Qualitatively Analyzing PR Rejection Reasons from Conversations in Open-Source Projects

Tanay Gottigundala, Siriwan Sereesathien, Bruno da Silva
Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
{pgottigu, ssereesa, bcdasilv}@calpoly.edu

***Abstract*—Software developers have largely relied on pull requests as a mechanism of collaboration in their projects. Researchers have collected and analyzed pull request data in different ways for different reasons. In particular, we have qualitatively analyzed pull request conversation data to understand the main reasons for pull request rejection from a developer's perspective. In this paper, we report results from ongoing research on identifying and categorizing pull request rejection factors. Two software developers, co-authors of this paper, manually analyzed 605 rejected PRs from Hexo and ESLint. We found that the most frequent reasons for PR rejection may vary depending on the project size and popularity. Still, some common rejection factors include implementing unnecessary functionality, conflicting PRs, agreement to make PR reattempts, and inactivity. Code quality issues are not among the most frequent reasons.**

***Index Terms*—pull request, code review, open-source software, developer communication, social software engineering**

I. Introduction

The pull-based development model has changed the way developers make contributions to software projects worldwide. Platforms like GitHub and GitLab typically offer additional support for collaboration around pull requests (PRs) on top of Git. Besides being popular in the software industry, researchers have also found scientific evidence of benefits that the pull request model offers. These benefits include a more efficient code review process, decreased time to incorporate changes, and increased opportunity for community engagement [1].

Recently, researchers have analyzed pull request data, especially from open-source projects, in several different ways and for different purposes [1] [2] [3] [4] [5] [6] [7]. However, only a few of them have analyzed a limited number of pull request conversations to fully understand PR rejection reasons in a qualitative fashion from a developer's perspective. For instance, in [1], the authors manually analyzed 350 PRs to identify and classify rejection reasons, whereas in [8] Steinmacher et al., with similar purposes, manually analyzed 263 PRs among other collected data. In [9], the authors also qualitatively analyzed 231 rejected PRs from one open-source project and proposed guidelines for open-source contributors.

In this work, we have qualitatively analyzed pull request conversation data to understand the main reasons for pull request rejection from a developer's perspective. We intend to complement existing research by iterating over a systematic process of manually analyzing and classifying rejected PRs by rejection reasons. We also planned to cover two different projects of diverse sizes and popularity, involving a number of PRs that have never been investigated by following such a qualitative approach. For this work, we analyzed two open-source projects: Hexo and ESLint. Our investigation involved 605 rejected PRs manually analyzed and classified by two separate developers and co-authors of this paper. The research questions we set for this work are:

RQ1: What are the most frequent reasons why pull requests get rejected in open source projects?

RQ2: Are there significant differences in pull request rejection reasons in projects of different sizes and popularity?

RQ3: Are code quality issues the most driving factor for rejecting pull requests in open source projects?

II. Study Settings

The two open-source projects we analyzed are ESLint and Hexo. ESLint is a static code analysis tool that helps identify problems in JavaScript code, whereas Hexo is a blog writing framework. We selected these two projects because they target different users in separate domains, have different sizes and popularity, and have a significant PR activity to be analyzed.

With over 8k issues and nearly 6k PRs in total, ESLint is a very popular plugin that attracts many developers willing to contribute. It has 17.8k stars and 3.2k forks on GitHub. ESLint also has an extremely high download rate, with its peak average weekly downloads setting at over 15 million in 2020 and totaling over 1.3 billion download counts overall. Hexo is less popular than ESLint but still highly used and active in terms of contributions, with about 3.5k issues and almost 1k pull requests in total. In terms of usage, it has 32k stars and 4k forks on GitHub. Hexo also set its peak weekly download at nearly 20k in 2018 with a total download count of over 2 million. All these numbers for both projects were checked on Jan 2021.

We manually analyzed a total of 605 rejected pull requests[1] (155 from Hexo and 450 from ESLint). The 155 PRs from Hexo represent all the rejected PRs in that project at the time we set to perform the study. The 450 PRs from ESLint

[1]Our dataset is publicly available at https://doi.org/10.5281/zenodo.4499541

represent the most recent 450 rejected PRs in that project at the time we gathered data for this work. For analyzing and classifying the rejection reasons, we followed a systematic coding process similar to [9] and [1]. We started with the rejection categories provided in [9]. Divided into multiple analysis sessions, two co-authors of this paper read through all the 605 PRs separately. During each session, they individually classified each PR based on the categories from [9]. Besides, they created their own labels and categories as they saw fit. In between individual sessions, they scheduled synchronization meetings to compare their analysis results, debating over different classifications they had, and coming to a consensus. Also, they discussed potential changes on the set of rejection categories and specific reasons in each category, thus iteratively updating the model provided by [9].

Regarding the possibility of having multiple reasons for rejecting a PR, the analysts always tried to identify the most relevant cause of the PR rejection (even if it potentially had other reasons). Generally, the rejection reason is usually aligned with the most recent messages in the PR conversation. Only a minimal number of PRs had more than one reason for rejection in the classification. For the final set of rejection reasons organized by category, see the following list.

- Author Issues [A]
  - **Author unable to fix:** Author unable to fix the issue that was found or gives up
  - **Closed accidentally:** Author unfamiliar with PRs and code review process, closes the PR accidentally and creates a new one
  - **Closed by author:** Author closed the PR without a discernible reason
  - **Closed due to inactivity**: Author abandoned the PR for a long period
  - **Ego issues:** Author or reviewer was generally hostile or hard to work with
  - **No description:** Reviewer closed without giving a reason
  - **Not in English:** The PR discussion or description was not in English
- Code Issue [CODE]
  - **Bad code formatting:** The code formatting was not up to industry or company standards
  - **Bad coding practice:** The PR did not follow proper coding conventions
  - **Hard to read:** The pull request description or code was difficult to understand
  - **Insufficient testing**: Feature or changes added did not include sufficient tests, which are required for the PR to be accepted according to the open-source project
  - **Ego issues:** Author or reviewer was generally hostile or hard to work with
  - **Wrong logic:** The author did not understand the existing code correctly, thus proposing wrong changes
- Unnecessary Issue [UNEC]
  - **Functionality already exists:** Functionality already exists as author tried to add similar functionality unknowingly
  - **Issue not reproducible:** The issues is not a real issue, author misunderstood the code base as wrong while it is working as intended
  - **Lack of team consensus:** Team maintaining project could not come to an agreement of accepting the change or is against the change
  - **Minor change**: The change was too insignificant to be accepted
  - **Unnecessary functionality:** Additional functionality that isn't something the maintaining team feel would fit with the project
  - **Unnecessary refactoring:** Changes in the code base that isn't significant enough for the team maintaining the project to think its essential
  - **Wrong code location:** PR changes are put in the wrong branch or repository of the project
- Side Effect Issue [SE]
  - **Breaks compatibility:** The PR is a breaking change for some/all use cases
  - **Fails tests:** Build failed due to failing tests
  - **Performance issues:** The changes caused a notable decrease in performance
  - **Unintended side effects**: The changes caused undesirable functionality elsewhere
- Version Control Issue [VC]
  - **PR reattempt:** Author has readdressed (or will readdress) changes in a new PR.
  - **PR conflict:** PR is a duplicate of another change by a different author
  - **PR outdated:** Issue regarding the PR changes were already closed/fixed, new changes to the code base makes the PR change unnecessary
  - **PR too large**: Changes for the PR are too large, so reviewers suggest to break it up into multiple PRs

III. Results

A. *RQ1: What are the most frequent reasons why PRs get rejected?*

For answering this research question, we decided to disregard the reason "closed by author" because it does not represent a PR rejected by reviewers. Several reasons may lead an author to close a PR, including the author's impetus for not working on the PR anymore. Also, in most of them, the authors do not even post a justification for closing.

Figure 1 summarizes the distribution of PR rejection reasons from Hexo. The brackets represent the rejection categories listed in the previous section. The green bars represent each of the five rejection categories' tallies, and the blue bars represent rejection reasons marked with tags representing the categories to which they belong.

The top five reasons include "Reattempt", "Closed due to inactivity", "Unnecessary functionality", "Wrong code location", and "No description". Since Hexo has several first-time contributors, the top reasons for rejection are issues that newer contributors would likely encounter.

The "Reattempt" reason was classified when the PR author created (or agreed to create) another PR to tackle the same issues in a way that improves the original PR contents. It turns out that an overwhelming amount of PRs is reattempted in Hexo. Also, PRs in this project were often "Closed due to inactivity", perhaps because new contributors lose interest quickly or realize the complexity of the changes, even when they get feedback and support from reviewers. Reviewers also considered several PRs with "Unnecessary functionality". The high frequency of "Wrong code location" in this repository was because Hexo has many plugins available that extend its functionality. The core maintainers often felt that certain features belonged in one of the plugin repositories rather than the main Hexo repository. Reviewers closed a surprisingly high number of PRs in this project since they had “No description”. From our observations, this was usually a result of inexperienced GitHub users creating PRs that did not follow basic contribution guidelines, such as including a description. Finally, looking at the rejection categories, apart from PR Reattempts, two out of the top five reasons were under the “Unnecessary Issue” category and another two under the “Author Issue” category. Side Effect and Code Issues did not bubble to the top as frequent reasons compared to others.

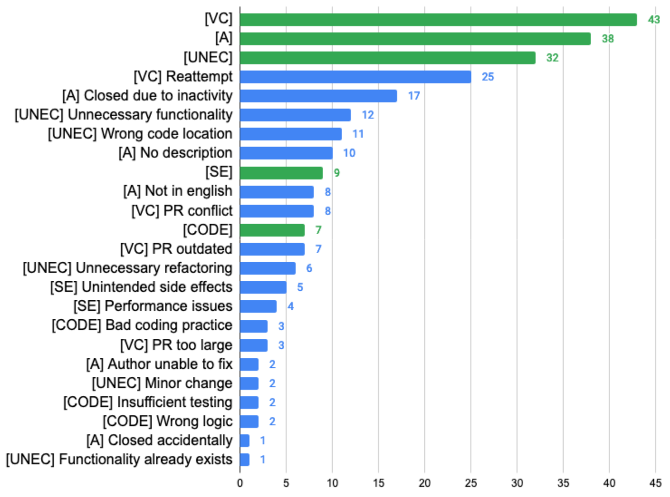

Fig. 1. Count of PR Rejection Reasons in Hexo (Closed by Author excluded)

Figure 2 summarizes the distribution of PR rejection reasons from ESLint. The top five reasons include “Lack of team consensus”, “Unnecessary functionality”, “PR conflict”, “Unnecessary refactoring”, and “PR outdated”.

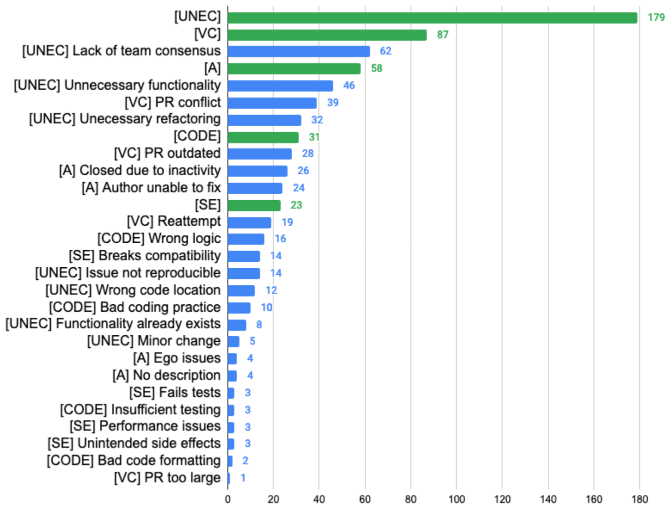

Fig. 2. Count of PR Rejection Reasons in ESLint (Closed by Author excluded)

One unique aspect of ESLint is its large and active maintaining team. This is necessary for an open-source project of that size. Due to its scale and popularity, they receive a high volume of PRs. As a result, they only accept PRs that strictly adhere to their vision, shown by the number of PRs rejected due to a “Lack of team consensus” and “Unnecessary functionality”. Another result of the project’s size was keeping consistent formatting, structuring, and practices across the project. Due to these guidelines’ strictness, a significant number of PRs were rejected because of “Unnecessary refactoring”. Because of its popularity, ESLint often had conflicting PRs that dealt with the same issues, leading to many contributions being rejected due to “PR conflict”. The popularity also played a role in rejected PRs labeled “PR outdated” because the codebase is continually changing, and the authors have trouble keeping up with the current state of the code base in the context of their contributions. Finally, looking at the rejection categories, we observe a significant concentration on rejections under the “Unnecessary Issue” category.

B. RQ2: Are there significant differences in PR rejection reasons in projects with different sizes and popularity?

As noted, the size, scale, and popularity of these two projects change significantly. As a result, we observed that most of their differences in PR rejection reasons stem from their distinct project size, popularity, and consequent level of organization and strictness for managing PRs.

When comparing Hexo and ESLint, we noticed that the only common reason among their top five reasons was “Unnecessary functionality”. It is also consistent with other works in this area, such as [1] [9] that pointed this reason as a reasonably common one over the open-source projects they analyzed.

Because of the volume of PRs in ESLint, the core development team could not review and merge every PR, even when the PR could improve the project. Hexo team, on the other hand, were able to review more PRs that helped authors adapt their change-set to the project expectations. As a result, “Lack of team consensus” was a reason that only arose in ESLint. In general, perhaps due to a lower number of PRs pending reviews, Hexo was more likely to guide authors in improving the PRs, leading some PRs to be reattempted. That is probably why “PR Reattempt” rose to the top in Hexo, whereas it stayed middle-ranked in ESLint.

Comparatively, ESLint had more “PR Conflicts”, though. This might be due to a much higher contribution activity in ESLint in contrast with a limited number of review resources, which leads to a higher frequency of conflicting contributions being proposed relatively in parallel.

Some reasons we only identified in either one of the projects. For instance, only Hexo had PRs “Not in English”. While only in ESlint we found “Lack of team consensus”, “Breaks compatibility”, and “Issues not reproducible”. ESLint guidelines for contributions seem clearer, which is probably why we did not see contributions in a spoken language not supported by the project. Also, as we noted in section III-A, “Lack of team consensus” as the top reason in ESLint and not present in Hexo is probably because in ESLint, a much larger project with many more PRs, the core maintainers had to follow their internal review policies strictly, which led several PRs being rejected in that project.

Comparing the most frequent categories in both projects, "Unnecessary Issue" (UNEC), as a category of different rejection factors, is a common area, with "Unnecessary functionality" being the most common reason across the two projects, as already discussed in this section. In general, for both projects, "Version Control Issues" (VC) and "Author Issues" (A) tend to be more frequent as categories compared to "Side Effect Issues" (SE) and "Code Issues" (CODE).

C. RQ3: Are code quality issues the most driving factor for rejecting PRs?

This question has been also explored in other works [10] [7]. Many might think that it is reasonable to have a high number of pull requests being rejected due to code quality issues (see our CODE category on Section II). However, if we consider results from [10] and [7] this is not necessarily true. Some results in the literature are not consensual. In [7], they found that code quality flaws measured by the PMD tool turned out not to affect the acceptance of a PR. However, in [10], they found that code quality issues are vital for rejecting PRs. Therefore, we set out to analyze this question as well, but with qualitative data as we have described in this paper.

Looking at our rejection reasons distributions for both projects, we observe that none of the CODE reasons rose to the top. The highest-ranked CODE reason was "Wrong Logic" in ESLint, taking the ninth position in that ranking. In general, CODE-related reasons take low to medium spots in the distribution of rejection reasons, which is consistent with [9] and [7] results.

Note on threats to validity: This study is limited by the manual analysis of two co-authors of this work. We mitigated researchers' implicit biases by defining a systematic process described in Section II that includes a calibration step with a small number of PRs and discussion rounds for consensus-based conflict resolution. All PRs were analyzed twice by two different human classifiers.

IV. Conclusion

It is not new that software development is strongly reliant on human collaboration. The pull request mechanism has shaped how many software teams and projects around the world have evolved their codebase. It has been particularly relevant to how global communities have maintained open-source software. Researchers have leveraged the abundance of pull request data to learn how PRs are reviewed and then provide scientific evidence on current practices' benefits and discover new information from open data.

We set out to qualitatively analyze and classify rejected PRs in two open-source projects with different sizes and popularity. Out of the 605 manually analyzed PRs, we provided evidence that PR rejection reasons may differ significantly depending on the project size and popularity. These factors may impact how organized the contribution guidelines are and how strict the maintainers apply a rigorous reviewing approach. Across both projects, the most frequent reason why PRs got rejected was "Unnecessary functionality". Also, "PR Conflict", "Closed due to inactivity", and "PR Reattept" took medium to high positions in both rankings. Finally, we also found evidence that code quality issues are not among the main reasons for rejecting PRs in both projects. Although with different methodologies, datasets, and research questions, our work has been somewhat consistent with results reported in [1] [8] [9] [7]. With this work, we have complemented the literature by providing additional and complementary evidence using a qualitative method over an amount of PRs that have never been manually analyzed to date.

Still, this work is a continuing attempt toward building more knowledge on PR rejection reasons by applying qualitative research methods. Such an effort can ultimately lead to practical guidelines for globally distributed contributors in open-source projects. Also, tool builders may use this research to help them build tools to support a more efficient pull request process. Our future work involves expanding the analysis to more projects and looking at additional factors (e.g., reviewer history of acceptances and rejections, and social aspects).

References

[1] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th Intl Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 345–355.

[2] D. M. Soares, M. L. d. L. Júnior, L. Murta, and A. Plastino, "Rejection factors of pull requests filed by core team developers in software projects with high acceptance rates," in *2015 IEEE 14th Intl Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 960–965.

[3] J. Terrell, A. Kofink, J. D. Middleton, C. Rainear, E. R. Murphy-Hill, C. Parnin, and J. Stallings, "Gender differences and bias in open source: pull request acceptance of women versus men," *PeerJ Computer Science*, vol. 3, p. e111, 2017.

[4] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey, "Effects of personality traits on pull request acceptance," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[5] D. Ford, M. Behroozi, A. Serebrenik, and C. Parnin, "Beyond the code itself: How programmers really look at pull requests," in *2019 IEEE/ACM 41st Intl Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2019, pp. 51–60.

[6] T. Dey and A. Mockus, "Effect of technical and social factors on pull request quality for the npm ecosystem," in *Proceedings of the 14th ACM / IEEE Intl Symposium on Empirical Software Engineering and Measurement (ESEM)*, ser. ESEM '20. New York, NY, USA: Association for Computing Machinery, 2020.

[7] V. Lenarduzzi, V. Nikkola, N. Saarimäki, and D. Taibi, "Does code quality affect pull request acceptance? an empirical study," *Journal of Systems and Software*, vol. 171, p. 110806, 2021.

[8] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost there: A study on quasi-contributors in open-source software projects," in *2018 IEEE/ACM 40th Intl Conference on Software Engineering (ICSE)*, May 2018, pp. 256–266.

[9] N. Papadakis, A. Patel, T. Gottigundala, A. Garro, X. Graham, and B. da Silva, "Why did your pr get rejected? defining guidelines for avoiding pr rejection in open source projects," in *Proceedings of the IEEE/ACM 42nd Intl Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 165–168.

[10] M. C. Silva, M. T. Valente, and R. Terra, "Does technical debt lead to the rejection of pull requests?" in *Proceedings of the XII Brazilian Symposium on Information Systems*, ser. SBSI 2016. Porto Alegre, BRA: Brazilian Computer Society, 2016, p. 248–254.