RESEARCH

Who Drives Company-Owned OSS Projects: Internals or Externals Members?

Luis F Dias[1], Igor Steinmacher[2,3] and Gustavo Pinto [4*]

*Correspondence: gpinto@ufpa.br
[4]Federal University of Pará, Faculty of Computing, Belém, Brazil
Full list of author information is available at the end of the article

Abstract

Open source software (OSS) communities leverage the workforce of volunteers to keep the projects sustainable. Some companies support OSS projects by paying developers to contribute to them, while others share their products under OSS licenses, keeping their employees in charge of maintaining the projects. In this paper, we investigate the activity of internal (employees) and external (volunteers) developers in this kind of setting. We conducted a case study using a convenience sample of five well-known OSS projects: `atom`, `electron`, `hubot`, `git-lfs`, and `linguist`. Analyzing a rich set of ∼12k contributions performed by means of pull requests to these projects, complemented with a manual analysis of ∼500 accepted pull requests, we derived a list of interesting findings. For instance, we found that both internal and external developers are rather active when it comes to submitting pull requests and that the studied projects are receptive for external developers. Considering all the projects, internal developers are responsible for 43.3% of the pull requests performed (external developers placed 56.7%). We also found that even with high support from the external community, employees still play the central roles in the project. We also found that the majority of the external developers are casual contributors (developers that placed only a single contribution to the project). However, we also observed that some external members play core roles (in addition to submitting code), like triaging bugs, reviewing, and integrating code to the main branch. Finally, when manually inspecting some code changes, we observed that external developers' contributions range from documentation to complex code. Our results can benefit companies willing to open-source their code, and developers that want to take part and actively contribute to company-owned code.

Keywords: Company-Owned OSS Projects; Employees; Volunteers

1 Introduction

Open source software (OSS) is one of the cornerstones of modern software development practice. Many existing software projects rely on OSS solutions either at compile time (*e.g.,* build tools or testing tools) or runtime (*e.g.,* webservers or databases). In spite of its ubiquitousness, several OSS projects rely on a single contributor to perform most of their needed tasks [1]. Due to this grim scenario, it is not uncommon to see core developers becoming tired and abandoning their own software projects [2].

To alleviate this situation, recently many software companies started to support open-source activities. For instance, open-source programming languages such as

Swift[1] and Scala[2] have their development process primarily driven by employees of a software company (Apple and Typesafe, respectively). In fact, there is a recurrent belief that most of the OSS contributions software are made by paid developers. As a recent article pointed out, *"More than 80 percent of [the Linux] kernel development is done by developers who are being paid for their work."*[3] While commercial contributions to the Linux kernel have been widely acknowledged, in a large-scale study of more than 9,000 OSS projects, Riehle and colleagues [3] observed that about 50% of the OSS contributors are actually paid ones. However, in their work, the authors consider "paid developers" the ones that performed commits from 9am to 5pm, local time. Using this simple rule, students, unemployed, or workers with flexible time schedules could be wrongly sampled as "paid developers". Therefore, we believe that more systematic approaches should be employed to shed additional light on the proportion of paid/non-paid developers. There are at least two reasons that support our claim:

1 If there are, indeed, too many paid developers, OSS communities may need to better explore these workforces. For instance, instead of concentrating too many paid developers in one single OSS project, OSS communities could try to gather some paid developers to OSS projects that are more in need.
2 On the other hand, if there are too few paid developers, this finding might not only refute previous studies, but yet can be used to better motivate software companies to support OSS projects.

It is important to note that the source of payment can vary greatly. For instance, one can get paid to fix a bug via a crowdsourcing system, whereas others can be full-time OSS contributors. In this study, we pay particular attention to developers that contribute to **Company-Owned OSS Projects**.

Company-Owned OSS Projects: This term refers to software companies that started and curated OSS projects in a private environment, but later on decided to open source them. Therefore, the OSS project that was previously restricted to the company's employees could now potentially receive contributions from contributors that are not anyhow affiliated with the given company.

This transition from proprietary to open source is particularly relevant to our work. Although proprietary from birth, the software companies that built these software projects have perceived benefits that motivated them to open source their software [4]. In order to differentiate the developers that are paid by the companies to work in the OSS projects and the OSS contributors that contribute for personal reasons, throughout this paper, we refer to contributors that work for the company that open-sourced the project as "internal developers". Developers that do not work for the company that open-sourced the given OSS project are referred as "external developers". More technical details on how we differentiate external and internal developers can be seen at Section 2.2.2.

[1] `https://github.com/apple/swift/`

[2] `https://github.com/scala/scala`

[3] `https://www.linuxfoundation.org/news-media/announcements/2015/02/linux-foundation-releases-linux-development-report`

In this paper, we extend a previous analysis [5], bringing a multi-case study investigating the contribution behavior of pull requests provided by internal and external developers in OSS projects. We used a convenience sample, composed by 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 an internal or external one (more details at Section 2). Through a set of quantitative and qualitative analysis, this paper makes the following contributions:

- We provide evidence that there is a workforce of developers who are external to the company who opened the code contributing to the project, creating a community that extends the boundaries of the company. The number of external developers can be up to 32× greater than internal ones.
- We show that, although the external community is engaging, external members face a hard time to get a contribution accepted. In 4 out of the 5 studied projects, most of the rejected pull requests were made by external developers. In terms of time taken to process a pull request, on average, externals take 11.37 days to be processed. Internals, on the other hand, take 2.61 days.
- We find that internal developers still play a crucial role in the project, playing the integrator role in two of the analyzed projects. However, external members are also acquiring this role. In project `hubot`, for instance, ∼80% of the team of integrators is composed by external developers.

2 Method

In this section, we report the studied projects (Section 2.1) and research approach, detailed according to our research questions (Section 2.2).

2.1 Studied projects

We provide an in-depth investigation of the contributions (*i.e.*, a pull request) made to five well-known OSS projects. They are:

- `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 ∼7 years of historical records. GitHub started its development in 2011[4], and open-sourced it in May 2014[5].
- `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 ∼5 years of historical records. GitHub started its development in March 2013[6], and open-sourced it in October 2015[7].
- `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

[4] `https://github.com/atom/atom/commit/3a09528a62f29e86bc15140a13d1bdbd9322e0e9`

[5] `http://blog.atom.io/2014/05/06/atom-is-now-open-source.html`

[6] `https://github.com/electron/electron/commit/e451d9212179197b88abeb752602de3859bb1765`

[7] `https://www.infoworld.com/article/2995384/application-development/easy-cross-platform-app-dev-with-githubs-electron.html`

written in JavaScript, and has ∼7 years of historical records. GitHub started its development in August 2011[8], and open-sourced it in October 2011[9].

- `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 ∼5 years of historical records. GitHub started its development in September 2013[10], and open-sourced it on April 2015[11].
- `linguist`, a library to detect blob languages. It has 5,600 commits, ∼2,400 pull requests, 684 source code contributors, ∼5,400 stars, and ∼2,000 forks. It is mostly written in Ruby, and has ∼7 years of historical records. GitHub started its development in May 2011[12], and open-sourced it in October 2015[13].

When analyzing the software history of these projects, we perceived that all of them but `linguist` started as a stand-alone software project. `linguist`, on the other hand, started as a unification of code scattered around the whole software system. Such a pattern of open-sourcing software projects was already reported elsewhere [4].

Figure 1 shows a distribution of additional characteristics of these projects.

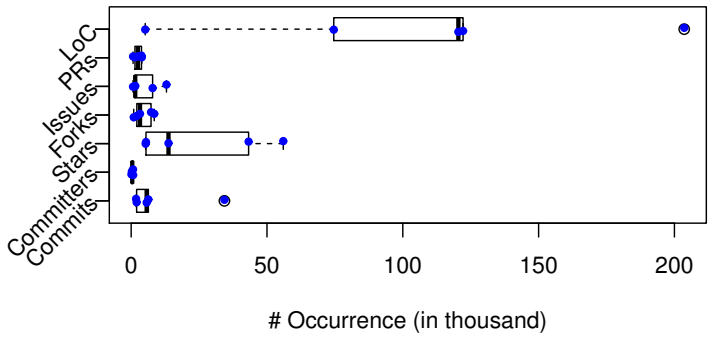

Figure 1 Characteristics of the analyzed projects

2.2 Overall approach

We followed a mix-methods approach, combining quantitative and qualitative research method. In this section we will present the common ground for all the research questions—including pull requests data collection, and how internal and external developers are classified—and, afterwards, we dive in the details of each specific RQ.

2.2.1 Pull request collection

The data reported in this paper is based on pull requests that were performed from the very beginning of the studied projects, up to January 2018 — when we

[8] `https://github.com/hubotio/hubot/commit/b253e94e051c017eb7c8c0101c30a24f0851a499`
[9] `https://blog.github.com/2011-10-25-say-hello-to-hubot/`
[10] `https://github.com/git-lfs/git-lfs/commit/d8f780329b64e789553bc8ccccfb993ebc430325`
[11] `https://blog.github.com/2015-04-08-announcing-git-large-file-storage-lfs/`
[12] `https://blog.github.com/2011-06-27-linguist/`
[13] `https://github.com/github/linguist/commit/559097ed6bae9b58987f969937f6c1de622b6487`

collected data. All data used in this study is available online at `https://github.com/fronchetti/JBCS-2018`.

We started our study by investigating all performed pull requests. A pull request can be found in three different stages:

- *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).

We studied the contribution behavior of internal and external developers taking into account each possible stage of a pull request. Additionally, we investigated other characteristics associated with the pull request, such as:

- the time taken to process a pull request.
- the number of comments during the code reviews per pull request;
- the number of commits per pull request;
- the number of changes (e.g., additions/deletions) per pull request;

2.2.2 Internal and External classification

Since the analyzed projects are developed by (and maintained at) GitHub, we reduce false positives by taking advantage of GitHub features used to identify developers roles. Within GitHub organizations, one coordinator can set the `site_admin` flag true for another user. If enabled, this flag promotes an ordinary user to be a site administrator. According to GitHub official documentation, a site administrator can *"manage high-level application and VM settings, all users and organization account settings, and repository data"*[14]. Therefore, for each pull request investigated, we verified whether the author has the `site_admin` flag enabled. If so, we marked she as *internal*; *external* otherwise.

To avoid false negatives (a paid developer that does not have its `site_admin` flag enabled), we analyzed the public profiles (e.g., Github affiliation, LinkedIn information, personal web page, among other sources) of the top-10 contributors (either internal or external). From the 48 profiles analyzed (2 members appeared in 2 different projects), we found 12 that worked for GitHub previously, but were not categorized as staff members. We manually identified these users as internal developers for our analysis. This misidentification is a potential threat and is further described in Section 6.

2.3 Research Question

To guide our research, we investigated the following important but overlooked research questions:

RQ1. Are OSS contributions mostly made by internal developers?

Rationale: This exploratory research question guides our case study on GitHub company-owned OSS projects. It also provides evidence to understand the role that the external developers play in this kind of endeavor.

[14] `https://enterprise.github.com/security`

Approach: To answer this RQ we quantitatively compared the number of internal and external contributors, as well as the number of pull requests submitted by them. In addition to characterizing and discussing the values using descriptive statistics, we compared the evolution of the number of pull requests submitted monthly by external and internal members, in a per project basis. It is important to mention that we computed the number of pull requests submitted per state (open, closed, merged). Since we compared the number of pull requests per month by two different samples, we applied the Wilcoxon signed-rank test for paired samples [6] to perform this comparison. We used Cliff's delta to verify how often values in one distribution are larger than values in another distribution. The thresholds are defined as follows: $delta < 0.147$ (*negligible*), $delta < 0.33$ (*small*), $delta < 0.474$ (*medium*), and $delta >= 0.474$ (*large*) [7].

We also graphically reported the distributions to enable the visualization of the temporal evolution of contributions. The results for this question are presented throughout Section 3.1.

RQ1.1. Are internals the top contributors of company-owned OSS projects?

Rationale: In this question we are aimed to provide a fine-grained perspective about the involvement of the contributors of company-owned OSS project. Answers to this question will further substantiate the role that our subjects play.

Approach: To answer this question we, firstly, analyzed the top-10 contributors for each project to check how many of them are internal, and how many are external members. Then, we compared internals and externals in terms of the number of pull requests per contributor in each project.

RQ2. Who faces a harder time to get the contributions accepted?

Rationale: In this research question, we focus our interest on understanding the how pull requests of internal and external are received. We focused on (i) acceptance and rejection rates; and (ii) on the priority given to the pull requests. As the literature suggests, it is not always easy to contribute to open source projects [8]. We, therefore, explore whether external developers are facing a harder time in terms of rejections and time to process when compared to their internal peers. If that is the case, answers to this question might help improve how company-owned OSS projects treat external developers.

Approach: We built upon the results of the comparisons made for RQ1 to understand the acceptance rate (number of merged pull requests versus the total submitted pull requests) for internal and external members. To compare the time to process, we first computed the number of days from the submission date until the decision date (when the pull request was closed or merged). Then, we compared this characteristic for pull requests submitted by internal and external members. We also used Mann-Whitney-Wilcoxon (MWW) tests [6] and, as for RQ1, Cliff's delta effect-size measures [9] to perform this comparison.

RQ3. Are externals more participative in the pull request review cycle?

Rationale: This research question explores the degree of involvement of externals and internals in company-owned OSS projects in terms of (1) **commenting/dis-**

cussing pull requests and (2) playing the **integrator role**. Commits or pull requests are not the only ways to measure participation. In fact, contributors might provide comments to pull requests under review as an attempt to contribute to the project. On GitHub, anyone can freely provide comments to a pull request, regardless if the GitHub user has contributed before to the project. Another facet of participation regards integrating pull requests. Since processing pull requests is a notorious activity that only experienced contributors are willing to perform [10], it is more likely that internal developers should conduct this process. However, if external developers are also playing this role, this might indicate that the company-owned OSS project succeeds in decreasing the barriers for external developers to join the project.

Approach: To analyzed these two participation perspectives of internal and external members in the code review cycle, we collected the number of comments per pull request, classifying them as comments made by internal or external members. We also verified who was responsible for integrating the pull requests submitted: internal or external members. We characterized this aspect in terms of the number of pull requests integrated by internal and external members, and the number of internal and external members who played the integrator role. We used descriptive statistics and graphics, in addition to the MWW test and Cliff's delta effect-size to compare the involvement of internal and external members as commenters and integrators.

RQ4. What are the characteristics of the contributions made by external developers?

Rationale: We are intended to understand what are the kinds of contributions performed by external members. We complemented this analysis with an investigation over the differences of pull requests placed by internal and external members in terms of the size of the commits, including the number of files changed and code churn. Answers to this question might enable companies to have a better picture of what to expect from the external community. Moreover, the literature is particularly rich when it comes to changes made by internal developers [11, 12, 13].

Approach: We selected a representative sample of a small number of pull requests that reflect the larger population. We selected 334 random pull requests made at `atom` for manual analysis, which represents a confidence level of 95% with a ±5% confidence interval. We also validated this analysis with another manual analysis in a random sample of 150 pull requests accepted at `hubot`. The qualitative analysis was conducted in parallel by two researchers, who investigated the pull requests individually. We also quantitatively compared the characteristics of the pull requests placed by internal and external members in terms of number files changed, added lines, deleted lines, and the number of commits per pull request. We considered each pull request as an observation and, once again, we used MWW tests [6] and Cliff's delta effect-size measures [9] to compare the groups. The results for this question are presented throughout Section 3.1.

3 Results

In this section, we discuss the results of our study organized in terms of the research questions.

3.1 RQ1. Are OSS contributions mostly made by internal developers?

Generally speaking, both internal and external developers are rather active when it comes to submitted pull requests, as it can be observed in Table 1. On one hand, internals contribute more pull requests on `atom` and `git-lfs`; on the other hand, external developers made a higher number of pull request in `electron`, `hubot`, and `linguist`. For `hubot` and `linguist`, external developers are responsible for more than 75% of the pull requests in the project. If we consider all projects, we found 5,895 pull requests provided by internal developers (43.3%) and 6,266 by external ones (56.7%). However, the number of contributors greatly differ between internals and externals, as it can be observed in Table 1. As an extreme case, project `electron` has 681 external contributors, and only 21 internal (while the number of contributions made by external developers is almost two times greater than those made by internal developers). That is, although the number of external developers is up to 32× greater than internal ones, most of externals developers perform few contributions.

Table 1 pull request submitted by external and internal developers

Projects	external		internal	
	#contributors	#pull requests	#contributors	#pull requests
`atom`	365	1,546	35	2,206
`electron`	681	2,442	21	1,385
`git-lfs`	82	435	6	938
`hubot`	241	557	20	131
`linguist`	645	1,860	29	579

To provide a more detailed perspective, Figure 2 depicts the evolution of pull requests, grouped by their states (open, closed, and merged) at collection time (Jan 2018). Each observation corresponds to the total number of pull request submitted per month by each type of member (internal and external). The same data was used to statistically compare (p-values and effect-size values) submissions by internal and external members, which is shown in Table 2. Our effect-size test follows the order internals, then externals. Therefore, negative values indicate effect size greater to the external developers. Positive values, otherwise.

Table 2 Comparing monthly number of pull requests submitted by internal and external members. Green cells indicate large effect size, whereas yellow cells indicate medium effect size. Projects `hubot`, `electron`, and `git-lfs` have NA in their p-values and effect size due to their small sample size.

Projects	Open PRs		Closed PRs		Merged PRs	
	p-value	$delta$	p-value	$delta$	p-value	$delta$
`atom`	<0.001	−0.867	<0.001	−0.377	<0.001	+0.599
`electron`	NA	NA	<0.001	−0.849	<0.001	−0.220
`git-lfs`	NA	NA	0.190	−0.127	<0.001	+0.540
`hubot`	NA	NA	<0.001	−0.956	<0.001	−0.573
`linguist`	0.002	−0.918	<0.001	−0.947	<0.001	−0.616

From the figures, it is first possible to see that the maintainers do a great job in processing pull requests, given the small number of pull requests kept open. Projects `electron`, `git-lfs` and `hubot` present low rates of open pull requests, 0.88%, 0.13%, and 0.08%(!), respectively. For the latter, at the time of data collection, only 3 pull requests were left open.

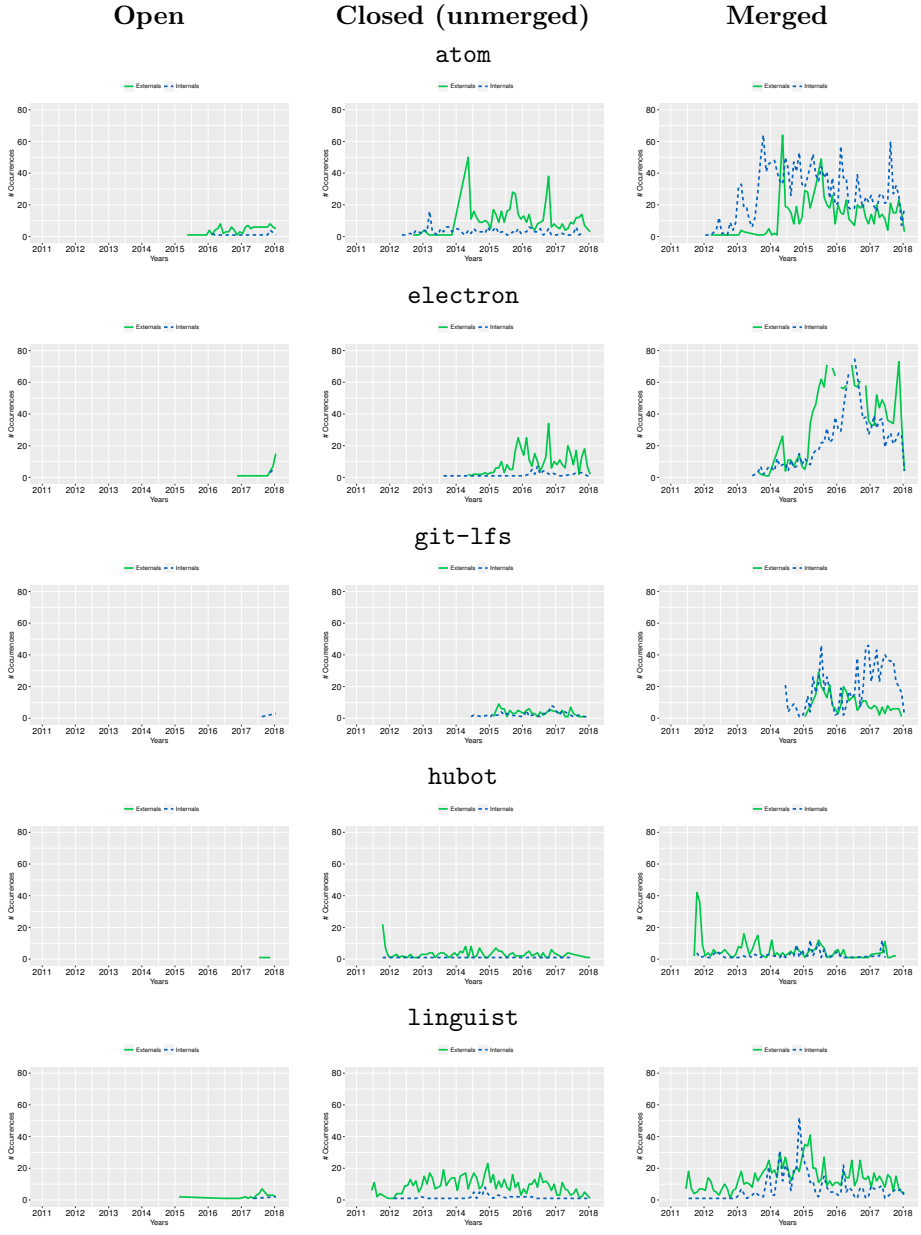

Figure 2 Time series for open, closed/unmerged, and closed/merged pull requests submitted by internals and externals.

RQ1.1. Are internals the top-contributors of company-owned OSS projects?
By analyzing the top-10 contributors for each project, we could observe that the "top contributor" of all projects are internal developers. As it can be observed in Table 3, in only one of the projects (`git-lfs`) the number of external developers is greater than the number of internal developers in the top-10 (6 externals, 4 internals). This finding suggests that externals are well-participative. However, even in this case, by analyzing the code-churn, the top-2 developers (both internal) are by far the main contributors of the `git-lfs` project (top-1: 124,197 additions and 75,831 deletions; top-2: 89,065 additions and 74,576 deletions; sum of top-3 to top 5: ≈61,300 additions and ≈33,600 deletions)

Table 3 Number of external and internal developers among top-10 contributors

Projects	# external	# internal
atom	1	9
electron	4	6
git-lfs	6	4
hubot	4	6
linguist	3	7

Finally, we also analyzed the number of pull requests placed per contributor, as shown in Figure 3. It is possible to observe that the small number of internal contributors place a higher amount of pull requests than external developers. It is also straight-forward to notice that the external contributors' population is mostly composed of casual contributors [14, 15], that is, developers that contributed only once to thr project. Table 4 brings the absolute number and the percentage of internal and external casual contributors per project. Overall, 76% of the external contributors of the analyzed projects made only one pull request to the project. This finding complements the study of Pinto and colleagues [14], which suggests that casual contributors are responsible for 49% of the whole population of contributors. More interestingly, we observe that there are internal developers that only contributed once (e.g., for `hubot`, 65% of the internals are casual).

Table 4 Population of casual contributors, grouped by external and internal developers

Projects	external		internal	
	# casuals	%	# casuals	%
atom	269	74%	5	14%
electron	480	70%	6	29%
git-lfs	55	67%	6	33%
hubot	200	82%	13	65%
linguist	534	83%	9	31%
TOTAL	1538	76%	39	35%

RQ2. Who faces a harder time to get the contributions accepted?
As aforementioned, we focus on understanding how pull requests of internal and external are received. We analyzed the reception in terms of **acceptance rate** and **time to process** the pull requests from internal and external members.

Regarding the **acceptance rate**, when studying the merged pull requests (the accepted ones) in RQ1, we can see that both groups are also fairly active in all the five projects analyzed. We can observe, though, different patterns depending on the project. For example, for `linguist`, we can see that the number of pull requests from externals outperforms those from employees by far, and for every

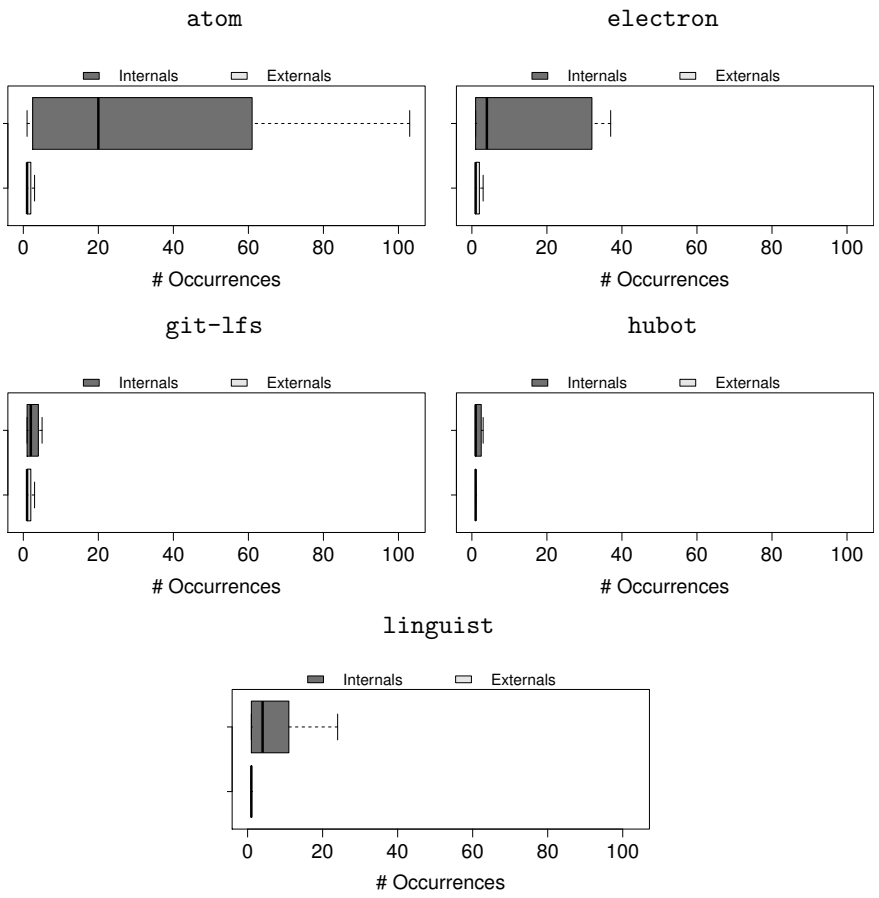

Figure 3 Distribution of number of pull requests made by internal and external developers (we dismissed outliers to ease visualization).

month. However, analyzing the closed but unmerged pull requests (the ones that were not accepted), we could notice that many external developers are having a hard time attempting to get their contributions accepted. This is noticeable in the second column of graphics in Figure 2. In Table 2, we could confirm that most of the unmerged (closed) pull requests were done by external developers for 4 out of 5 projects (p-value ≤ 0.001), with a medium or large (negative) effect size. A possible explanation is that employees work on critical and follow project directions (defined inside the company), while externals submissions are, sometimes, motivated by specific needs, not necessarily aligned with the project's direction.

In terms of **time to process** the pull requests, we analyzed the number of days taken between when the pull request was opened to when the pull request was merged. On average, pull requests filled by externals take 11.37 days to be processed (min: 0, max: 1,144, 3rd quartile: 5, std deviation: 55). In comparison, pull requests from internals take 2.61 days (min: 0, max: 558, 3rd quartile: 1, std deviation: 18). Figure 4 shows the average number of days for each studied project.

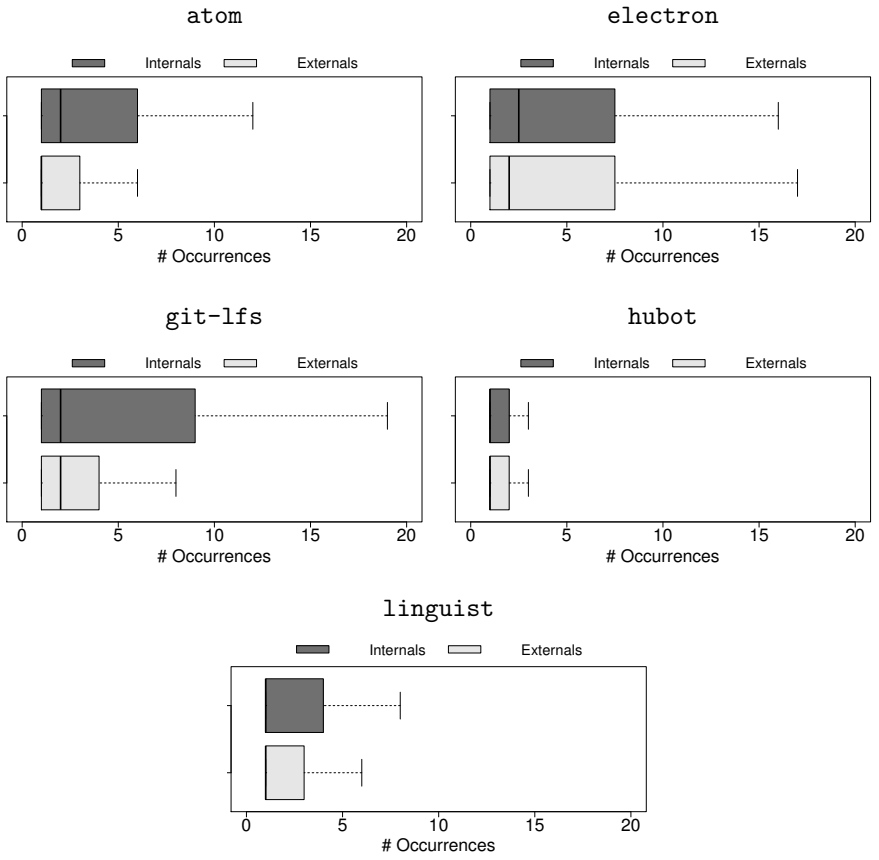

Figure 4 Average of days needed to process a pull request

As we can see in the figure, for all studied projects, on average, pull requests submitted by internals are process faster than the ones submitted by externals; a small effect size confirmed this trend ($p-value = 0.001$, $delta = 0.243$). In particular, projects `hubot` and `linguist` are the ones that take more time to process pull

requests, either from internals (333 and 426 days for `hubot` and `linguist`, respectively) or externals (1,144 and 832 days for `hubot` and `linguist`, respectively). To better understand why these pull requests made by externals are taking too much time to be processed, we investigated the ones that lasted the most.

The pull request #678 submitted to `hubot` project was aimed to improve the documentation (it adds 32 lines in a Markdown file); five commits had been made to this pull request. Although project maintainers needed some time to review the contribution (the final modification suggested was about 300 days after the pull request was created), it seems that the pull request was forgotten, and only 2 years after the last change was made, another project maintainer passed through the pull request and merged the patch. On the other hand, the pull request #2070 submitted to the project `linguist` is a bit more complex. It was aimed to introduce PEP8 support, which is the code convention for writing Python code. Similar to the previous pull request, in this one, the maintainers also seem to forgot to follow up with the code review. The external member brought back the attention to this pull request, mentioning: "*I'm recalling this pull request has been open for over a year now (wow, nearly two, time flies), is there anything I can do to help it being merged into master aside from fixing the conflicts that have arisen since its opening ?*". Four months after this message, another maintainer provided additional comments, and one month after the pull request was merged.

RQ3. Are externals more participative in the pull request review cycle?

In this RQ we are interested in exploring how internal and external members participate in the process by both **commenting/discussing pull requests** and acting as **integrators**.

We first investigated how internal and external contributors differ in terms of the **number of comments** received during the code review of a pull request. Figure 5 shows the distribution of this metric.

As we can see, both groups receive comments on their pull requests, with external developers receiving more in most of the projects. Although internal developers might be more aware of project domain, the integration process, and their peers, they face a similar pull request review process (in terms of receiving comments), when compared to external developers. By analyzing Table 5, it is possible to confirm what is shown in Figure 5: external developers receive more comments than internal developers (p-value < 0.01 for four out of 5 projects, with small and medium effect-size). This finding, to some extent, shows that our studied projects welcome external developers, by providing comments, which might be used for reviewing, guiding, and supporting developers getting their changes merged.

Table 5 Statistical results: Comments received by internal and external contributors' pull requests. Green cells indicate large effect size, yellow cells indicate medium effect size, and red cells indicate small effect size.

Projects	p-value	*delta*
`atom`	<2.2e-16	−0.223
`electron`	<2.2e-16	−0.419
`git-lfs`	<2.2e-16	−0.334
`hubot`	0.8572	0.010
`linguist`	<2.2e-16	−0.390

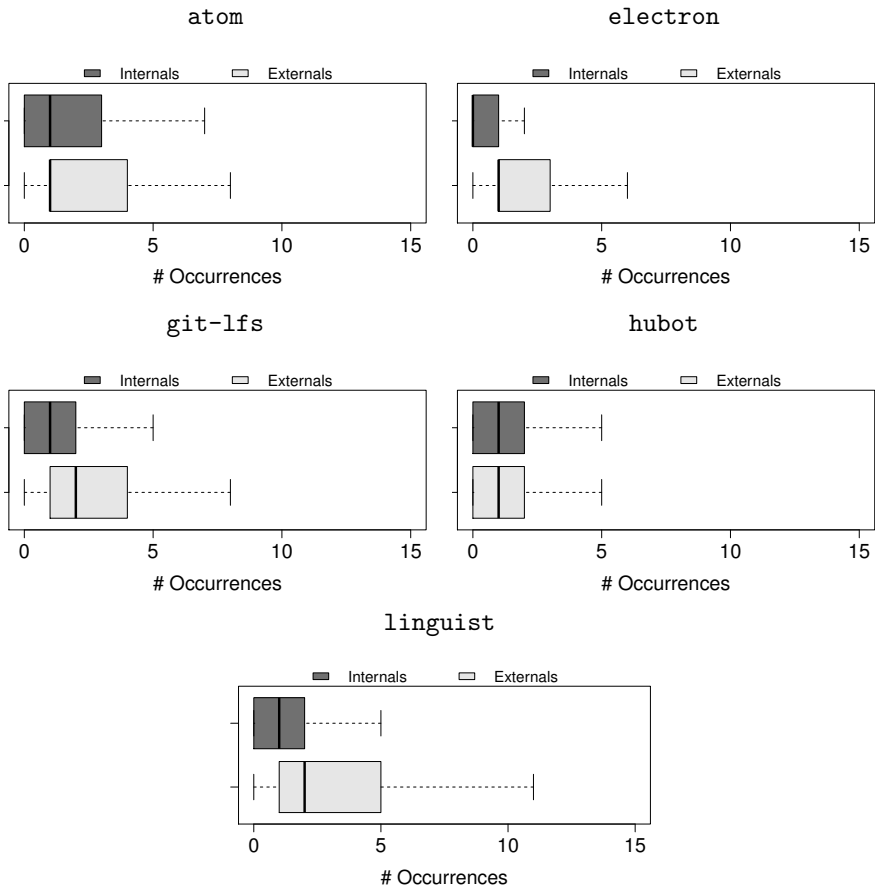

Figure 5 Number of comments received per pull request (we dismissed outliers to ease visualization)

To understand the participation in the review cycle, we also studied whether the **integrator role** (the developer that integrates a pull request play) is performed by an internal or by an external member. Figure 6 shows the percentage of pull request processed by internals and externals members.

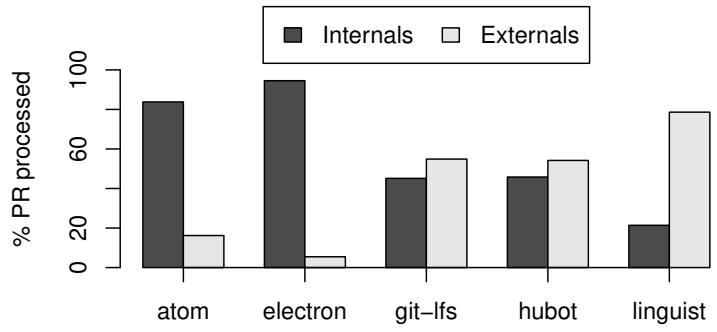

Figure 6 Percentage of pull requests integrated by external and internal developers

As we can see, the majority of the pull requests submitted to projects `atom` and `electron` are processed by internal developers (83% and 94%, respectively). However, for the remaining projects, the number of pull requests processed by external developers is indeed greater than the ones processed by internal developers. In particular, project `linguist` is an extreme example, with 78% of the pull requests being processed by external developers. However, after a closer look at the data, we found that few integrators are responsible for processing the majority of the pull requests. For instance, two internal integrators processed 85% of the pull requests submitted to project `electron`. Figure 7 shows a different perspective: the percentage of unique integrators that are internal or external developers.

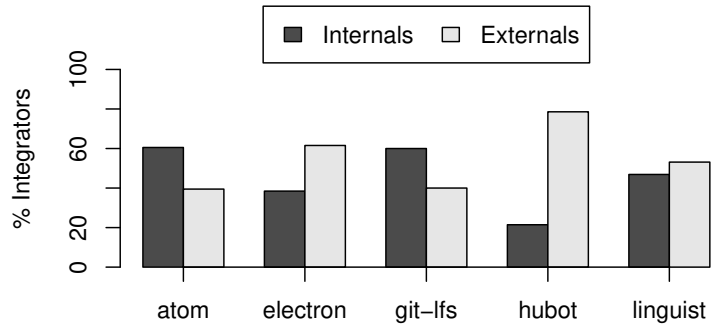

Figure 7 Percentage of unique integrator per project

The number of unique integrators for both kind of contributors is roughly similar in four out of the five analyzed project (*e.g.,* the `linguist` project has 15 internal integrators and 17 external). The only exception to this trend is the project `hubot`, in which 11 (78%) of the integrators are external developers (which corroborates

with the findings of Section 3.1, that indicates a large proportion of internals are casual contributors for this particular project). Regarding the amount of work devoted to each kind of contributor (either internal or external), we observed that internal integrators processed more pull requests on projects `atom`, `hubot`, and `electron`. In particular, internal integrators of project `electron` processed 28× more pull requests than their counterparts. Moreover, although the project `hubot` has more unique external integrators (11 externals and 3 internals), internals integrators are responsible for managing the majority of the pull requests (internals integrators processed 3× more than external ones). On the other hand, on projects `linguist` and `git-lfs`, external integrators processed more pull requests than internals (3.26× and 1.82×, respectively).

Additionally, we also investigated the proportion of pull requests submitted by internals that are also processed by internals (and vice-versa). We observed that 86.4% of the pull requests submitted by internals are also processed by internals. In comparison, 55.4% of the pull requests submitted by externals are also processed by externals.

RQ4. What are the contributions' characteristics made by externals?

To better understand the characteristics of the accepted contributions, we conducted a qualitative analysis aimed at investigating the reasons for pull request acceptance, in particular, the ones proposed by external members.

For the `atom` project, before creating a pull request, internal developers create an issue that describes what are the project needs. Therefore, most of the pull requests proposed are accepted because internal developers were *expecting* it. For externals, pull requests that fix documentation problems are the most common ones (we found 27 instances of them). Some example include: broken URL[15], not enough information[16], and code comments[17]. Notwithstanding, non-trivial code changes often come with a detailed description (images are common). We found a similar pattern for `hubot`. Most of the pull requests from external developers are related to documentation issues[18], although complex code changes exist[19]. Finally, these two projects seem to welcome external users: they not only answer most of the requests from external developers, but they also guide their contributions to an acceptable state (as mentioned before, providing comments to improve the pull request).

In addition, as presented in Figure 8, contributions from external developers are, in general, slightly shorter than internal ones in terms of lines added, lines removed and files changed. For `electron`, for example internal developers added 173,319 lines in total (mean=130.51 lines per pull request; median=19.5; q3=71.25; stdev=630.50) and changed 10,092 files (mean=7.60 files per pull request; median=3; q3=6; stdev=20.57), while external added 150,667 lines (mean=75.30 lines per pull request; median=12; q3=52; stdev=267.56) and changed a total of 8,067 files (mean=4.03 files per pull request; median=1; q3=4; stdev=10.52).

As one can observe in Table 6, in general, internal developers indeed include more files that external ones in all analyzed projects. For number of deleted lines,

[15] `https://github.com/atom/atom/pull/1929`
[16] `https://github.com/atom/atom/pull/2602`
[17] `https://github.com/atom/atom/pull/8452`
[18] `https://github.com/hubotio/hubot/pull/788`
[19] `https://github.com/hubotio/hubot/pull/489`

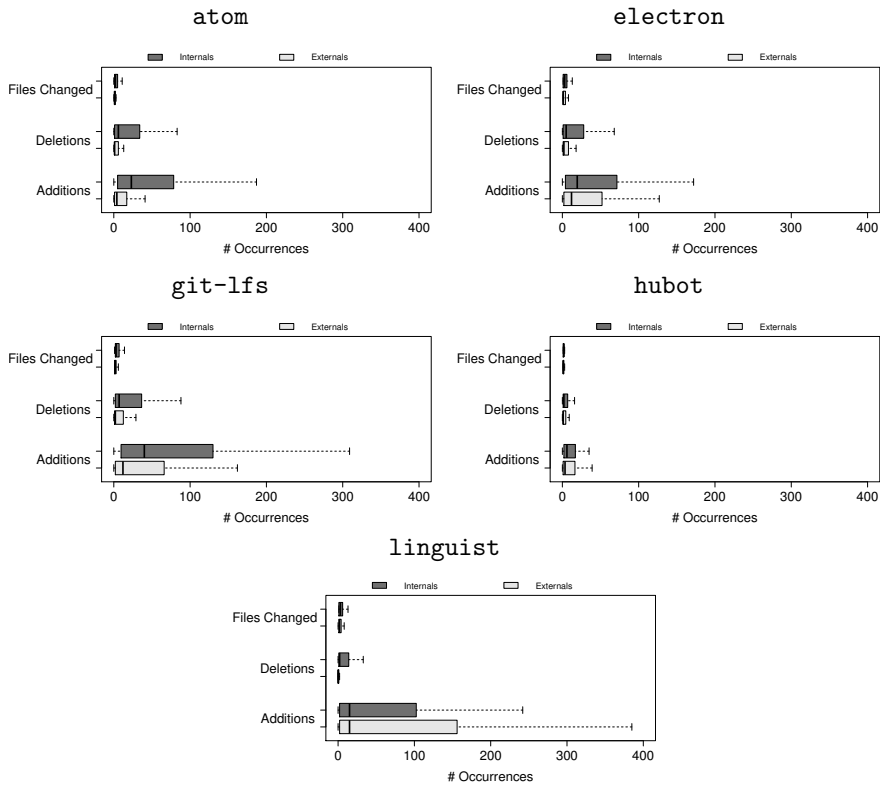

Figure 8 Files changed, lines removed, and lines added per pull request (we dismissed outliers to ease visualization)

this does not hold true for project `hubot`; for additions, there is no statistically significance for both `hubot` and `linguist`. Overall, we can see that both internal and external contributions are small (few files, and small additions and deletions). As noted elsewhere, smaller changes are more likely to be accepted [10] and can also reduce the chance of breaking the continuous integration build [16].

Table 6 Statistical comparison on changes submitted by internal vs. external developers. Green cells indicate large effect size, yellow cells indicate medium effect size, and red cells indicate small effect size.

Projects	Additions		Deletions		Files changed	
	p-value	*delta*	*p*-value	*delta*	*p*-value	*delta*
`atom`	<0.001	+0.413	<0.001	+0.363	<0.001	+0.390
`electron`	<0.001	+0.131	<0.001	+0.255	<0.001	+0.211
`git-lfs`	<0.001	+0.251	<0.001	+0.264	<0.001	+0.290
`hubot`	0.125	+0.091	0.024	+0.133	<0.001	+0.205
`linguist`	0.162	−0.042	<0.001	+0.453	<0.001	+0.155

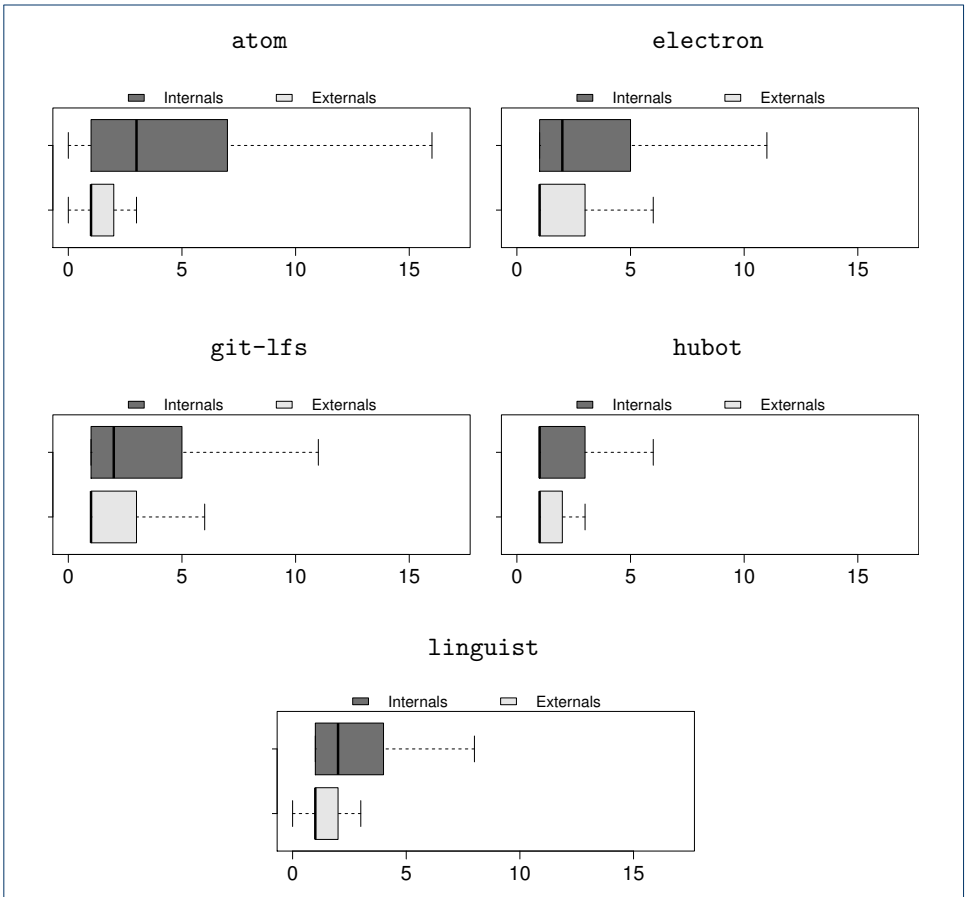

Figure 9 Number of commits per pull request (we dismissed outliers to ease visualization)

By observing Figure 9, we also notice that external developers' pull requests are also smaller in terms of the number of commits. Single-commit pull requests are rather common, accounting for more than 50% of the pull requests received from externals (overall, and for each project). This is expected since shorter contributions

(mainly documentation and typo fixes) are made in single files. For internals, we can observe a higher number of commit per pull request—which can be noticed by comparing the median and the whiskers. This was statistically confirmed for all projects (p-values $\ll 0.01$), with small effect-size for all projects, except for `hubot` in which we found a medium effect size ($delta$=0.350). of commits are not common. This finding suggests that both groups follow well-known guidelines for contributing to OSS (small commits and few commits per pull request [17, 10]).

4 Discussion

In this section, we summarize the main findings of this study (Section 4.1) and provide additional reflections on them (Section 4.2).

4.1 The main Takeaways

External developers are welcome. Our results showed that the external community is supporting the companies maintaining the project by means of contributing to them. In particular, we found cases which external members play crucial roles in the projects, such as reviewing and integrating pull requests. This could only be possible because the studied projects welcome externals members (which is not always the case of open-source software [18]). We further support this claim by inspecting welcoming-community features [20] available in the studied projects. All of the studied projects present a description, a README.md file, a Code of Conduct file, a CONTRIBUTING.md file, and a license file.

External developers still need guidance. Some projects tag the issues to make it easier for externals to find a task to solve (including `atom`, `electron`, and `linguist` which provide specific tags for newcomer-friendly tasks). However, given the high number of unmerged pull requests from external developers (Figure 2), external developers have to understand the project's direction and follow its guidelines when submitting a pull request; otherwise, their contributions are more likely not to be accepted [19].

Few External developers become long-term contributors. Even though we found external developers supporting the studied projects, few of them have a long-term contribution history (the only exceptions are the outliers). As one can observe in Figure 3, the majority of external developers place a single contribution to the projects and never show up again. For some projects (`hubot` and `linguist`, in particular), even internal developers do not place too many pull requests. However, when looking from a different perspective, the total number of pull requests placed by external developers is greater than those submitted by employees, as it can be noticed from Table 1. Similarly, there are projects with small participation from employees (although they company keep contributing to it). This result might indicate that the company-owned project is now a community effort.

External developers can wear the integrator hat. Although integrators are usually employees, we also found externals that play this role, which indicates a high involvement from the external community in company-owned OSS

[20] `https://opensource.guide/building-community/`

projects. However, when analyzing `atom`, we could find external developers who are in charge of triaging and commenting on issues (who are also among the top contributors). These externals describe themselves as "@atom community volunteer" or "@atom maintainer". Therefore, further research is needed to understand what are the actual roles played by external and internal developers in this kind of project. Figuring out the boundaries of responsibilities is an interesting future direction for this research that can benefit companies and communities.

4.2 Wrap up

From previous studies on casual contributors [14] and quasi-contributors [19], we found out that the main reason for a developer to place a contribution to a project is to "scratch his/her own itch". In many cases, this motivation was triggered by the company where the developer worked. We hypothesize that this can be the case for many contributors to these company-owned projects. Interestingly, we found cases in which developers voluntarily contribute, for a long period. It is the case of one of the top-10 contributors of `atom`, who, in his personal home page, mention that "*In my free time I contribute to Atom, GitHub's text editor, as one of the community maintainers of the project.*" We found similar when analyzing the top contributors of `git-lfs` and `electron`. This might suggest that altruism is still present in open-source communities.

However, we are not aware of the motivations that drive external contributors that volunteer to these projects. One can hypothesize that this can be a way to showcase their skills to the project maintainers, so they can be hired by the company. However, an interesting point of discussion is whether the company is indeed interested in hiring key or highly productive members of the external community. From the hiring perspective, observing potential candidates contributing to the project can be seen as a live screening process, in which the company can cherry pick good contributors. From a community perspective, taking "core external contributors" can harm the externals structure, since the role they play outside the company can change. Moreover, it is also important to understand the goals of the company when they open their code, and if they are willing to pay for someone who is already contributing voluntarily to the project. Although we did not investigate this specific point (using the community contributions as a hiring area), we believe that our findings might foster other researchers to conduct more research, especially from the perspective of the company willing to make that move.

5 Related Work

In this section, we discuss some of the studies that relate with the scope of this work.

5.1 Commercial Involvement/Paid Developers in OSS Projects

It is possible to notice an increase in the participation of companies in OSS and in the contributions of employees paid to work on OSS projects [3, 20]. Homscheid and Schaarschmidt [21] investigated the role of external developers who are paid by third-party companies ("firm-sponsored developers"). By conducting a survey with

Linux developers, they found that the perceived external reputation of the employing organization reduces turnover intention towards the company, and the perceived own reputation dampens turnover intention towards the OSS community. Atiq and Tripathi [22] explored how the developers perceive the differences in rewards in OSS projects, by analyzing their opinion on how the project's financial resources influence the progress of the project. By analyzing an open question sent to OSS developers, they found that OSS projects where only some people get directly paid may fail if they are mismanaged.

Riehle *et al.* [3] analyzed more than 5,000 active OSS projects, from 2000 to 2007, and found that around 50% of all contributions have been paid work. Their perspective is that any contribution made from Monday to Friday, between 9am and 5pm are paid contributions. However, as highlighted by Crowston [23], even employed developers are not paid directly by the projects to which they contribute, so from the project perspective, they are volunteers. Thus, differently from Riehle and colleagues, we analyzed the amount of effort put by the developers of the company that open-sourced the project – directly paid by the "owner" – comparing with the contributions made by any external developer. Our results showed that, for the analyzed projects, 45% of the pull requests are placed by internal developers (GitHub employees). The results seem to be in line with previous work, except for the fact that the concept of paid developers used previously, is not the same as the concept of external developers applied here.

In a previous work, we studied the challenges that software companies face when open sourcing their software products [24]. In this work, we studied 8 well-known proprietary projects that kept their software history while transitioning to open source. Two of these eight projects were also studied in this current work: `atom` and `hubot`. Analyzing the software history, we observed that external developers often onboard company-owned OSS projects in the very first weeks after open sourcing, but abandon few commits ahead (the so-called newcomers' wave). In this work, we also observed that the majority of External contributors are casual ones (e.g., have contributed at most with one commit). We also observed a burst in the number of issues and pull requests right after open sourcing the software project. In a follow-up study, we studied the reasons that motivated 50 company-owned OSS projects to delete their software history before going open source [4]. Among the reasons, we observed that code that contains sensitive information (e.g., user credentials) is one of the most common reasons for deleting the history, although other so far uncommon reasons such as the lawyers having to inspect each commit was also observed.

5.2 Casual Contributors Phenomenon

Some recent studies explore the casual contributors' phenomenon (or drive-by commits) in the context of social coding environments. Several authors have acknowledged the existence and the growth of this behavior [17, 25, 26, 14, 27, 28, 29]. It is found that this kind of behavior can be beneficial for both projects and developers [14]. We could observe that this phenomenon is also quite common in this scenario, accounting for 76% of the external contributors, reaching up to 83% for `linguist` project. This is larger than the results we have in a previous study [14], in

which we identified that casual contributors account for up to 61% of the contributors of open source projects written in JavaScript (4 out of the 5 projects analyzed here are written in JavaScript). Investigating the reasons behind this large number of casual contributors in this kind of project can be an interesting future direction.

6 Limitations

In a study such as this, there are always many limitations and threats to validity.

Our first limitation is regarding the number of projects studied. Although one might consider difficult to draw conclusions based on five projects owned by the same software company, we argue that our intention was not to picture a definitive landscape of company-owned OSS projects. Instead, our intention was two-fold: (1) to call the attention to this relevant yet not fully understood problem and (2) to help us to better evaluate the approach used to classify the contributors manually. With our approach, we expect similar analysis can be conducted in the future when other aspects of company-owned open-source projects become relevant.

Moreover, we rely on our inference algorithm to verify whether a contributor is an internal or external one. We made use of a flag (site_admin) made available in the pull request to make this decision. We acknowledge that this can be a threat, since even relying on this flag, it is possible that some developers had left the company previously, so they would be incorrectly identified. Still, we got in touch with GitHub support regarding this issue and they mentioned that "not every employee will have that flag set as some employees choose not to make their affiliation with the company known". To minimize this threat, we analyzed the profile of the top-10 external contributors (in terms of # of pull requests) and found that 12 of them left GitHub and were working in other companies. We classified these developers as external to conduct our analysis, reducing the threats.

For those classified as internal developers, all listed themselves as GitHub staff in their profile. Still, we got in touch with GitHub representatives whether this flag can be employed in other OSS projects, and they answered that "*The site_admin flag is only true for GitHub employees.*"

One might argue that we could differentiate paid and non-paid developers by looking at the email address used for their contributions (if it is a company email, then the developer is a paid one). We argue that many developers are free to choose whenever email account they want to use at the git repository. Therefore, a paid developer can also contribute with her personal email account (which would represent a false positive). We use the site_admin flag to mitigate this threat.

Another limitation is related to the GitHub API. We found some inconsistencies while mining data and metadata of the studied projects. For instance, in the API, some pull requests appear with strange characteristics such as zero additions, zero deletions in zero files[21], even though the original pull request on the web interface does have additions and deletions[22]. We found 1,107 pull requests with this characteristic. Instead of discarding them, we manually verified the number of changes in the web interfaced and fixed these numbers in our dataset. However, we also found 8 pull requests with zero changes in the GitHub API and on its web interface. We removed these pull requests.

[21] `https://api.github.com/repos/atom/atom/pulls/16491`

[22] `https://github.com/atom/atom/pull/16491/files`

7 Conclusions

In this paper we analyzed the contribution behavior of internal and external developers of five well-known company-owned open-source projects: `atom`, `electron`, `git-lfs`, `linguist`, and `hubot` projects. We found that these projects are very receptive for external developers: many externals play important role in the studied projects, such as reviewing and integrating pull requests. Considering all the projects, internal developers are responsible for 43.3% of the pull requests performed (external developers placed 56.7%). Analyzing just `hubot` project, we observed that only 18% of the pull requests had been placed by internal developers. However, the absolute number of external members is many more times greater than internals ones. As a consequence, many externals are casual contributors (i.e., developers that only contributed once (although we also identified internals that are also casual contributors).

These differences indicate that it is necessary to analyze each project individually to better understand this phenomenon, since there can be different factors influencing the behavior, like the priority the company is giving to the project; the project attractiveness; and vendors who make use of the project. We also noticed that, contributions from external developers are shorter than those sent by internal ones, and that external developers contribute more documentation related pull request, although we also found complex code pull request.

7.1 Future Work

This study can be a fruitful research area which can benefit companies willing to open-source their code and developers who are afraid of contributing to recently open-sourced projects. For future work, we plan to expand the scope of this study by investigating additional OSS projects. In addition, we plan to conduct surveys and interviews with developers in order to cross-validate the findings from the repositories.

Abbreviations

OSS: Open Source Software; RQ: Research Question; MWW: Mann-Whitney-Wilcoxon;

Declarations

Availability of data and materials

All data used in this paper can be found online at `https://github.com/fronchetti/JBCS-2018`.

Competing interests

The authors declare that they have no competing interests.

Funding

This work is supported by the CNPq (Grants #406308/2016-0 and #430642/2016-4); PROPESP/UFPA; and FAPESP (Grant #2015/24527-3).

Author's contributions

LFD carried out the experiments and drafted the manuscript. IS conceived of the study and participated in the design of the study and performed the statistical analysis. GP participated in its design and helped to draft the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We thank the reviewers for their valuable comments.

Author details

[1]University of São Paulo, Institute of Mathematics and Statistics, São Paulo, Brazil. [2]Federal University of Technology, Paraná, Department of Computing, Campo Mourão, Brazil. [3]Northern Arizona University, Flagstaff, AZ, USA. [4]Federal University of Pará, Faculty of Computing, Belém, Brazil.

References

1. Avelino, G., Passos, L.T., Hora, A.C., Valente, M.T.: A novel approach for estimating truck factors. In: 24th IEEE International Conference on Program Comprehension, ICPC 2016, Austin, TX, USA, May 16-17, 2016, pp. 1–10 (2016)
2. Coelho, J., Valente, M.T.: Why modern open source projects fail. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017, pp. 186–196 (2017)
3. Riehle, D., Riemer, P., Kolassa, C., Schmidt, M.: Paid vs. volunteer work in open source. In: HICSS ' 14, pp. 3286–3295 (2014). doi:10.1109/HICSS.2014.407
4. Pinto, G., Steinmacher, I., Gerosa, M.A.: Leaving behind the software history when transitioning to open source: Reasons and implications. In: Open Source Systems: Enterprise Software and Solutions - 14th IFIP WG 2.13 International Conference, OSS 2018, Athens, Greece, June 8-10, 2018, Proceedings, pp. 50–60 (2018)
5. Dias, L.F., Santos, J., Steinmacher, I., Pinto, G.: Who drives company-owned oss projects: Employees or volunteers? In: V Workshop on Software Visualization, Evolution and Maintenance. VEM, p. 10 (2017)
6. Wilks, D.S.: Statistical Methods in the Atmospheric Sciences. Academic Press, ??? (2011). https://books.google.com.br/books?id=IJuCVtQ0ySIC
7. Romano, J., Kromrey, J., Coraggio, J., Skowronek, J.: Should we really be using t-test and cohen's d for evaluating group differences on the nsse and other surveys? In: Annual Meeting of the Florida Association of Institutional Research (2006)
8. Steinmacher, I., Wiese, I.S., Conte, T., Gerosa, M.A., Redmiles, D.: The hard life of open source software project newcomers. In: Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering. CHASE '14, pp. 72–78. ACM, ??? (2014)
9. Grissom, R.J., Kim, J.J.: Effect Sizes for Research: Univariate and Multivariate Applications. Taylor & Francis, ??? (2005)
10. Gousios, G., Zaidman, A., Storey, M.D., van Deursen, A.: Work practices and challenges in pull-based development: The integrator's perspective. In: 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1, pp. 358–368 (2015)
11. Guo, P.J., Zimmermann, T., Nagappan, N., Murphy, B.: Characterizing and predicting which bugs get fixed: an empirical study of microsoft windows. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010, pp. 495–504 (2010)
12. Potvin, R., Levenberg, J.: Why google stores billions of lines of code in a single repository. Commun. ACM **59**(7), 78–87 (2016)
13. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11**(3), 309–346 (2002)
14. Pinto, G., Steinmacher, I., Gerosa, M.A.: 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, 2016 - Volume 1, pp. 112–123 (2016)
15. Lee, A., Carver, J.C., Bosu, A.: Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey. In: Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017, pp. 187–197 (2017)
16. Rebouças, M., Santos, R.O., Pinto, G., Castor, F.: How does contributors' involvement influence the build status of an open-source software project? In: Proceedings of the 14th International Conference on Mining Software Repositories. MSR '17, pp. 475–478 (2017)
17. Gousios, G., Pinzger, M., van Deursen, A.: An exploratory study of the pull-based software development model. In: 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014, pp. 345–355 (2014)
18. Dias, L.F., Steinmacher, I., Pinto, G., da Costa, D.A., Gerosa, M.A.: How does the shift to github impact project collaboration? In: 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, Raleigh, NC, USA, October 2-7, 2016, pp. 473–477 (2016)
19. Steinmacher, I., Pinto, G., Wiese, I.S., Gerosa, M.A.: Almost there: a study on quasi-contributors in open source software projects. In: Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018, pp. 256–266 (2018)
20. Zhou, M., Mockus, A., Ma, X., Zhang, L., Mei, H.: Inflow and retention in oss communities with commercial involvement: A case study of three hybrid projects. ACM TOSEM **25**(2), 13 (2016)
21. Homscheid, D., Schaarschmidt, M.: Between organization and community: investigating turnover intention factors of firm-sponsored open source software developers. In: WebSci '16, pp. 336–337 (2016). ACM
22. Atiq, A., Tripathi, A.: Impact of financial benefits on open source software sustainability. In: 37^{th} ICIS (2016)
23. Crowston, K.: Open source technology development. In: Handbook of Science and Technology Convergence, pp. 475–486 (2016)
24. Pinto, G., Steinmacher, I., Dias, L.F., Gerosa, M.: On the challenges of open-sourcing proprietary software projects. Empirical Software Engineering (2018). doi:10.1007/s10664-018-9609-6
25. Pham, R., Singer, L., Liskin, O., Figueira Filho, F., Schneider, K.: Creating a shared understanding of testing culture on a social coding site. In: Proceedings of the 2013 International Conference on Software Engineering. ICSE '13, pp. 112–121 (2013)
26. Pham, R., Singer, L., Schneider, K.: Building test suites in social coding sites by leveraging drive-by commits. In: 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013, pp. 1209–1212 (2013)
27. Vasilescu, B., Filkov, V., Serebrenik, A.: Perceptions of diversity on git hub: A user survey. In: 8th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015, Florence, Italy, May 18, 2015, pp. 50–56 (2015)
28. Lee, A., Carver, J.C.: Are one-time contributors different? a comparison to core and periphery developers in

floss repositories. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–10 (2017). doi:10.1109/ESEM.2017.7

29. Barcomb, A.: Episodic volunteering in open source communities. In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering. EASE '16, pp. 3–133. ACM, New York, NY, USA (2016). doi:10.1145/2915970.2915972. http://doi.acm.org/10.1145/2915970.2915972