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
[1]University of São Paulo, Institute of Mathematics and Statistics, São Paulo, 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, 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 settings. 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 (i.e., pull-requests) made 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 external community, the employees still play the most part of 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 the needed tasks [1]. Due to this grin scenario, it is not uncommon 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 the more systematic approaches should be employed to shed additional light on this important direction. 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 needed.
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 payed 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 OSS as part of their daily job (*e.g.,* the company that they work for maintain an OSS project). Throughout this paper, we refer to them as "internal developers". Developers that do not work for the company that maintains the OSS project are refereed as "external developers".

In this paper we extend a previous analysis [4], 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 of 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 internal or external one (more details at Section 2). Through a set of quantitative and qualitative analysis, this paper makes the following contributions:

- Extending the evidence related to the contributions performed by internal and external developers on company-owned OSS projects;
- Understanding the receptivity of external developers on company-owned OSS projects;

[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`

- Shedding the light that the contribution behavior is project-dependent, and it is necessary to study the projects individually to better understand the phenomenon.

2 Method

In this section we report our research question (Section 2.1), the studied projects (Section 2.2), and the research approach (Section 2.3).

2.1 Research Question

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

RQ1. Are OSS contributions mostly made by internal developers?

Why? This exploratory research question draws the landscape of the actual stage of financing OSS projects. It also provide evidence to understand the role that the external developers play in this kind of endeavor.

RQ1.1. Who faces a harder time to get the contributions accepted?

Why? In this sub-research question, we focus our interest on understanding the rate of success and rejection that internals and externals have. As the literature suggest, it it not easy to contribute to open-source projects [5]. We therefore explore whether external developers are facing a harder time, when compared to their internal peers.

RQ1.2. Are internals the top contributors of company-owned OSS projects?

Why? In this research 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.

RQ2. Who is in charge of processing pull-requests?

Why? This research question explores the degree of involvement of externals and internals in company-owned OSS projects. Since processing pull-requests is a notorious activity that only experienced contributors are willing to perform [6], 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 succeed in decreasing the barriers for external developers joined the project.

RQ3. Are pull-requests submitted from internals processed faster than the ones submitted from externals?

Why? In this research question we are intended to understand whether pull-requests made by internal developers have higher priority than external developers, and, as consequence, are processed faster. If that is the case, answers to this question might help improve how company-owned OSS projects treat external developers.

RQ4. Are externals more participative in the pull-request review cycle?

Why? Commits or pull-requests are not the only way to measure participation. In fact, commits and pull-requests can be seen as the end of the road: the contributor successfully placed a contribution. However, before contributing, potential contributors might provide comments to pull-requests under review. On Github, anyone can freely provide comments to a pull-request, regardless if the GitHub user has contributed before to the project. Here we investigate whether external members are more participative, in terms of commenting in pull-requests, than internal members.

RQ5. What are the characteristics of the contributions made by external developers?

Why? Finally, we qualitatively inspected the source code changes to understand what are the kind of contributions performed. 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 contributions made by internal developers. We complemented this analysis with an investigation over the number of changes and commits in a pull-request.

2.2 Studied projects

We provide an in-depth investigation on 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 ∼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, ∼5,400 stars, and ∼2,000 forks. It is mostly written in Ruby, and has ∼6 years of historical records.

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

Since these projects are developed by (and maintained at) GitHub, we reduce false positives by taking advantage of GitHub features used to identify developers roles. For instance, 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"*[4]. Therefore, for each pull-request investigated, we

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

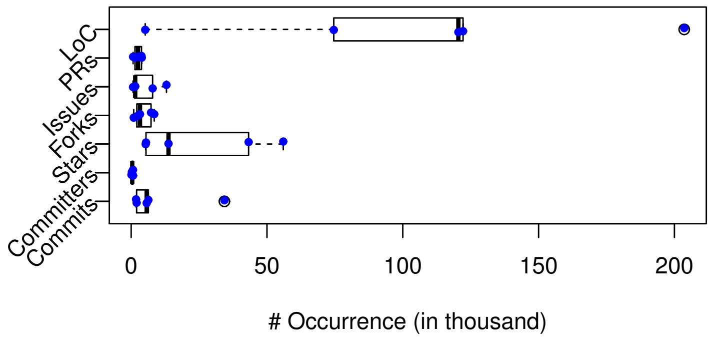

Figure 1 Characteristics of the analyzed projects

verified whether the author has the `site_admin` flag enabled. If so, we marked she as *internal*; *external* otherwise.

2.3 Approach

We followed a mix-methods approach, combining quantitative and qualitative research method. Our research can be summarized in three steps.

2.3.1 Step 1: Quantitative Analysis

In step 1, we start 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 number of commits per pull-request;
- the number of changes per pull-request;
- the number of comments during the code reviews per pull-request;
- the time taken to process a pull-request.

The data reported in this paper are based on pull-requests that were performed from the very beginning of the studied projects, up to January, 2018 — when we collected data. All data used in this study is available online at: `https://github.com/fronchetti/JBCS-2018`.

2.3.2 Step 2: Qualitative Analysis

Still, in order to uncover the reasons for acceptance, we selected a representative sample (confidence level of 95% with a ±5% confidence interval) of 334 accepted pull-requests at `atom` for manual analysis. We also validated this analysis with another manual analysis in a random sample of 150 pull-requests accepted at `hubot`. The results are present throughout Section 3.5.

Additionally, 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.

2.3.3 Step 3: Statistical Analysis
For statistics, we used Mann-Whitney-Wilcoxon (MWW) tests [7] and Cliff's delta effect-size measures [8]. We used MWW tests to verify if two distributions come from the same population ($\alpha = 0.05$). 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*) [9].

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 pull-requests submitted, as it can be observed in Table 4. On one hand, external 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 texttlinguist. For `hubot` and texttlinguist, external developers are responsible for more than 75% of the pull-requests in the project. If we consider all the projects, we come to 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 are 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 overview, Figure 2 depicts the evolution of pull-requests, grouped by their states (open, closed, and merged) at collection time (Jan 2018). Finally, all of our obtained statistical results (p-values and effect-size values) can be found 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 Statistical results. Green cells indicate large effect size, whereas yellow cells indicate medium effect size. Projects `hubot`, `electron`, and `git-lfs` have NA as their p-values and effect size since they have too small sample size.

Projects	**Open PRs**		**Closed PRs**		**Merged PRs**	
	p-value	*delta*	*p*-value	*delta*	*p*-value	*delta*
`atom`	0.003	–0.665	<0.001	–0.645	<0.001	+ 0.534
`electron`	NA	NA	<0.001	–0.740	0.001	–0.370
`git-lfs`	NA	NA	0.06	–0.270	0.003	+0.387
`hubot`	NA	NA	<0.001	–0.694	0.002	–0.339
`linguist`	0.366	–0.428	<0.001	–0.855	0.001	–0.504

From the figures, it is first possible to see that the maintainers do a great job on processing pull-requests, given the small amount 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.

3.1.1 RQ1.1. Who faces a harder time to get the contributions accepted?

When studying the merged pull-requests (the accepted ones), 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 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 pull-requests is done by external developers for 4 out of 5 projects (p-value ≤ 0.001), with a 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.

3.1.2 RQ1.2. Are internals the top contributors of company-owned OSS projects?

By analyzing the top-10 contributor 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 repository (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)

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

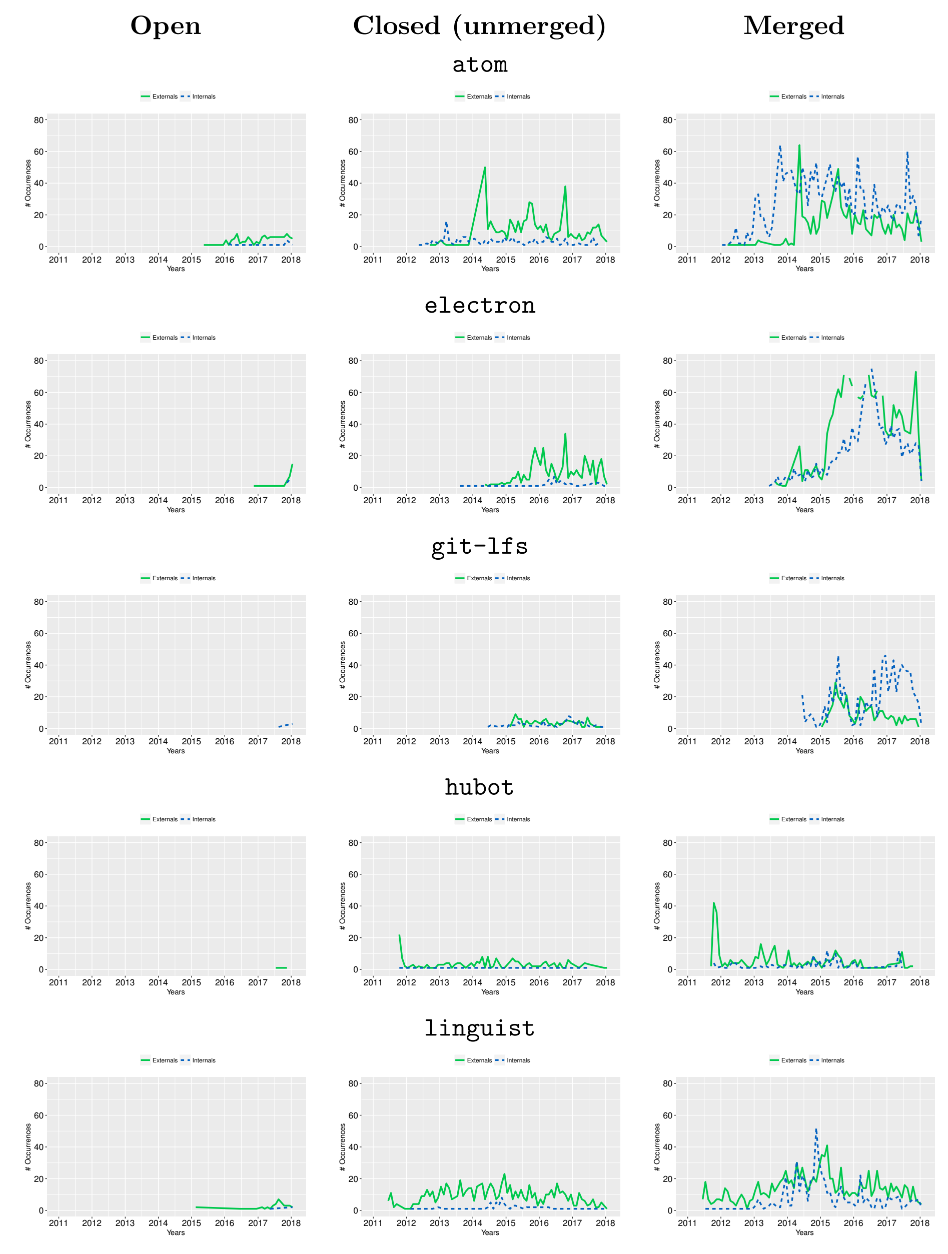

Figure 2 Time series for open, closed/merged, and closed/unmerged Pull-requests submitted by internals and externals.

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

easy to notice, the external contributors population is mostly composed of casual contributors [10, 11]. 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 with the study of Pinto and colleagues [10], 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 internals are casual).

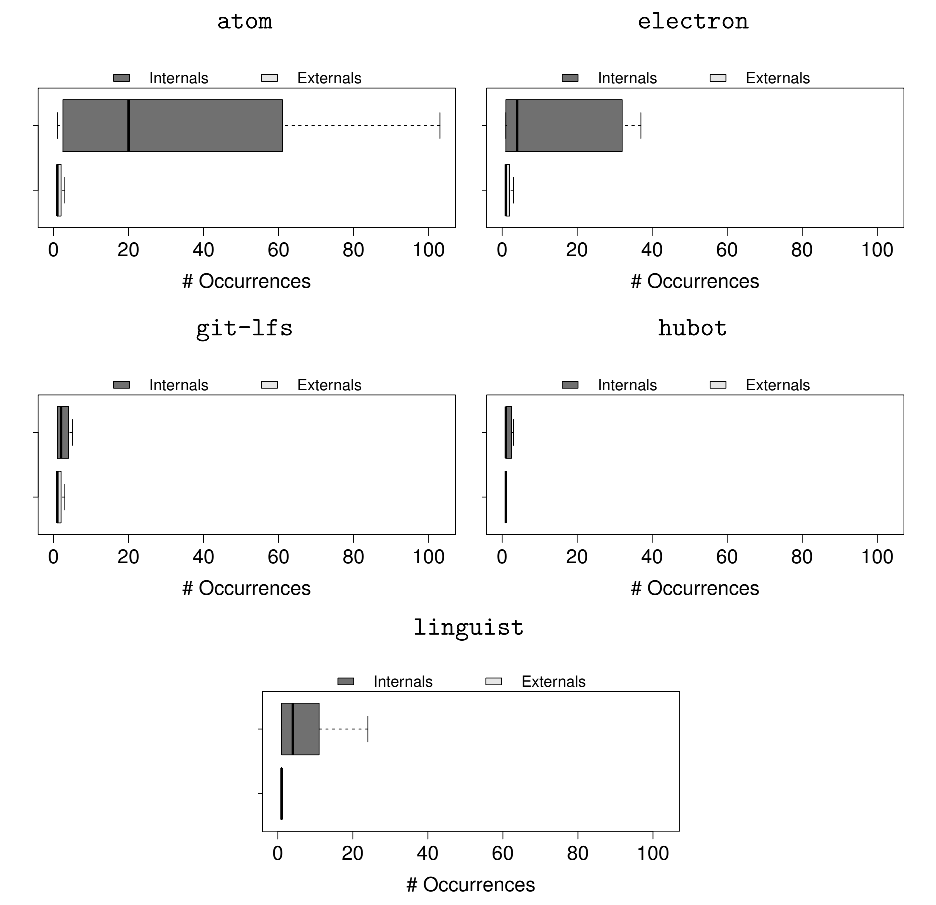

Figure 3 Distribution of number of pull-requests made by internal and external developers (we dismissed outliers to ease visualization).

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%

3.2 RQ2. Who is in charge of processing pull-requests?

To answer this RQ, we studied whether the integrator (the role that the developer that integrates a pull-request play) is performed by an internal or by an external member. Figure 4 shows the percentage of pull-request processed by internals and externals members.

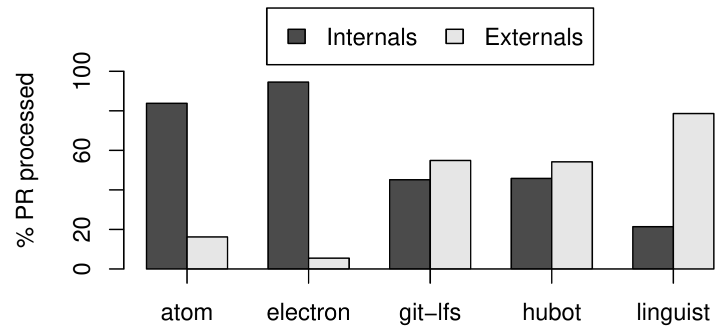

Figure 4 Percentage of pull-requests integrated by external and internal developers

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 5 shows a different perspective: the percentage of unique integrators that are internal or external developers.

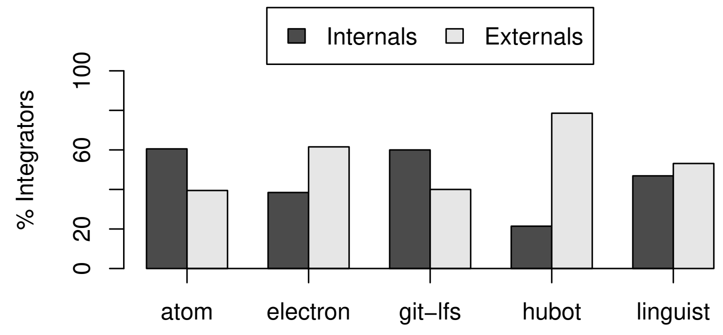

Figure 5 Percentage of unique integrator per project

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 project `hubot`, in which 11 (78%) of the integrators are external developers (which corroborates with the findings of Section 3.1.2, 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.

3.3 RQ3. Are pull-requests submitted from internals processed faster than the ones submitted from externals?

In this research question, we studied the difference, in terms 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 6 shows the average number of days for each studied project.

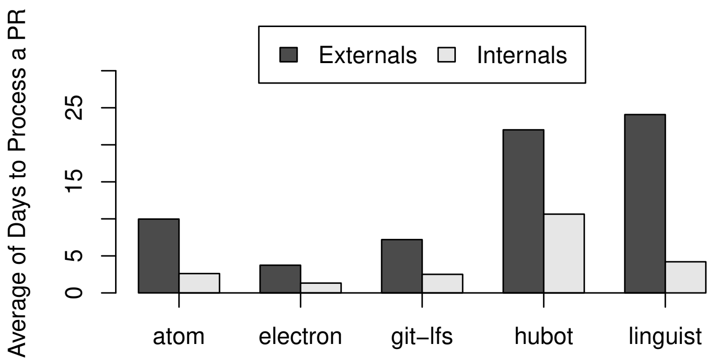

Figure 6 Average of days needed to process a pull-request

As we can see in the figure, for all studied projects, on average, pull-requests submitted from internals are process faster than the ones submitted from 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 is aimed to improve the documentation (it adds 32 lines to 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 over and merged the patch. 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: "*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.

3.4 RQ4. Are externals more participative in the pull-request review cycle?

We also investigated how internal and external contributors differ in terms of the **number of comments** received during code review of a pull-request. Figure 7 shows the distribution of this metric.

As we can see, both groups receive comments in 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 7: 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, show 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 🟨

3.5 RQ5. 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.

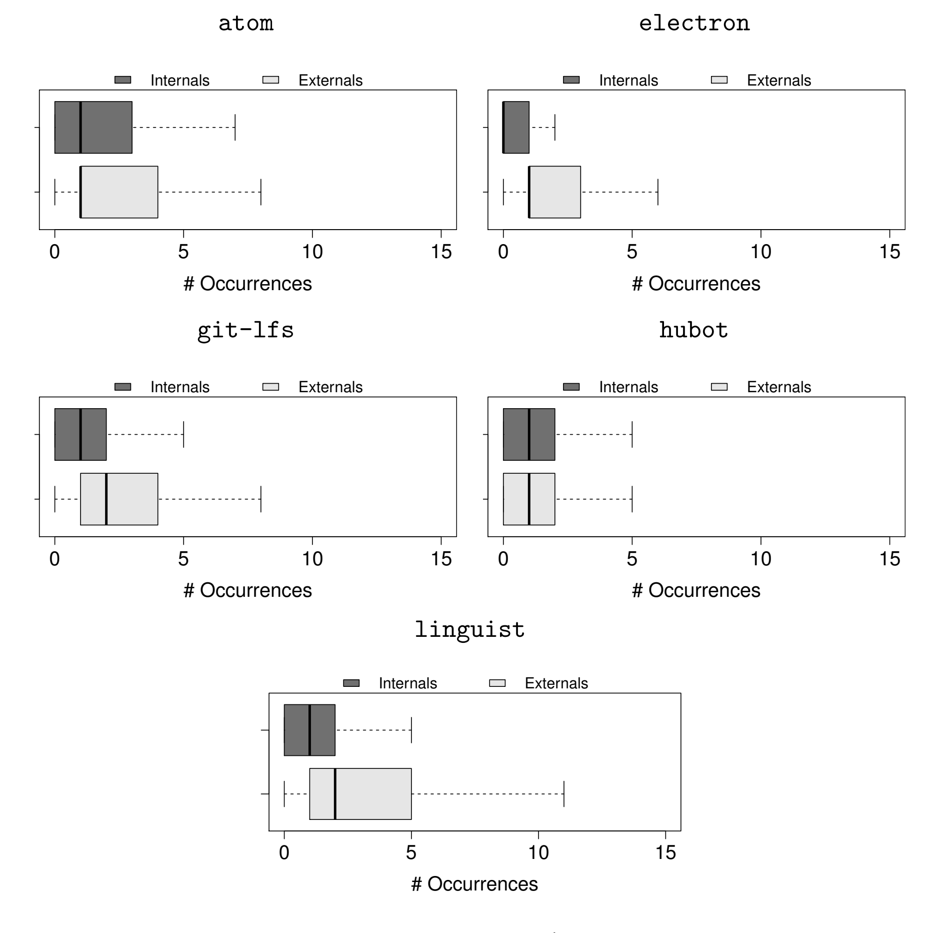

Figure 7 Number of comments received per pull-request (we dismissed outliers to ease visualization)

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[5], not enough information[6], and code comments[7]. 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[8], although complex code changes exist[9]. 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).

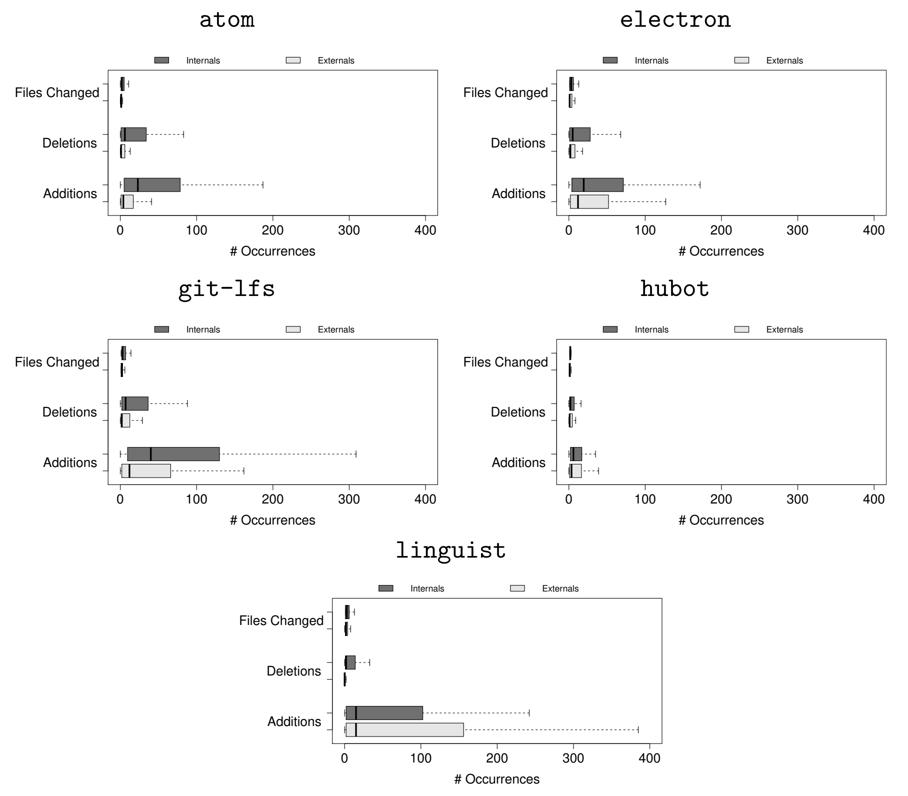

Figure 8 Files changed, lines removed, and lines added per pull-request (we dismissed outliers to ease visualization)

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;

[5]`https://github.com/atom/atom/pull/1929`
[6]`https://github.com/atom/atom/pull/2602`
[7]`https://github.com/atom/atom/pull/8452`
[8]`https://github.com/hubotio/hubot/pull/788`
[9]`https://github.com/hubotio/hubot/pull/489`

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, the internal developers indeed include more files that external ones in all analyzed projects. For number of deleted lines, 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 [6] and can also reduce the chance of breaking the continuous integration build [12].

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

By observing Figure 9, we also notice that external developers' pull-requests are also smaller in terms of number of commits. Single-commit pull-requests are rather common, accounting for more that 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 employees, we can observe a higher number of commit per pull-request—which can be noticed 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). This finding suggests that both groups follow well-known guidelines for contributing to OSS (small commits and few commits per pull-request [13, 6]).

4 Discussion

Our results showed that the external community, not only make use of the studied projects, but also support the companies maintaining the project by contributing. In particular, we found cases were external members play crucial roles inside the projects, such as reviewing and integrating pull-requests. This can only be possible because the studied projects welcome externals members (which is not always the case of open-source software [14]). We further support this claim by analyzing welcoming-community features [10] 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, a license file. They also 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

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

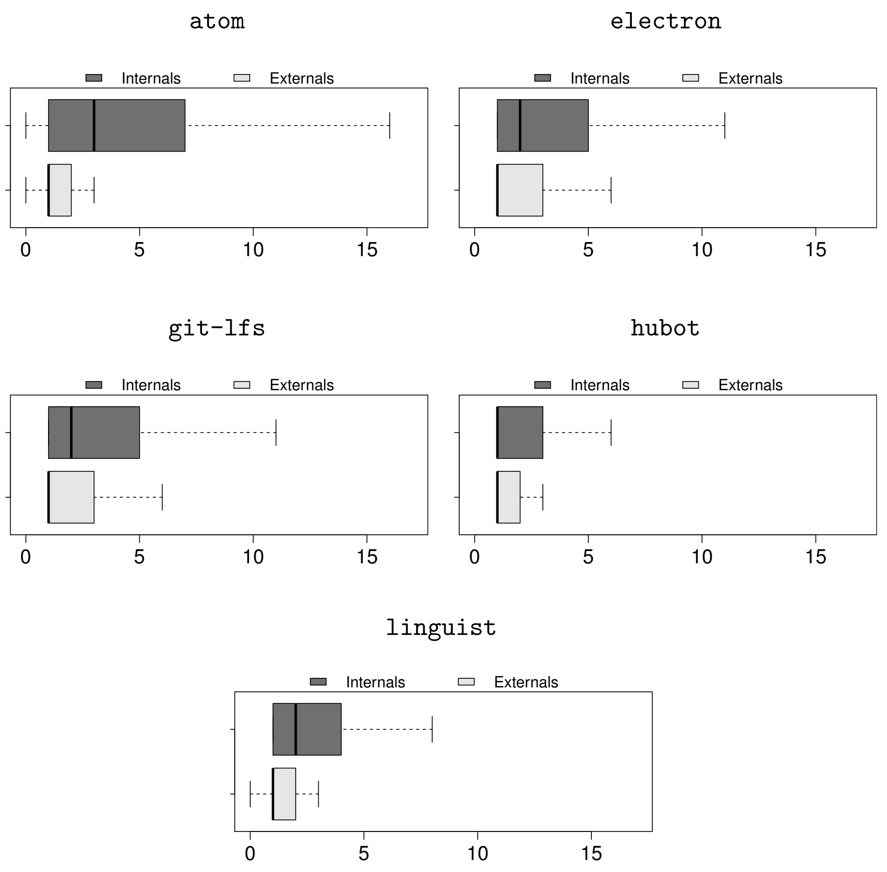

Figure 9 Number of commits per pull-request (we dismissed outliers to ease visualization)

number of unmerged pull-requests from external developers (Figure 2), external developers have to verify the project's direction and follow its guidelines to submit a pull-request, otherwise their contribution may not be accepted.

Although we found external developers supporting the studied projects, few of them have a long-term contribution history (all of them are outliers). As one can observe in Figure 3, the huge 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, 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 the company keeps contributing to it).

Moreover, although integrators are usually employee, we also found externals that play this role, which indicates a high involvement from the external community in company-owned OSS 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.

From previous studies on casual contributors [10] and quasi-contributors [15], 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 of 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 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, specially 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, 16]. Zhou *et al.* [16] analyzed how commercial involvement in OSS communities influenced the onboarding of new developers. By studying OpenStack, Docker, and Android, they found that the way the commercial involvement takes place is associated with developers participation. Homscheid and Schaarschmidt [17] 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 [18] explored how the developers perceive the differences of rewards in OSS projects, by analyzing their opinion on how project's financial resources influences 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 [19], 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 inline 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.

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 [13, 20, 21, 10, 22, 23, 24]. It is found that this kind of behavior can be beneficial for both projects and developers [10]. 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 [10], 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 wrote 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.

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. To minimize this, 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 at their contributions (if it is a corporative 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[11], even though the original pull-request on the web interface does have additions and deletions[12]. 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.

Finally, as we analyzed just five projects from the same company, we understand that the results cannot be generalized to other OSS projects being developed in other software companies. However, this study helped 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-owner open-source projects become relevant.

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

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

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

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 internals that are also casual contributors were found).

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, contribution 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 codes, 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 VEM reviewers and attendees for their valuable comments in the initial draft of this paper.

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: ICPC 2016, pp. 1–10 (2016)
2. Coelho, J., Valente, M.T.: Why modern open source projects fail. In: FSE 2017, pp. 1–11 (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. 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). http://gustavopinto.org/lost+found/vem2017.pdf
5. 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)
6. Gousios, G., Zaidman, A., Storey, M.D., van Deursen, A.: Work practices and challenges in pull-based development: The integrator's perspective. In: ICSE, pp. 358–368 (2015)

7. Wilks, D.S.: Statistical Methods in the Atmospheric Sciences. Academic Press, ??? (2011). https://books.google.com.br/books?id=IJuCVtQ0ySIC
8. Grissom, R.J., Kim, J.J.: Effect Sizes for Research: Univariate and Multivariate Applications. Taylor & Francis, ??? (2005)
9. 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)
10. Pinto, G., Steinmacher, I., Gerosa, M.: More common than you think: An in-depth study of casual contributors. In: SANER '16, pp. 112–123 (2016)
11. 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)
12. 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)
13. Gousios, G., Pinzger, M., Deursen, A.v.: An exploratory study of the pull-based software development model. In: ICSE '14, pp. 345–355 (2014)
14. Dias, L., Steinmacher, I., Pinto, G., Costa, D., Gerosa, M.: How does the shift to github impact project collaboration? In: 32^{nd} ICSME, pp. 473–477 (2016)
15. Steinmacher, I., Pinto, G., Wiese, I., Gerosa, M.A.: Almost there: A study on quasi-contributors in open-source software projects. In: ICSE'18 (2018)
16. 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)
17. 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
18. Atiq, A., Tripathi, A.: Impact of financial benefits on open source software sustainability. In: 37^{th} ICIS (2016)
19. Crowston, K.: Open source technology development. In: Handbook of Science and Technology Convergence, pp. 475–486 (2016)
20. Pham, R., Singer, L., Liskin, O., Figueira Filho, F., Schneider, K.: Creating a shared understanding of testing culture on a social coding site. In: ICSE '13, pp. 112–121 (2013)
21. Pham, R., Singer, L., Schneider, K.: Building test suites in social coding sites by leveraging drive-by commits. In: ICSE '13, pp. 1209–1212 (2013)
22. Vasilescu, B., Filkov, V., Serebrenik, A.: Perceptions of diversity on github: A user survey. In: CHASE 2015 (2015)
23. 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
24. 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

Figures

Figure 10 Characteristics of the analyzed projects.

Figure 11 Time series for open, closed/merged, and closed/unmerged pull-requests submitted by internals and externals

Figure 12 Distribution of number of pull-requests made by internal and external developers (we dismissed outliers to ease visualization)

Figure 13 Percentage of pull-requests integrated by external and internal members

Figure 14 Percentage of unique integrators per project

Figure 15 **Average of days needed to process a pull-request**

Figure 16 **Number of comments received per pull-request (we dismissed outliers to ease visualization)**

Figure 17 **Files changes, lines removed, and lines added per pull-request (we dismissed outliers to ease visualization)**

Figure 18 **Number of commits per pull-request (we dismissed outliers to ease visualization)**