small" $|d|<0.474$ "medium", otherwise "large".

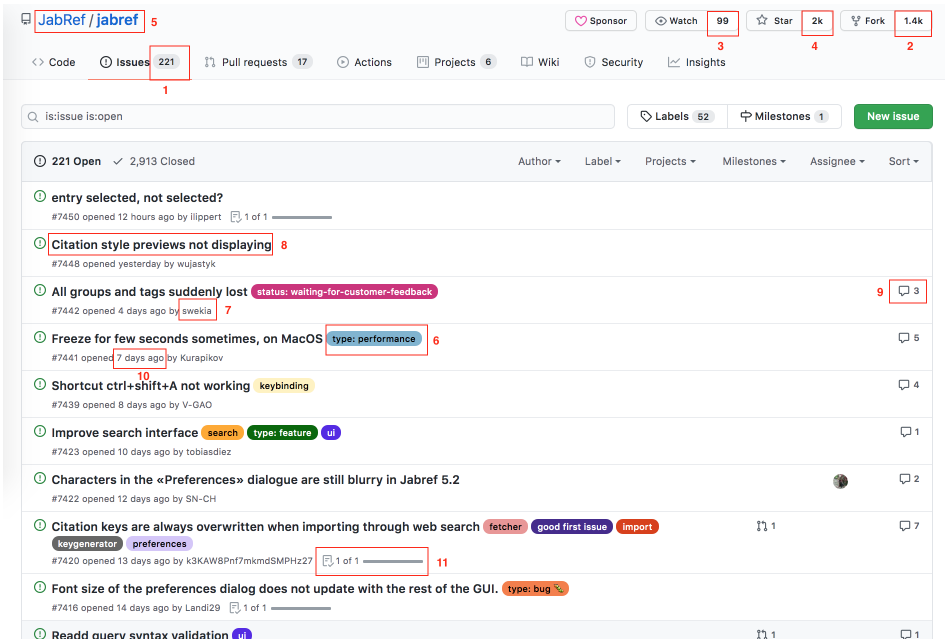

Figure 1: List of issues showing elements mapped. 1-Number of Open issues, 2-Number of Forks in repository, 3-Number of users watching repository, 4-Number of users who starred the repository, 5-Project Name, 6-Issue Label, 7-Username, 8-Issue Name, 9-Number of comments on the issue, 10-Time issue was opened, 11-Task completion bar.

3.7 Eye Tracking Apparatus

The Tobii TX 300 eye tracker to collect eye-tracking data. The eye tracker was set to run at 60Hz. i.e., producing 60 samples per second. We ran the Olsson fixation filter [Olsson 2007] on raw gazes to generate fixations of gazes that were 60 milliseconds or longer and were dispersed by less than 35px.

3.8 Replication Package

A comprehensive replication package including our anonymized dataset, instruments, and scripts is available in the Zenodo repository[3].

4 EXPERIMENTAL RESULTS

The results for each of the RQs is described next. Even though we report inferential statistics with a low sample, we also, more importantly, provide a qualitative analysis of describing the patterns and trends we see in the data.

4.1 RQ1 Results: What pieces of information do developers use to rank issues and pull requests?

To answer this question we analyzed data from Task 1 (rank issues to work) and Task 2 (prioritize the pull requests to review). On average, participants looked into 1,144 and 1,050 AOI elements to rank the issues and pull requests. On average, participants spent ~3 minutes to complete each task. Participants behave and focus differently. To effectively answer our RQ, we went in-depth analyzing how participants interacted with the AOIs in each task.

To rank the most visualized AOIs, we summed up the total time spent by all participants to complete Task 1. The five most visualized AOIs are "IssueTitle" (884,667 *ms*), "Comment" (630,084 *ms*), "CommentWord" (216,699 *ms*), "IssueLabel" (164,475 *ms*) and "CommentWordUnorderedList" (115,117 *ms*). Interestingly, we noticed that participants focused more on AOIs that provide information compared to meta-information (4 out of 5 AOIs in the Top-5 ranking). It is worth noticing that the total time spent in the Top-1 AOI (Issue Title) is 40% larger than the time spent in the TOP-2 (Comment). Figure 2 presents the AOI elements visualized by each participant. All participants presented a similar behavior regarding the Top-5 AOIs.

Finally, we manually inspected the most visualized words and labels, however, we found no pattern but do report the results to inform the reader. We found 1,446 words focused on by participants in Task 1. Inspecting the words, we found 412 interesting cases regarding the project domain (like JabRef, DOI, Bibtex, Latex), totaling 104 fixations.

During Task 2, participants focused on 846 words. Once again, no specific patterns were identified. We manually classified the words and grouped them into a few categories: "process discussion" (49 occurrences - e.g question, problem, TODO, etc), "domain" (25 - e.g bibtex, latex, JabRef), "UI" (16 - button, radio, table), "database" (10 - shared database), "network related" (8 - http, sync), "java related" (6, method, call, antlr3), 'system build"(5)", and "security" (2).

[3]https://doi.org/10.5281/zenodo.6402728

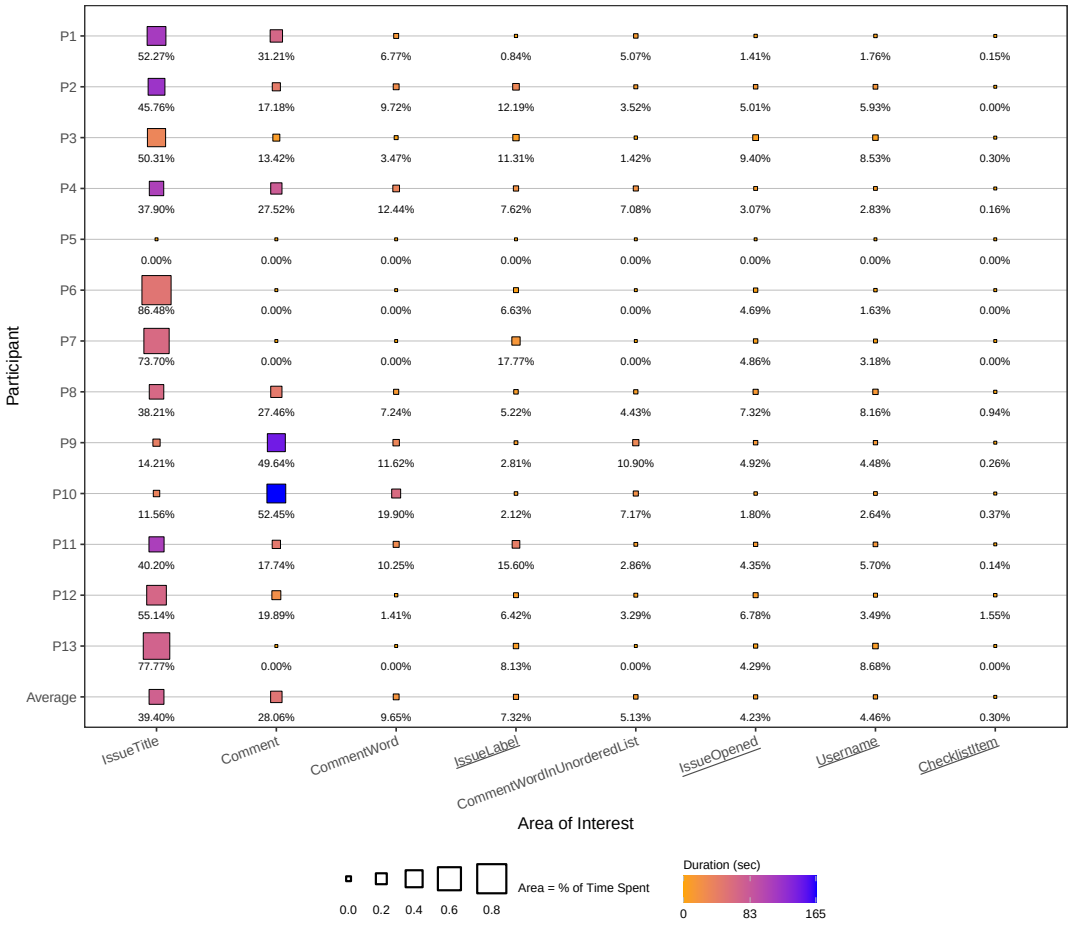

Figure 2: Task 1 AOI by participants. Meta-information items are underlined (x-axis). The square size represents the percentage of time each user fixates on each specific AOI; the color represents the absolute time spent on each AOI (orange represents the shortest time and blue represents the highest). We removed the AOI with less than 3% fixation for all participants to make the figure readable.

RQ1 Findings: Participants spend more time looking at information AOIs than meta-information ones to rank the issues and pull requests. Participants spent 40% more time looking at the title than at the comments to rank issues and pull requests.For the meta-information, labels played a central role in this process. For pull requests, people looked for cues provided by the usernames to understand who was involved in the tasks. Data showed that people use different approaches to rank the issues—some are more comprehensive and others are more selective.

4.2 RQ2 Results: What do developers look at when assessing a potential pull request?

To answer the second research question we analyzed how participants assessed the likelihood to accept 10 pull requests. To understand the outcome of their analysis, we compared the participants' outputs with what happened in the project.

For three pull-requests (#5789, #5614, and #5378 -available in replication package) we observed a tie, with 6 participants choosing to accept and 6 to reject the pull request. In two pull requests (#5702, #5749), most of the participants provided an opinion that disagree with what happened in JabRef. Task #5702 (the one with the highest discrepancy) was closed with no resolution and then re-opened. After 4 months after it was first opened, the task received the label "ready-for-review", after 14 new commits. There was a thorough discussion around some code snippets and some architectural decisions, before the merge. Another highlight is that the comment was the mostly visualized AOI. Still, the PR title does not seem as important as it was when ranking the pull requests. Once again the usernames of the stakeholders are among the Top-5.

When we asked participants how difficult it was to judge the likelihood of acceptance, 8 out of the 13 participants classified the task as hard/difficult, 3 of them as easy and 2 participants rated it as neutral. Given that the participants were not used to the project, we expected that the participants would perceive this as a difficult task.

P8 mentioned the importance of code and the number of comments involved in the code review process: *"I tend to look primarily for attention to detail, mostly in the code, but elsewhere as well. If the submission has lots of comments about, say, not following coding standards or PR guidelines, that's going to be a red flag. A lot of times, I wouldn't necessarily have the time/energy to really give a full, detailed review of the PR, so if I get the impression that the contributor really put a lot of detailed effort into it, that's going to be a mark in favor of accepting the PR."*. On the other hand, participants also reported the importance of checking the code style, linting tools, and the reviewer's opinion about the code submitted.

RQ2 Findings: Participants used more information than meta-information AOIs to assess the likelihood of pull request acceptance. Comments and code elements (e.g. lines of codes added, deleted, unchanged) attracted a fair amount of attention. Username appears as a top-5 AOI showing that the "social" side of the platform was considered in addition to explicit mention of reputation aspects by one participant.

4.3 RQ3 Results: What profile characteristics are more often assessed?

Building on the studies that show that social aspects may influence the evaluation of contributions [Ford et al. 2019; Marlow et al. 2013; Terrell et al. 2016; Tsay et al. 2014], in this research question we investigate which characteristics of GitHub profile may influence code reviewers. To answer this RQ, we asked the participants to access 25 GitHub profiles. We selected the GitHub profiles focusing on having a diverse set of characteristics, as follows: gender (man, woman, non-binary, non-identifiable from profile), nationality/first language, presence/absence of picture/avatar, use of alias or name, age of the profile after the first contribution (newcomers, experienced members), company workers vs. people who do not disclose their affiliation, and number of contributions (based on the contributions heat map and activity overview). In addition to these characteristics, we added seven actual JabRef contributors. After this process, we created a sandbox environment linking pull requests to the profiles we selected. In general, the agreement level among the participants was quite high. For 12 profiles, 10 or more participants voted that they are likely to have a contribution accepted on JabRef. Eight profiles also received 10 or more votes pointing that a contribution from that person would not be accepted. Only two profiles had a "closer call" with 7 vs. 6 votes (one was prone to have a contribution accepted, and the other was most likely to have a contribution rejected).

An interesting finding regards the seven profiles that made contributions to JabRef. The participants rated 4 of them as not likely to have their contribution accepted. Among the three profiles likely to have a contribution accepted, one was unanimous—13 of 13 votes to be accepted. In this case, the developer has 4 well-known organizations on their profile, more than 100 followers, and personal information available (email, company, country), however, no profile picture identifying them visually was present. The second highest-ranked profile (11 out of 13 positive votes) had a picture, personal information available, and more than 100 followers. Interestingly, the third-highest-ranked profile (10 positive votes) did not provide any personal information.

More specifically, the AOIs we found statistically significant differences were: Contribution Heat Map with large effect size ($p<0.05$, $|d| = 0.65$), Year with large effect size ($p<0.05$, $|d| = 0.64$), repository language with medium effect size ($p<0.05$, $|d| = 0.33$), user information with small effect size ($p<0.05$, $|d| = 0.25$), and Profile Label with small effect size ($p<0.05$, $|d| = 0.18$). We did not find any statistical difference in Avatar Image and the Repositories listed in the profile webpage.

RQ3 Findings: The contribution heat map and repository language were relevant AOIs that attracted more attention from participants when they assessed GitHub User Profiles.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we investigate how developers make decisions about issues and pull requests on GitHub, based on their eye movements. We found that artifact information is used more —in terms of fixation time—than meta-information available when developers are choosing what they should do first and when reviewing the pull requests. During the choice process, the title is the information that was the most used/read, while labels are the top-used/read on the meta-information side. For the pull-request review, comments and the code itself are the ones that attracted most of the attention (as expected). Our results shed light on the importance of having informative titles supported by labels to facilitate the decision. When looking at the social signals, the username was among the top-5 AOIs used during the review. Going in-depth on the profile side, we found that technical aspects and reputation (contributor heat map, repositories they contributed to, and languages used on these repositories) play a role in the decision to accept a pull-request. These insights could inform GitHub's UI such as making the username and label more prominent or refining the pull request ranking algorithms. Some interactive tools could also guide reviewers to any overlooked elements.

As part of future work, it would be interesting to conduct longer-term studies with OSS developers during the code review activity to understand how they behave in a real-world setting. Analyzing contributors performing other activities like bug triaging and labeling would also be great steps to inform automated approaches to aid software maintainers.

ACKNOWLEDGMENTS

We thank all the participants in this study. This work was funded in part by the US National Science Foundation under grant numbers CCF 18-55756, CNS 18-55753, IIS-1815503, and IIS-1900903, CNPq/MCTI/FNDCT grant 408812/2021-4; MCTIC/CGI/FAPESP grant 2021/06662-1; and Fundação Araucária grants PRD2023361000043.

REFERENCES

Deepika Badampudi, Ricardo Britto, and Michael Unterkalmsteiner. 2019. Modern code reviews-Preliminary results of a systematic mapping study. *Proceedings of the Evaluation and Assessment on Software Engineering* (2019), 340–345.

Andrew Begel and Hana Vrzakova. 2018. Eye Movements in Code Review. In *Proceedings of the Workshop on Eye Movements in Programming (EMIP '18)*. Association for Computing Machinery, New York, NY, USA, Article Article 5, 5 pages. https://doi.org/10.1145/3216723.3216727

Marcus Vinicius Bertoncello, Gustavo Pinto, Igor Scaliante Wiese, and Igor Steinmacher. 2020. Pull Requests or Commits? Which Method Should We Use to Study Contributors' Behavior?. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 592–601.

Ian Bertram, Jack Hong, Yu Huang, Westley Weimer, and Zohreh Sharafi. 2020. Trustworthiness Perceptions in Code Review: An Eye-Tracking Study. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20)*. Association for Computing Machinery, New York, NY, USA, Article 31, 6 pages. https://doi.org/10.1145/3382494.3422164

D. Ford, M. Behroozi, A. Serebrenik, and C. Parnin. 2019. Beyond the Code Itself: How Programmers Really Look at Pull Requests. (2019). https://doi.org/10.1109/ICSE-SEIS.2019.17

Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th international conference on software engineering*. 345–355.

Drew T. Guarnera, Corey A. Bryant, Ashwin Mishra, Jonathan I. Maletic, and Bonita Sharif. 2018. ITrace: Eye Tracking Infrastructure for Development Environments. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. Association for Computing Machinery, New York, NY, USA, Article 105, 3 pages. https://doi.org/10.1145/3204493.3208343

Haytham Hijazi, Joao Duraes, Ricardo Couceiro, Joao Castelhano, Raul Barbosa, Julio Medeiros, Miguel Castelo-Branco, Paulo De Carvalho, and Henrique Madeira. 2022. Quality Evaluation of Modern Code Reviews Through Intelligent Biometric Program Comprehension. *IEEE Transactions on Software Engineering* (2022), 1–1. https://doi.org/10.1109/TSE.2022.3158543

Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. 2020. Biases and Differences in Code Review Using Medical Imaging and Eye-Tracking: Genders, Humans, and Machines. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 456–468. https://doi.org/10.1145/3368089.3409681

Julia Koehler Leman, Brian D Weitzner, P Douglas Renfrew, Steven M Lewis, Rocco Moretti, Andrew M Watkins, Vikram Khipple Mulligan, Sergey Lyskov, Jared Adolf-Bryfogle, Jason W Labonte, et al. 2020. Better together: Elements of successful scientific software development in a distributed collaborative community. *PLoS computational biology* 16, 5 (2020), e1007507.

Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*. 117–128.

Pontus Olsson. 2007. Real-time and Offline Filters for Eye Tracking. , 42 pages.

Kang-il Park and Bonita Sharif. 2021. Assessing Perceived Sentiment in Pull Requests with Emoji: Evidence from Tools and Developer Eye Movements. In *2021 IEEE/ACM Sixth International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. 1–6. https://doi.org/10.1109/SEmotion52567.2021.00009

Mohammad Masudur Rahman and Chanchal K Roy. 2014. An insight into the pull requests of github. In *Proceedings of the 11th working conference on mining software repositories*. 364–367.

Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, and Jeff Skowronek. 2006. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys. In *annual meeting of the Florida Association of Institutional Research*. 1–33.

Bonita Sharif, Michael Falcone, and Jonathan I. Maletic. 2012. An Eye-Tracking Study on the Role of Scan Time in Finding Source Code Defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. Association for Computing Machinery, New York, NY, USA, 381–384. https://doi.org/10.1145/2168556.2168642

Jairo Souza, Rodrigo Lima, Baldoino Fonseca, Bruno Cartaxo, Márcio Ribeiro, Gustavo Pinto, Rohit Gheyi, and Alessandro Garcia. 2022. Developers' viewpoints to avoid bug-introducing changes. *Information and Software Technology* 143 (2022), 106766.

Igor Steinmacher, Tayana Uchôa Conte, and Marco Aurélio Gerosa. 2015. Understanding and supporting the choice of an appropriate task to start with in open source software communities. In *2015 48th Hawaii International Conference on System Sciences*. IEEE, 5299–5308.

Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson R Murphy-Hill, and Chris Parnin. 2016. Gender bias in open source: Pull request acceptance of women versus men. *PeerJ PrePrints* 4 (2016), e1733.

Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proceedings of the 36th international conference on Software engineering*. 356–366.

Mariot Tsitoara. 2020. *Better Project Management: Pull Requests*. Apress, Berkeley, CA, 159–182. https://doi.org/10.1007/978-1-4842-5313-7_12

Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. 2006. Analyzing Individual Performance of Source Code Review Using Reviewers' Eye Movement. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications (ETRA '06)*. Association for Computing Machinery, New York, NY, USA, 133–140. https://doi.org/10.1145/1117309.1117357

Hana Vrzakova, Andrew Begel, Lauri Mehtätalo, and Roman Bednarik. 2020. Affect Recognition in Code Review: An In-situ Biometric Study of Reviewer's Affect. *J. Syst. Softw.* 159 (2020). https://doi.org/10.1016/j.jss.2019.110434

Daniel S Wilks. 2011. *Statistical methods in the atmospheric sciences*. Vol. 100. Academic press.