How Does Contributors' Involvement Influence the Build Status of an Open-Source Software Project?

Marcel Rebouças, Renato O. Santos
Federal University of Pernambuco
Recife, Brazil
{mscr, ros3}@cin.ufpe.br

Gustavo Pinto
Federal Institute of Pará
Belém, Brazil
gustavo.pinto@ifpa.edu.br

Fernando Castor
Federal University of Pernambuco
Recife, Brazil
castor@cin.ufpe.br

***Abstract*—The recent introduction of the pull-based development model promoted agile development practices such as Code Reviews and Continuous Integration (CI). CI, in particular, is currently a standard development practice in open-source software (OSS) projects. Although it is well-known that OSS contributors have different involvements (*e.g.,* while some developers drive the project, there is a long tail of peripheral developers), little is known about how the contributor's degree of participation can influence the build status of an OSS project. Through TravisTorrent's dataset, we compare the success rates of builds made by casual and non-casual contributors and what factors on their contributions may influence the build result. Our results suggest that there is no representative difference between their build success (they are similar in 85% of the analyzed projects), meaning that being a casual contributor is not a strong indicator for creating failing builds. Also, factors like the size of their contributions and the number of project configurations (jobs) have the potential of impacting the build success.**

I. Introduction

The introduction of pull-based development practices, along with tools such as version control systems and integrated issue trackers, created environments that facilitates the process of creating, maintaining, and contributing to open-source software [4]. It also allowed the usage of practices such as Continuous Integration, which is seen as a key practice to deliver high quality working software [5].

Moreover, it is well-known that open-source software development is driven by a diverse community of developers, with a variety of skills and interests [11]. It is also part of the process of open-source software to recruit, onboard, and retain new contributors [9]. Thus, CI systems are particularly relevant to make sure that contributions from a not-well-known developer are properly tested and integrated in a timely manner.

In a recent study, Pinto *et al.* [7] observed several developers do not want to have a key role on the project, but want to contribute nevertheless. These not-that-involved developers were called "casual contributors". More interestingly, however, is the fact they are popular: 48.98% of the contributors of the most popular open-source projects hosted on Github were, in fact, casual contributors. Still, different than what was thought [6], casual contributions are far from trivial: 30.20% fixed bugs, 18.75% added features, and 8.85% refactored code. It is also important to note that being a casual is not the same as being inexperieced or a novice.

However, regardless the interest that one might have in a project, it is not always straightforward to make the very first contribution [9]. Project newcomers may face challenges in (1) finding a suitable task to start, (2) coding issues, or (3) dealing with documentation problems [7, 9]. As a result, they might ended up placing contributions that do not fully adhere with the project's guidelines or, at the worst scenario, break the build.

One might believe that casuals are more prone to break the build, since they might have little or no prior knowledge on the project domain. However, since the build status is often taken into account when maintainers are reviewing new contributions [12], they might put more effort to create their first pull-requests. To shed light on these questions, we analyze if the contributor's involvement can influence the build status in an open-source project that uses CI. More concretely, we raise the following research question:

> **RQ.** Are casual contributors more prone to create a failing build?

To answer this research question, we used TravisTorrent[1], an easy-to-use backup of Travis CI, which is by far the most used continuous integration tool [5].Our key finding is: the rate of build success of casual contributors is similar to those of non-casuals in 85% of the analyzed projects, meaning that being a casual contributor is not a strong indicator for creating failing builds. However, there seems to be factors, like the number of jobs, that increases their chance of build failure.

II. Methodology

Data collection. We used the data released by TravisTorrent on 06-Dec-2016 as our main data source. TravisTorrent provides Travis CI build analyses, annotated with GitHub[2] meta-data, and tests-related data [1], such as the duration of a build, the number of jobs, the number of commits and the amount of tests ran in the said build. TravisTorrent, however, does not include committer-related data (*i.e.,* name and email), which are necessary to detect the contributor's involvement with the project. For this reason, we used the official Travis API to enrich the original dataset with the committers' name

[1]https://travistorrent.testroots.org
[2]https://github.com

and email address[3]. Since one potential contributor can use several different email addresses, we used a disambiguation technique [2] to mitigate the threat of counting the same contributor twice for the same project. We substituted 7,285 occurrences of duplicated users and ended up with 35,360 users across all projects, considering users as unique per project (a contributor can be a casual contributor in one project and a core participant of another one).

Data cleaning. Before handling the data, we removed projects that:

- Do not have a long history of Travis usage. We removed 189 projects that did not have at least 5 casual and 5 non-casual builds.
- Do not use Travis effectively. We removed 19 projects that did not have a single successful build made by casuals and non-casuals.
- The disambiguation technique was not accurate. For instance, `cloudfoundry/cloud_controller_ng` project joined the name of each contributor of a contribution, making it not suitable for disambiguation (*e.g.,* Name: Mark and Anna, Email: mark+anna@host.com).

We ended up with a curated list of 1074 projects (755 using Ruby, 315 using Java and 4 using JavaScript).

Data preprocessing. At Travis, a single push or pull-request can trigger a build with multiple jobs. A job corresponds to a configuration of the building step (*i.e., the SDK version or the DBMS*). While there are projects that just use one job per build (*i.e.* 249 projects had a mean of 1 job per build), we found that there are 39 projects that use, on average, more than 20 build configurations, which triggers more than 20 jobs per push or pull-request[4]. Since jobs share the same build-id, we joined them to correctly consider the number of builds triggered per contributor in a project, totaling 619,370 builds.

We classify the contributors' involvement based on the number of builds triggered. We grouped contributors in two sets: casual contributors and non-casuals. While Pinto *et al.* [7] considered a casual as a contributor that just made at most one contribution, it would not be fair to apply the same reasoning in our dataset. For instance, when using Travis, builds are triggered when a pull-request is sent for approval. If changes are suggested, the contributor should incorporate the changes in a new commit (triggering a new build at Travis). Thus, a single pull-request might trigger multiple builds. We chose 4 as the threshold. Close values (*e.g.* 3 or 5), however, affects less than 2% of the commits. Table I presents descriptive statistics about the distribution of contributors.

Data interpretation. We used Fisher's Exact Test [3] to compare the proportions of successful and failing builds between casuals and non-casuals. This nonparametric test is useful for comparing the proportions of two nominal variables. We also further analyze the projects that contained a significant statistical difference in the chance of build success with multiple statistical tests over other factors such as size of contributions and number of jobs. From these projects, we tried to identify the causes that led to the higher or lower chance of success of casuals' builds, when compared to the ones of frequent contributors. When appropriated, we used the non-parametric Mann-Whitney-Wilcoxon (MWW) test [13].

[3]http://gustavopinto.org/casual-contributors-msr-challenge/

[4]https://travis-ci.org/slim-template/slim/builds/179405105

TABLE I
DESCRIPTIVE STATISTICS OF CONTRIBUTORS PER LIMIT # OF BUILDS, THE NUMBER OF CONTRIBUTORS, THE TOTAL OF BUILDS MADE BY THOSE CONTRIBUTORS, AND % OF SUCCESSFUL BUILDS.

Limit	Contributors		Sum of Builds		Success
#	#	%	#	%	%
1	18,739	56.4%	18,739	3.0%	76.5%
2	23,958	72.1%	29,176	4.7%	75.5%
3	26,035	78.3%	35,407	5.7%	75.2%
4	27,136	81.6%	39,811	6.4%	75.2%
10	29,192	87.8%	53,692	8.6%	74.8%

TABLE II
AVERAGE NUMBER OF CONTRIBUTORS PER PROJECT, WITH 1ST QUARTILE, 3RD QUARTILE AND MAXIMUM NUMBER OF OCCURRENCES.

	1st	Median	Mean	3rd	Max
Casuals	8	14	24.24	28	749
Non-Casuals	2	4	6.69	8	132

III. RESULTS

The number of casual and non-casual contributors greatly varies in the projects of our dataset, as shown in Table II. Some projects contained more than 300 casual contributors (*i.e.,* `ruby/ruby` has 749, `mitchellh/vagrant` has 444, and `leereilly/swot` has 354), while maintaining a much smaller number of frequent developers: 132, 35, and 13 respectively. Also, before the filtering process, we found that only 22 (4.15%) projects had builds made by one contributor — the owner of the project — and 31 (5.92%) did not have any casual contributors. Moreover, 23.5% of the casual contributors that just made one build (as shown in Table I) had a failing state. This suggests that these contributions were never fixed, or might have been accepted without considering their statuses.

A. RQ. Are casual contributors more prone to create a failing build?

For each project, we compared the number of successful and failing builds of casual contributors to those of non-casuals, using the Fisher's Exact Test [3]. This served to identify if the rate of success of casual builds in a certain project was statistically higher than non-casuals, if the contrary applies, or if no significant difference exists with 95% confidence.

Applying the test, we found that 913 (85%) projects had no significant difference, 107 (9.97%) had more non-casual success, and 54 (5.02%) had more casual success. From now on, we refer to these sets of projects as *draw-projects*, *non-casual-wins-projects* and *casual-wins-projects*.

The result goes against our initial thought that the barriers faced by casuals would direct reflect at builds statuses, which is not true for the majority of projects. We believe, however,

that many factors, such as the size of the contribution and the number of jobs and tests, might explain the causes of such result.

1) Size of contributions: The average size of contributions made by casual contributors are smaller than those made by non-casual contributors (Wilcoxon, two-sided, p-value = 2.2e-16), both in source code churn (the median was 36.04[5] for casual ones and 85.77 for non-casuals) and in the quantity of modified files (the median was 2.56 files for casual ones and 4.48 files for non-casuals). This difference indicates that, on average, casual contributions involve a smaller set of changes, which can help to reduce the chance of breaking a build.

In the projects `mitchellh/vagrant` and `leereilly/swot`, mentioned at the beginning of this section, the median of source code lines changed by casual contributions was rather small. For instance, *mitchellh/vagrant* had a median of 4 modified source lines per casual contribution, while in *leereilly/swot* the median of modified source code lines was 0, which suggests a high number of non-code contributions (*e.g.,* documentation updates). When taking a closer look, the majority of casual contributions were made by adding .txt files. Both projects were in the set *casual-wins-projects*.

We also compared whether the size of average casual contribution size varied on *non-casual-wins-projects* and *casual-wins-projects*. However, there is not a relevant difference neither in churn (Wilcoxon, two-sided, p-value = 0.7513) nor in modified files (Wilcoxon, two-sided, p-value = 0.7581).

2) Number of jobs: The number of jobs in a build represents the different combinations of environment choices at the building process. We found that the average number of jobs in *non-casual-wins-projects* is higher than those in *casual-wins-projects* (Wilcoxon, two-sided, p-value = 0.00133) and the projects where casuals do not lose — *draw-projects* + *casual-wins-projects* — (Wilcoxon, two-sided, p-value = 0.002536). The median of jobs per build for *non-casual-wins-projects* was 4.68, while being 2.275 and 3.08 for *casual-wins-projects* and *draw-projects*.

3) Number of tests: When analyzing the number of jobs in the projects, we noticed that the projects of *non-casual-wins-projects* also had a higher average number of tests ran than those in *casual-wins-projects* (Wilcoxon, one-sided, p-value = 0.0417) and *draw-projects* (Wilcoxon, one-sided, p-value = 0.0298). This seems fair, since more tests are needed to cover a broader set of environments. For instance, the median number of tests in projects of *non-casual-wins-projects* was 68, while being 29 for the *casual-wins-projects*.

We expected that, in projects with more tests, casuals would end up failing more since the test suite would be more rigorous. Yet, this higher rigor applies to every contributor, and casuals do not have a higher proportion of failure due to tests when compared to non-casuals (Wilcoxon test, two-sided, p-value = 0.8858) in *non-casual-wins-projects*. Interestingly, builds made by non-casuals at projects where casuals do not lose — *draw-projects* + *casual-wins-projects* — have a higher chance of failing due to tests (Wilcoxon test, two-sided, p-value = 1.877e-11). This fact seems to go along with the fact that non-casuals normally make more complex contributions.

[5]The median value is decimal because we calculated the average size per project before calculating the median across projects. We did the same for the other factors.

IV. Discussion

In this section, we provide additional discussions to the results found.

Size of contributions. It was expected that the size of casual contributions, in terms of lines of code, would be lower than those of non-casuals, since they start contributing with changes that offer a lower entry barrier [9, 7]. However, since contributions can be made to files that are not directly tied to the source code, such as documentation and adding assets, it may be possible that the contributions never even presented a risk of failing a build. Thus, the nature of how contributions are made to a specific project, which may provide a low entry barrier, can lead to a high success rate of casual contributions.

Number of jobs. The median of jobs per build for projects in *non-casual-wins-projects* was significantly higher than *casual-wins-projects* and *draw-projects*. It seems that if a project contains a broader set of configurations, it might be more challenge to contribute to it. Also, the contributor might place an additional care in investigating the different environment characteristics. Such specificities of how the project works under different combinations might then require a deeper understanding of it, resulting in casual contributions failing more. Documentation regarding the peculiarities of the project under the desired configurations would probably help casual contributors to be more aware and fail in less builds.

Number of tests. A higher number of tests in the set of projects that also contains more jobs supports the fact contributing to those projects must be more rigorous. However, although the amount of builds that fail because of tests is higher, it happens to both casuals and non-casuals at *non-casual-wins-projects*. It is interesting to see that, for the rest of the projects, non-casuals had more failing builds due to failing tests. We believe this might happen due to (1) the higher complexity of the contributions, (2) the complexity of the logic being added, or even (3) with the usage of techniques such as test-driven development (TDD). We leave this better understanding for future work.

Implications. Project maintainers can improve documentation regarding the project specifics in different environments so that casual and non-casual contributors can better understand how to properly test the contribution. Still, researchers can use our findings to understand the dynamics of casual contributions, since this phenomenon is not full understood yet. For instance, differently than what was found in recent studies [7], several casual-contributions were not code-related. Finally, since newcomers in general, and students in particular, might be afraid to contribute to open-source projects [9], CS professors can use our findings to motivate students to contribute to open-source software. In particular, software testing disciplines can

place emphasis on writing tests for different environments.

V. Related Work

Vasilescu *et al.* [10] found that 92% of their selected projects are configured to use Travis-CI, although 45% of them have no associated builds recorded in the Travis database. Hilton *et al.* [5] conducted a survey and studied repositories to understand how software developers use CI. Pinto *et al.* [8] investigated how developers perceive the benefits and limitations of CI techniques. Among the findings, the authors observed that although CI increases the confidence that the system is in a known state, they also found that CI users reported a false sense of confidence, when blindly trusting in tests. Vasilescu *et al.* [12] found that CI helped to increase the number of accepted pull-requests from core developers, and to reduce the quantity of rejected from non-core developers, without affecting code quality. To the best of our knowledge, this work is the first aimed at understanding the relationship how casual contributions influence the build status.

VI. Threats to Validity

Builds can be triggered by pull-requests that involves multiple committers. However, we are only considering the author of the build as the sole contributor, since the chance of these occurrences is small. For instance, we randomly selected 3000 pull-requests, and less than 15% of those contains multiple commits made by different contributors.

The act of accepting a pull-request triggers a build at Travis, in which the author is the maintainer who accepted it, and not the committer of the pull-request. This might create a bias that increases the average chance of build success towards frequent contributors, since accepting a successful pull-request always creates a successful build. Since the Travis data do not create a connection between the build of the pull-request and the acceptance, we were not able to remove the acceptance builds. Also, data from Travis do not comprehend the complete phase of evolution of a project. For instance, it is possible that Travis was just used for a small period of time and then replaced by another CI mechanism. Because of this fact, we may capture the whole scope of contributions in the evolution of a project. Also, since some projects may be inactive or not using Travis' result, we decided to remove projects without a single successful build.

As regarding the definition of a casual contributor, there is not a "correct" limit number of builds that defines it. To choose an appropriate value, we analyzed the distribution ranging from 1 to 10. Finally, since we just analyzed a subset of possible factors, we may not be capturing the full scope of possible reasons that are affecting the build status. We leave this for future work.

VII. Conclusion and Future Work

Continuous Integration aims at increasing the understanding of software changes. However, contributors still face challenges when trying to contribute for the very first times. In this paper we studied whether the contributor involvement (i.e., being a casual contributor) influences the status of a build. Our results suggests that, in 85% of the cases, there is no representative difference between contributions placed from casual and non-casual contributors, meaning that being a casual is not strong indicator for creating failing builds. For future work, we plan to analyze how the contributions between casuals and non-casuals differ, for instance, if the domain of the project can lead to the attraction of more casuals or if certain configurations of environments can be more problematic to a project's build success.

Acknowledgements. We would like to thank Paulo Borba and the anonymous reviewers for helping to improve this paper. This research was partially funded by CNPq (304755/2014-1 and 406308/2016-0), FACEPE (APQ-0839-1.03/14), and FACEPE PRONEX (APQ 0388-1.03/14).

References

[1] M. Beller, G. Gousios, and A. Zaidman. Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration. In *MSR*, 2017.

[2] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR*, pages 137–143, 2006.

[3] A. Fisher. Statistical methods for research workers. Cosmo study guides. Cosmo Publications, 1925.

[4] G. Gousios, M. Pinzger, and A. v. Deursen. An exploratory study of the pull-based software development model. In *ICSE*, pages 345–355, 2014.

[5] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig. Usage, costs, and benefits of continuous integration in open-source projects. In *ASE*, pages 426–437, 2016.

[6] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider. Creating a shared understanding of testing culture on a social coding site. In *ICSE*, 2013.

[7] G. Pinto, I. Steinmacher, and M. A. Gerosa. More common than you think: An in-depth study of casual contributors. In *SANER*, 2016.

[8] G. Pinto, M. Rebouças, and F. Castor. Inadequate testing, time pressure, and (over) confidence: A tale of continuous integration users. In *CHASE*, 2017.

[9] I. Steinmacher, I. S. Wiese, A. P. Chaves, and M. A. Gerosa. Why do newcomers abandon open source software projects? In *CHASE*, pages 25–32, 2013.

[10] B. Vasilescu, S. van Schuylenburg, J. Wulms, A. Serebrenik, and M. G. J. van den Brand. Continuous integration in a social-coding world: Empirical evidence from github. In *ICSM*, pages 401–405, 2014.

[11] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov. Gender and tenure diversity in github teams. In *CHI*, 2015.

[12] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov. Quality and productivity outcomes relating to continuous integration in github. In *ESEC/FSE 2015*, 2015.

[13] D. Wilks. *Statistical Methods in the Atmospheric Sciences*. Academic Press, 2011. ISBN 9780123850225.