Who Gets a Patch Accepted First? Comparing the Contributions of Employees and Volunteers

Gustavo Pinto
Federal University of Pará
Belém, PA, Brazil
gpinto@ufpa.br

Luiz Felipe Dias
University of São Paulo
São Paulo, SP, Brazil
fronchetti@usp.br

Igor Steinmacher
UTFPR, Campo Mourão, PR, Brazil
Northern Arizona University, USA
igorfs@utfpr.edu.br

ABSTRACT

Although many software companies have recently embraced Open Source Software (OSS) initiatives, volunteers (i.e., developers who contribute to OSS in their spare time) still represent a wealthy workforce that have the potential of driving many non-trivial open source projects. Such volunteers face well-known barriers when attempting to contribute to OSS projects. However, what is still unclear is how the problems that volunteers face transcend to the problems that employees (i.e., developers hired by a software company to work on OSS projects) face. In this paper we aim to investigate the differences on the acceptance of patches submitted by volunteers and employees to company-owned OSS projects. We explore different characteristics of the patches submitted to company-owned OSS project, including: the frequency of acceptance and rejection; the total time to review and process a patch, and; whether the changes proposed follow some contribution best practices. We found that volunteers face 26× more rejections than employees. Volunteers have to wait, on average, 11 days to have a patch processed (employees wait 2 days, on average). 92% of the dormant pull-requests (e.g., pull-requests that take too long to be processed) were submitted by employees. Finally, we observed that the best practices that had the patches are most adherent to is "commit messages need to be written in English."

KEYWORDS

Company-Owned OSS Projects; Employees; Volunteers

ACM Reference Format:
Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. 2018. Who Gets a Patch Accepted First? Comparing the Contributions of Employees and Volunteers. In *Proceedings of Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'18)*. ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

Along the last decades, Open Source Software (OSS) development was often regarded as a voluntary activity [9],in which developers spend their own free time to construct/design/test/refactor projects. However, the last few years introduced fundamental changes on how OSS is developed. These changes are mostly due to the wave of software companies that started not only to embrace OSS initiative (e.g., fostering hackathons or code jams), but also supporting and releasing OSS [10]. As a notable example, Google alone has contributed to more than 2,000 OSS projects[1]. As a consequence, OSS contributors are now a mix of both volunteers and employees (i.e., developers hired by a company specifically to contribute to OSS projects).

However, what is so far not clear is how these contributions, made by developers with conflicting interests, differ. This is particularly relevant in the pull-request[2] era, in which everyone interested in contributing to an OSS project might follow the same process (e.g., filling a pull-request with the proposed change) and are subject to the same scrutiny from their peers (e.g., during review cycle). In this scenario, one might believe that since employees have more access to integrators [6] chances are that they are more likely to have a contribution reviewed and processed faster than volunteers, who are in the other side of the wall and have less communication channels to bring integrators' attention to their patches. Unfortunately, delay to process a patch is only one possible side effect. In this paper we studied this and some other characteristics related to patches submitted by employees and volunteers.

We used a convenience sample to find five GitHub-owned projects: `atom`, `electron`, `hubot`, `git-lfs`, and `linguist`. We chose these projects because they were initially developed by (and are maintained at) GitHub, therefore we could take advantage of GitHub features to understand whether a contributor is a employee or a volunteer (more details at Section 2.2). Through an extensive analysis of ~12k patches submitted, we evidenced contribution behaviors that are intrinsic for each particular kind of contributor.

2 METHOD

In this early report, we studied three research questions:

RQ1: Do volunteers have to try more than employees to have a patch accepted? In this question we investigated the number of attempts that volunteers and employees have made to contribute to company-owned OSS. We hypothesize that volunteers have a higher rate of non-accepted contributions.

RQ2: Do volunteers have to wait much more than employees to have a patch processed? Here we inspected the number of days that a patch takes to receive a decision (accept or reject). We hypothesize that volunteers have a higher rate of dormant patches(i.e., patches that take too long to be processed).

RQ3: Do volunteers follow contributing best practices? Here we employed three well-discussed best practices [1, 5, 6, 13–15]: (1) The contribution should be small (**BP1**) [5, 15]; (2) The contribution

[1]https://opensource.google.com/
[2]Throughout this paper, we employ the terms "pull-request" and "patch" interchangeably.

should be accompanied with test cases (**BP2**) [6, 14]; and (3) the contribution should have a descriptive commit message (**BP3**) [1, 13] . We hypothesize that volunteers do not follow most of the best practices.

2.1 Mining Repositories Data

First, we had to differentiate employees and volunteers. To distinguish these developers, we relied on GitHub `site_admin` flag. This flag is set to GitHub users that are GitHub employees. Therefore, for any user that does not work for GitHub, this is set to false. Consequently, we used this flag to categorize employees and volunteers in the analyzed OSS projects, which are owned by GitHub (the company).

For each project, we collected data of the pull-requests using GitHub API. We ended up with a list of 11,895 pull-requests: 5,143 submitted by employees, and 6,742 by volunteers. For each pull-request, we considered the three possible statuses for our analysis:

- *open*: waiting for code reviews and/or a final decision;
- *closed*: the code reviews were done, but the pull-request was not accepted (the status in GitHub is closed/unmerged);
- *merged*: the code reviews were done, and the pull-request was accepted (the status in GitHub is closed/merged).

For each pull-request, we payed particular attention to:

- the number of commits per pull-request;
- the number of changes per pull-request;
- the commit message, and;
- the time taken to process the pull-request.

The data reported in this paper is based on pull-requests performed from the very beginning of the studied projects, up to January, 2018 — when we collected the data. All data used in this study is available online at the companion website[3].

2.2 Studied Projects

In this study, we characterize our sample as multiple cases of company-owned OSS. To define our sample, we searched for software companies that have made some of their software publicly available as OSS. However, as a constraint of our approach (as we discussed in Section 2.1), we have to focus our search on projects developed by GitHub (the company) at GitHub (the social coding website). We studied five projects in this regard:

- `atom`, a cross-platform text editor. It has ~34,300 commits, ~3,750 pull-requests, 400 contributors, ~43,000 stars, and ~8,400 forks. It is mostly written in JavaScript and CoffeeScript, and has ~6 years of historical records.
- `electron`, a tool to build cross platform desktop apps with JavaScript, HTML, and CSS. It has ~18,000 commits, ~3,800 pull-requests, 721 contributors, ~56,000 stars, and ~7,200 forks. It is mostly written in C++, and has ~4 years of historical records.
- `hubot`, a customizable life embetterment robot. It has ~2,000 commits, ~700 pull-requests, 253 contributors, ~13,700 stars, and ~3,200 forks. It is mostly written in JavaScript, and has ~6 years of historical records.
- `git-lfs`, a git extension for versioning large files. It has ~6,300 commits, ~1,300 pull-requests, 99 contributors, ~5,300 stars, and ~900 forks. It is mostly written in Go, and has ~4 years of historical records.
- `linguist`, a library to detect blob languages. It has 5.600 commits, ~2,400 pull-requests, 684 source code contributors, ~5400 stars, and ~2,000 forks. It is mostly written in Ruby, and has ~6 years of historical records.

We used the `cloc`[4] utility to calculate the Lines of Code (LoC). It includes code from all the languages in which a project was developed, as well as blank lines and commented lines. The largest project is linguist, with 203K lines of code. `atom` is the project with the greatest number of commits (343K), while `linguist` is the project with the greatest number of unique committers (688). With regard to their popularity, `electron` is the most stared (56K) and `atom` is the most forked one (13K).

3 RESULTS

This section presents reports the results obtained from both repository mining and manual patch analysis. Section 3.1 analyzes who gets a patch accepted first. The differences on the time taken to review a patch is investigated next, in Section 3.2. Finally, the results of the patch analysis can be found in Section 3.3.

3.1 RQ1: Do volunteers have to try more than employees to have a patch accepted?

We analyzed the number of rejected patches for both employees and volunteers. We found 1,453 unique volunteers and 56 employees that have patches rejected. We found that most of the developers have very few rejections. This is particularly true for volunteers, which usually do not provide many patches if they have already a patch rejected [14]. The average number of patches rejected per volunteer is 1.42 (median=1, q3=1). At an extreme case, in project `electron`, we found one volunteer that had 29 rejected patches. We hypothesize that this high number of rejections from volunteers are because volunteers might be motivated by particular needs, which might not be necessarily aligned with the project's roadmap. This can also be related to the way the project integrators merge pull-requests, like cherry-picking commits or using Command Line Interface to merge particular commits [14]. In contrast, employees are more likely to have more rejected patches; the average number of patch rejected per employee member is 5.78 (median=2, q3=7). Similarly, one employee had 54 patches rejected. We believe that this happens due to the role that each type of contributor have in the project: volunteers might not have strong ties with the project that they are contributing to, and in the face of rejections, they may become demotivated [14], and may not contribute anymore. In fact, 1,200 (82%) volunteers have tried only once.

The project `hubot` is the one with the highest number of volunteers' patches rejected (64% of the patches were rejected). In the same project, we found that only 2% of patches from employees were rejected. On the other hand, project `atom` has the highest number of patches rejected for employees (52% of the patches were rejected). In the same project, 28% of volunteers' did not make it.

[3]To be published upon acceptance.

[4]https://github.com/AlDanial/cloc

Among the 1,453 volunteers with patches rejected, only 414 (28%) also have at least one patch accepted. For the employees, the percentage is even smaller: 9 employees (15%) who had a patch rejected also had (at least) one patch accepted. Furthermore, when analyzing the patches accepted, we observed that few employees concentrate much of the development process: 7 employees (5%) are responsible for 58% of the patches accepted; employees have, on average, 39 patches accepted (median=3, q3=2). On the other hand, contributions from volunteers tend to be much more sporadic: 76% of the volunteers are casual contributors (i.e., contributors that have only a single contribution accepted [8, 11]). On average, volunteers have 2.3 patches accepted (median=1, q3=2).

3.2 RQ2: Do volunteers have to wait much more than employees to have a patch processed?

In this research question, we studied the difference, in terms of days, between the date that the pull-request was opened and the date when it was merged. In this case, we are only considering accepted pull-requests since they take longer to be processed than non-accepted ones (e.g., developers that fill the pull-request often have to complement the pull-request with additional changes [5]).

On average, patches proposed by volunteers take 11.37 days to be processed (min=0, max=1,144, q3=5, stdev=55). In comparison, pull-requests from employees take 2.61 days (min=0, max=558, q3=1, stdev=18). For all studied projects, on average, pull-requests submitted by employees are processed faster than the ones submitted by volunteers. In particular, projects `hubot` and `linguist` are the ones that take more time to process pull-requests, either those submitted by employees (333 and 426 days for `hubot` and `linguist`, respectively) or by volunteers (1,144 and 832 days for `hubot` and `linguist`, respectively).

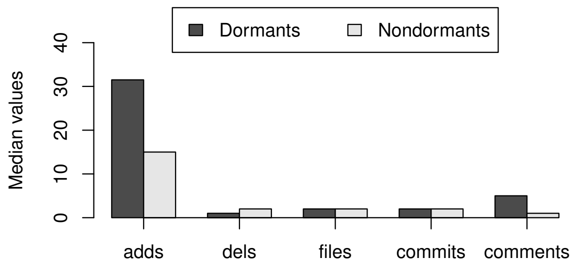

Figure 1: Median values for dormant and non-dormant pull-requests

We then analyzed such pull-requests that take too long to be processed (i.e., the ones that take more than 100 days to be processed, the so called "dormant pull-requests"). We found 144 dormant pull-requests (`atom`: 29, `electron`: 6, `hubot`: 22, `git-lfs`: 5, `linguist`: 82). Only 12 out of the 144 dormant pull-requests were submitted by employees. All of these dormant pull-requests were accepted; no pull-request rejected took more than 100 days to be processed. Notwithstanding, the number of additions, deletions, files changes, commits, and comments per dormant pull-request are greatly higher than non-dormant pull-requests. Figure 1 compares the median values of these metric in the two groups. The averages were skewed by the outliers present in the dormant group.

3.3 RQ3: Do volunteers follow contributing best practices?

As discussed in Section 2, the listed best practices indicate that a patch should be small (BP1), the contribution should be accompanied with tests (BP2), and the commit message should be descriptive (BP3). For BP1, we understand that a small patch should change at most 2 files and add at most 20 lines. These values were chosen because they represent the median values of these metrics [5]. Figure 3 shows the results for BP1. Considering both closed and merged pull requests, most of the volunteers submitted patches that conform to BP1. Interestingly, employees present a lower adherence to BP1. One explanation to this fact is because employees might work on critical issues, while volunteers might dedicate their efforts on simpler changes.

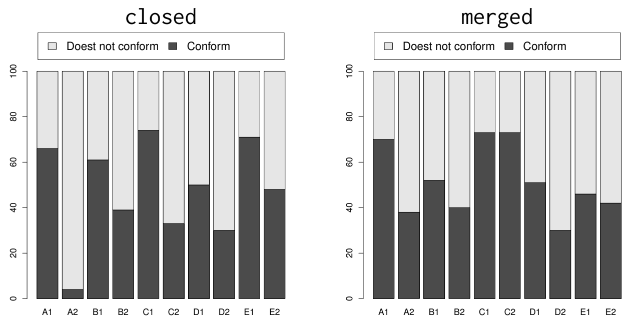

Figure 2: Percentage of pull-requests submitted by volunteers and employees that conform BP1. Each studied project is represented by a sequence of two bars. The first bar (A1, B1, etc) represents volunteers data, while the second (A2, B2, etc) represents employees. A means `atom`, B means `electron`, C means `hubot`, D means `git-lfs`, and E means `linguist`.

To find test files (BP2), we made use of a regular expression evaluated elsewhere [3], slightly changing it to cover different file extensions (the original was used only to find Java files). Figure 3(a) shows that the majority of the accepted pull-requests does not include test files. This finding goes against recent studies that suggest that project integrators are more likely to reject pull-requests that do not come with test cases [6, 14]. The main exception is project `git-lfs`, in which 40% of the volunteers and 59% of the employees provide test cases.

Finally, for this study, a descriptive commit message (BP3) is the accumulation of three sub-best practices: it should be short (i.e., at most 50 characters long) [1], the second line should always be blank, and the message should be written in English [13]. We removed merge commits due to their automatically generated commit messages. To identify English-language commits, we used `langid.py`[5] to estimate the probability that all project's commits are in English. As Figure 3(b) shows, BP3 has low adherence for both groups. However, the majority of the pull-requests are written in English

[5]https://github.com/saffsd/langid.py

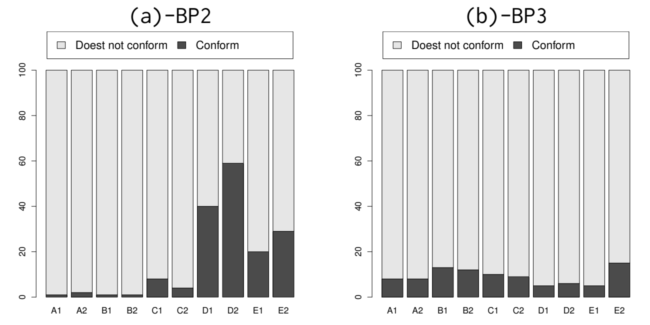

Figure 3: Percentage of accepted pull-requests submitted by volunteers and employees that conform BP2 (a) and BP3 (b). Each studied project is represented by a sequence of two bars. The first bar (A1, B1, etc) represents volunteers data, while the second (A2, B2, etc) represents employees. A means `atom`, B means `electron`, C means `hubot`, D means `git-lfs`, and E means `linguist`.

(97% submitted by volunteers and 98% submitted by employees). The pain point here is regarding short commit messages: 74% of the volunteers and 81% of the employees write commit messages longer than 50 characters.

4 THREATS TO VALIDITY

First, we rely on our approach to verify whether a contributor is an employee or a volunteer. We made use of a flag available in the pull-request to make this decision. We acknowledge that this can be a threat. To minimize this, we manually investigated the affiliation of the contributors. We found that two members classified as volunteer presented GitHub as their organizations. We further analyzed their profile, and found that they left GitHub and are now working in other companies. For those classified as employee members, all listed themselves as GitHub staff in their profile. Second, as we analyzed only five projects from the same company, we understand that the results cannot be generalized. However, a small sample enabled us to better experiment our approach. Finally, our strategy to identify test cases might introduce false-positive and false-negatives. However, we believe that, even in the presence of false-positive, our main finding (i.e., few pull-requests are accompanied with test cases) will not be challenged. We leave a detailed analysis of these corner cases for future work.

5 RELATED WORK

Zhou *etal.* [16] studied how industry involvement influence the onboarding of developers. Homscheid and Schaarschmidt [7] investigated the drivers that explain organizational and community turnover intentions of volunteer developers who are paid by third-party companies. Atiq and Tripathi [2] explored how the developers perceive the differences of rewards in OSS projects, and found that OSS projects where only some people get directly paid may fail if they are mismanaged. Riehle *et al.* [12] compared "paid" and "volunteer" work in OSS projects and found that around 50% of the contributions have been paid work. They considered that any contribution made from Monday to Friday, from 9am to 5pm are paid contributions. Differently from Riehle and colleagues, we compared the developers that are employees of the company that owns the project. In a preliminary study [4], we found that both employees and volunteer developers are rather very active in the analyzed projects, when it comes to pull-requests submitted

6 CONCLUSIONS

In this paper we investigate how the contributions form employees and volunteers differ in a number of metrics. When considering the number of accepted and rejected contributions, volunteers face 26× more rejections than employees. On the other hand, few employees are responsible for the majority of the the software development. Volunteers have to wait, on average, 11 days to have a patch processed (employees wait 2 days, on average). 92% of the dormant pull-requests (e.g., pull-requests that take too long to be processed) were submitted by employees. Finally, we observed that the best practices are not systematically followed; the best practice that had the most adherence is commit messages written in English.

ACKNOWLEDGMENTS

This work is supported by CNPq #406308/2016-0; PROPESP/UFPA and FAPESP #2015/24527-3).

REFERENCES

[1] A. Alali, H. Kagdi, and J. I. Maletic. What's a typical commit? a characterization of open source software repositories. In *ICPC'08*, pages 182–191, 2008.
[2] A. Atiq and A. Tripathi. Impact of financial benefits on open source software sustainability. In 37^{th} *ICIS*, 2016.
[3] N. C. Borle, M. Feghhi, E. Stroulia, R. Greiner, and A. Hindle. Analyzing the effects of test driven development in github. *Empirical Software Engineering*, 2017.
[4] L. F. Dias, I. Steinmacher, and G. Pinto. Who drives company-owned oss projects: Employees or volunteers? In *V Workshop on Software Visualization, Evolution and Maintenance*, VEM, page 10, 2017.
[5] G. Gousios, M. Pinzger, and A. v. Deursen. An exploratory study of the pull-based software development model. In *ICSE '14*, pages 345–355, 2014.
[6] G. Gousios, A. Zaidman, M. D. Storey, and A. van Deursen. Work practices and challenges in pull-based development: The integrator's perspective. In *ICSE'15*, pages 358–368, 2015.
[7] D. Homscheid and M. Schaarschmidt. Between organization and community: investigating turnover intention factors of firm-sponsored open source software developers. In *WebSci '16*, pages 336–337. ACM, 2016.
[8] A. Lee and J. C. Carver. Are one-time contributors different? a comparison to core and periphery developers in floss repositories. In *ESEM 2017*, pages 1–10, Nov 2017.
[9] G. Pinto and F. Kamei. The census of the brazilian open-source community. In *Open Source Software: Mobile Open Source Technologies*, pages 202–211, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
[10] G. Pinto, I. Steinmacher, L. F. Dias, and M. A. Gerosa. On the challenges of open-sourcing proprietary software projects. *Empirical Software Engineering*, 2018.
[11] G. Pinto, I. Steinmacher, and M. Gerosa. More common than you think: An in-depth study of casual contributors. In *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016, Suita, Osaka, Japan, March 14-18*, pages 112–123, 2016.
[12] D. Riehle, P. Riemer, C. Kolassa, and M. Schmidt. Paid vs. volunteer work in open source. In *HICSS ' 14*, pages 3286–3295, Jan 2014.
[13] E. A. Santos and A. Hindle. Judging a commit by its cover: Correlating commit message entropy with build status on travis-ci. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, 2016.
[14] I. Steinmacher, G. Pinto, I. Wiese, and M. Gerosa. Almost there: A study on quasi-contributors in open-source software projects. In *ICSE'18*, pages 358–368, 2017.
[15] P. Weissgerber, D. Neu, and S. Diehl. Small patches get in! In *2008 International Working Conference on Mining Software Repositories*, MSR '08, pages 67–76, 2008.
[16] M. Zhou, A. Mockus, X. Ma, L. Zhang, and H. Mei. Inflow and retention in oss communities with commercial involvement: A case study of three hybrid projects. *ACM TOSEM*, 25(2):13, 2016.