Analyzing first contributions on GitHub: what do newcomers do

Vikram N. Subramanian
SWAG Lab
David R. Cheriton School of Computer Science
University of Waterloo
vnsubram@uwaterloo.ca

Ifraz Rehman
Nara Institute of Science and Technology
rehman.ifraz.qy4@is.naist.jp

Meiyappan Nagappan
SWAG Lab
David R. Cheriton School of Computer Science
University of Waterloo
mei.nagappan@uwaterloo.ca

Raula Gaikovina Kula
Nara Institute of Science and Technology
raula-k@is.naist.jp

ABSTRACT

The popularity of Open Source Software(OSS) is at an all-time high and for it to remain so it is vital for new developers to continually join and contribute to the OSS community. In this paper, to better understand the first time contributor, we study the characteristics of the first pull request(PR) made to an OSS project by developers. We mine GitHub for the first OSS PR of 3501 developers to study certain characteristics of PRs like size and nature of change. We find that PRs were a mixture of trivial and non-trivial changes. Studying a smaller sample, we find that 47% of all changes were bug fixes, 35% of changes added new features and 13.25% of changes changed documentation. By using the data presented, OSS moderators can organize their project's issue-tracker to attract more developers and first-timers can learn from other first-timers and understand what they should focus on.

KEYWORDS

GitHub, Open Source Software, First time contributions

1 INTRODUCTION

Open Source Software (OSS) has always been a vital part of software development. As per the yearly report published by GitHub[4], millions of repositories use an OSS project as a dependency. These OSS projects couldn't exist without continued contributions from those in the OSS community and therefore it is vital to ensure that new developers are regularly joining the ranks.

According to Riehle, attracting new contributors to open source projects is an investment on the project and a necessity for growth [9]. Understanding the behaviour of the target audience, first time contributors, plays an important role in ensuring this investment is productive. According to Fitzpatrick and Collins-Sussman, the scarcest resource an open source project has is attention and focus[7] and this makes knowing the target audience and their behavior even more important. Our motivation, therefore, is to precisely understand contributions by first time contributors to open source projects.

Using the data presented in this paper, moderators of OSS projects can begin to precisely understand which tasks are appealing to first timers and must be used to attract them. Project leaders and experienced users can also better understand which tasks they must spend their time on and which tasks can be delegated to beginners and individuals who are not heavily invested/involved in the project. This will help in planning future coding sprints, estimating required resources and time. The information can also help first timers understand what sort of task they can take up.

We define an open source 'contribution' as a pull request (PR) that has been successfully merged to the parent of a publicly available repository.

Many curated data sets of GitHub already exist for others to use but none allow us to efficiently select particular commits/users based on special criteria (such as a user's first OSS PR). Therefore, we develop our own approach to collect and curate data using the GitHub REST API[2].

We then conduct 2 different analyses on the collected data - a quantitative study of first-time contributions as described in section 3 and a qualitative study which required further classification as described in section 4 to answer the following research questions:

RQ1: What is the size of contribution in a user's first OSS PR?

RQ2: What type of contributions do first time OSS contributors make?

RQ3: What type of bug-fixes do first time OSS contributors make?

RQ4: How many first-time contributions are supported by documentation and what is the nature of documentation added?

2 DATA COLLECTION

GitHub follows a 'fork and pull' system where users have to create their own copy of a project (fork a project), make changes, commit them and create a pull request (PR) to the parent of the forked

Conference'17, July 2017, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

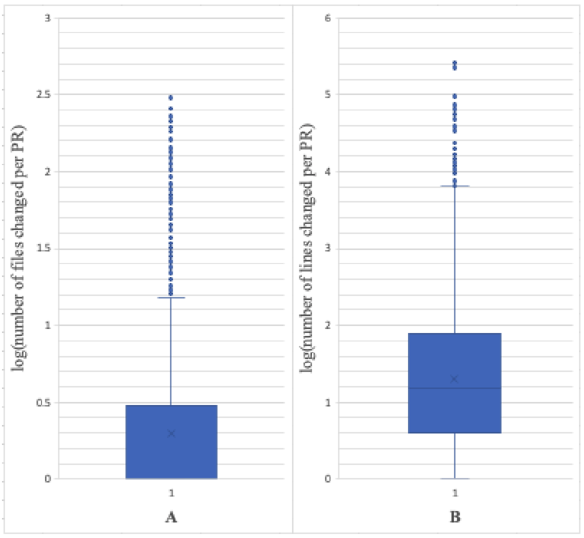

Figure 1: A- Box plot of the log of number of files changed per PR. B- Box plot of the log of number of lines changed per PR.

repository requesting to merge the commits. The moderators of the project and the general public can then review the changes made, request modifications and then the moderators can finally decide to merge the changes or not. Assuming such a frame of reference we follow the below steps to collect the data:

(1) We use the REST API[2] to obtain a list of 1000 repositories from The Apache Software Foundation's GitHub page[3] (limited by the API - there are around 1900 repositories in total). We use Apache projects as we wish to study contributors who have contributed to at least one well established project.
(2) We collect the usernames of the top 1000 most recent contributors to each of the 1000 repositories we gathered in step 1. We get 15,535 unique users.
(3) For every user, we collect a list of all their forked repositories. If the user has no forked repositories, they are dropped. Note that this list is not limited to only Apache projects and consists of any and all repositories the user has forked.
(4) We then go through each of the forked repositories and see if the user has made any commits. Any repository that the user has not made contributions to is dropped. At the end of this step, we have a list of forked repositories where the user has possibly made an OSS contribution (to the parent of the fork).
(5) Then, for each repository from step 4, we check the parent of that repository and see if commits by the user exist. If it does, we get the first commit chronologically. This commit is the user's first open-source contribution to that specific repository. This commit is possibly the user's first open-source contribution on GitHub.
(6) Once we collect commits from all the repositories as described in step 5, we sort the commits chronologically and pick the first commit and check to make sure that a PR is associated with this commit. This will be that user's first OSS contribution on GitHub.
At the end of this step, we get the first PR of 3501 users. Note that in steps 3, 4 and 5 we drop users who don't match the criteria and hence we reach 3501 users from 15,535.

Out of the 3501 PRs we studied, only 1193 were to Apache projects. This means that 34% of the developers we studied made their first contribution to Apache projects and this could possibly be their only open source contribution. Also 66% of the developers we studied have opened at least 2 open source pull requests- their first pull request in a non Apache project and another to an Apache project from where we picked this developer.

3 QUANTITATIVE ANALYSIS: WHAT IS THE SIZE OF CONTRIBUTION IN A USER'S FIRST OSS PR?

Motivation: Understanding how many lines of code and files were changed per contributions helps roughly gauge the difficulty of tasks first timers are taking up. OSS moderators can also understand which tasks to allocate for first timers.

Approach: The GitHub API returns the nature of change and number of lines changed for each file in a PR. We got 18,009 files from 3501 PRs. We process that data for our analysis.

Note: A line is classified as 'changed' by GitHub when it is modified, created or deleted. Modifications can change (add or delete) any number of characters to any line including comments and empty lines. The total number of lines changed is calculated per file by simply counting the number of lines changed. Non-textual contributions such as creating new files are classified as 0 line changes.

Results:

Fig 1B describes the number of lines changed. 31.5% of the changes were 1-5 lines and 18.8% of the changes were 6-15 lines with some extreme outliers. Alali et al.[5] conducted a very similar study on all contributions to the GNU gcc system (not limited to first time contributions) and found that 19.9% of changes changed 0-5 lines.

The number of files changed per PR follows a distribution similar to the number of lines changed per PR as shown by Fig 1A. Most changes are single file changes with a few extreme outliers that pushes the average to 5.1 changed files. Hattori and Lanza[8] studied the number of files changed by open source contributions in general and found a very similar distribution as well with 80% of all commits changing 0-5 files and less than 1% changing more than 126 files.

File changes are classified as modified, added, renamed or removed by GitHub. 54.77% of all files changed were modifications, 35.41% were added, 5.85% were renamed and 3.94% were removed.It is interesting to note that contributions with multiple file changes are usually when new files are created to initialize a new feature/aspect of that project. This accounts for a large percentage of the 35.41% of files that were 'added'.

Takeaway 1: First time contributors should not be discouraged to take up big tasks as the entire 4th quartile of PRs studied were multifile changes over 75 lines. However, with a median change of 15 lines and over half the contributions being only modifications, it is clear that most first timers take up small to medium sized changes. OSS moderators interested in selecting tasks specifically for newcomers should try to chose tasks were the number of lines changed is relative low and if possible, maybe even split up large tasks that would normally be tagged as a good first issue.

4 QUALITATIVE ANALYSIS

4.1 What type of contributions do first time OSS contributors make?

Motivation: Understanding the precise nature of the contributions that first time contributors have made is the first step in setting up schemes to encourage more people to join the OSS community. OSS moderators and contributors could identify which tasks are popular with first timers and which are not.

Approach: In this qualitative study, we want to analyze the nature of the changes made by first time contributors - the purpose of the changes. In order to do that, we classify the purpose of commits into broad categories which are presented below.

In order to produce the classifications, the first two authors independently classified 400 of 3501 commits (resulting in a confidence level (margin of error) of 95% with confidence interval of 5 [1]) into the main categories and sub categories (as described in 4.2 and 4.3). They studied the actual code changed, the commit message, the PR message, the issue/ticket (if available) and comments made in the PR . They then compared their classifications and resolved any differences. The other two authors acted as tie breakers for commits where there was no mutual agreement even after dialogue.

In the 400 commits classified, the 2 authors disagreed on the main categories for 79 commits (19.75%) and disagreed on the sub categories for 106 commits (26.5%). After discussing the disagreements, the first two authors still disagreed on 25 commits and the other two authors acted as tie breakers. Around 30% of the initial disagreements came from misunderstanding features for bugs and confusing between minor features and major features. 13% of errors came from confusing the types of documentation, particularly release notes and READMEs. Around 15% of the erros came from disagreements within the types of semantic bug fixes.

The following are the broad categories:

1) Documentation - Changes and additions made to documentation files such as READMEs and/or comments explaining code. Note: Only commits where the majority change is documentation is classified a documentation change.

2) Feature - Adding new functionality/features to the project.

3) Bug - Fixing unexpected behaviour in code.

4) Refactoring - Restructuring code to make it more understandable/readable and/or conform to coding standards

5) GIT related issues - Solving merge conflicts, adding elements to .gitignore files and other changes related to GIT.

6) Test cases - Adding test cases and/or adding code to facilitate testing.

7) Other - Anything that does not fall in the above categories.

In order to produce these broad categories, two of the authors initially came up with a classification by assigning categories to 25 commits. After doing so, the effectiveness of the categorization, what each category meant and how to identify each was discussed and changes were made to make the categorization more meaningful. After two more iterations of doing so, the categories presented were finalised and 325 more commits were classified to produce the final 400 classifications.

Results: From table 1 we see that 47% of all changes are bug fixes. Followed by addition of features at 35% and changes to documentation at 13.25%. We also see the median number number of lines changed for each type: 8 for bug fixes, 52.50 for features and 7 for documentation.

To get a more precise picture, we decided to classify the commits that added a new feature into 'minor features', 'major features' and 'Configuration and system updates'. We define them as-

2.1) Major feature - A substantial stand alone contribution or a sizeable contribution to a feature being developed.

2.2) Minor feature - A contribution that is adding functionality that is not substantial enough to be called a major feature but also not a bug fix.

2.3) Configuration and system updates - Changes to cater with changing/updated technologies and dependencies.

The results of this classification are presented in table 3 - 7% of all contributions were major features, 24.25% were minor features and 3.75% were configuration and system updates. The median number of lines changed is 36 for minor features, 698.5 for major features and 17 for configuration and system updates.

Takeaway 2: According to Steinmacher et al. [10], finding a task to start with was the second most common difficulty faced by first time contributors. Using our results, OSS moderators could reduce this problem by flagging minor feature additions that need a change of around 36 lines, minor documentation changes, select bug-fixes (as described below), and changes catering to changed dependencies as first-timer friendly.
According to Capiluppi and Michlmayr [6] "the success of a [OSS] project is often related to the number of developers it can attract.". By reserving the most popular first-time contributor tasks for first timers, projects can increase the number of contributors it has and thereby increase its popularity.

	Count	Percentage of all changes	Lines changed - Mean	Lines changed - Median	Files changed - Mean	Files changed - Median
Documentation	53	13.25%	60.30	7	2.19	1
Feature	140	35.00%	839.57	52.5	6.24	2
Bugs	188	47.00%	37.85	8	2.13	1
Refactoring	2	0.50%	2051.00	2051	19.00	19
GIT related issues	2	0.50%	3.00	3	1.50	1.5
Test Cases	13	3.25%	244.46	21	3.92	2
Other	2	0.50 %	36.00	36	69.50	69.5

Table 1: Distribution of contributions made

	Count	Percentage of all changes	Percentage of all features	Lines changed - Mean	Lines changed - Median	Files changed - Mean	Files changed - Median	Example
Minor Feature	97	24.25%	69.28%	323.96	36	3.2	2	bit.ly/2Xz4Yta
Major Feature	28	7.00%	20.00%	2655.50	698.5	18.60	8	bit.ly/2V6Sd7u
Configuration and system updates	15	3.75%	10.71%	784.20	17	2.4	1	bit.ly/3c9LZJw

Table 2: Distribution of contributions to features

	Count	Percentage of all changes	Percentage of all bug fixes	Lines changed - Mean	Lines changed - Median	Files changed - Mean	Files changed - Median	Example
Memory Bug	3	0.75%	1.59%	75.66	25	2.33	1	bit.ly/34xUX0E
Concurrency Bug	4	1.00%	2.18%	158.75	113.5	2.75	2.5	bit.ly/2VwMpDk
Semantic Bug	108	27.00%	57.45%	24.96	4	1.51	1	bit.ly/3b6KdJh
File-System Bug	9	2.25%	4.78%	31.88	9	1.66	2	bit.ly/3a8oX4e
Configuration Issues	41	10.25%	21.80%	52.50	18.5	3.70	2	bit.ly/2XyqY7t
Performance Bug	6	1.50%	3.19%	27.83	21	2.83	1.5	bit.ly/2RBr7D7
Other	17	4.25%	9.04%	44.82	28	2.00	2	bit.ly/34xUI5K

Table 3: Distribution of contributions to bug fixes

	Count	Percentage of all changes	Percentage of all bug fixes	Lines changed - Mean	Lines changed - Median	Files changed - Mean	Files changed - Median	Example
Typographic Bug	54	13.50%	28.72%	21.27	2	1.52	1	bit.ly/3ejFsxN
Exception Handling	6	1.50%	3.19%	6.83	4	1.17	1	bit.ly/2XxOn8X
Processing Error	2	0.50%	1.06%	3.50	3.5	1.00	1	bit.ly/2yhuGaH
Wrong Control Flow	8	2.00%	4.26%	50.25	16	2.75	2.5	bit.ly/3a6eB4X
Corner Cases	6	1.50%	3.19%	48.83	7.5	1.83	1.5	bit.ly/2xr11fj
Missing Cases	32	8.00%	17.02%	32.625	7.5	1.375	1	bit.ly/2V3rF70

Table 4: Distribution of contributions to semantic bug fixes

	Count	Percentage of all changes	Percentage of all contributions to documentation	Lines changed - Mean	Lines changed - Median	Files changed - Mean	Files changed - Median	Example
Release Notes	9	2.25%	16.98%	49.66	2	1.10	1	bit.ly/2ZF9p5E
Comments	9	2.25%	16.98%	21.77	8	3.88	1	bit.ly/3elESiX
Readme	33	8.25%	62.26%	76.67	12	2.10	1	bit.ly/3cdjxGI
License	2	0.50%	3.77%	11.50	11.5	1.00	1	bit.ly/2Ryn36w

Table 5: Distribution of contributions to documentation

4.2 What type of bug-fixes do first time OSS contributors make?

Motivation: Understanding the nature of bug fixes by first timers helps moderators put away a subset of bugs for first timers. It shows which tasks are popular amongst first timers and can be quickly fixed by beginners and individuals who were not involved in the project before.

Approach: We further classify the 118 bugs we got in section 4.1 into the types as described below, using the same methods as in section 4.1.

3) Bug - Fixing unexpected behaviour in code.

3.1) Memory bug - Fixing bugs related to memory allocation or garbage collection.

3.2) Concurrency Bug - Fixing bugs related to concurrent tasks and race conditions.

3.3) Semantic Bug - Fixing inconsistencies with requirements or programmer intentions

3.3.1) Typographic bug - Fixing typographic mistakes such as misspellings, mistakenly assigning values to incorrect variables and calling incorrect methods, and confusing positional and keyword arguments.

3.3.2) Exception Handling - Fixing error handling and/or added error handling for new run time errors

3.3.3) Processing error - Correcting incorrect mathematical expressions and/or equations.

3.3.4) Wrong control flow - Correcting incorrect control flow.

3.3.5) Corner cases - Accounting for previously missed corner cases.

3.3.6) Missing cases - Accounting for a previously missed case in functionality.

3.4) File-System bug - Correcting issues related to reading, writing and I/O.

3.5) Configuration issues - Correcting issues related to configuration, building, compiling, continuous integration and deploying.

3.6) Performance bug - Fixing bugs that caused degradation in performance.

3.7) Other

To produce this classification of bugs, we took the classification Valdivia Garcia [11] used to classify bugs and added/removed categories to have a more informative classification pertaining to our study using the method as described in section 4.1.

Results:

Table 4 describes the distributions of bug types. Over 57% of all bugs were semantic bugs. Table 5 describes the distribution of semantic bugs. Perhaps unsurprisingly, 28.72% of all bug fixes (and 13.5% of all changes) were fixing typographic errors with the median number of lines changed being 2. The second most popular kind of bug fixes were fixing configuration issues. The third most popular was adding missing cases.

Takeaway 3: According to Wang and Sharma [12] "new developers have difficulty finding the bugs that are of interest, that match their skill sets, are not duplicates, and are important for their future community". Our data can fill part of the puzzle. First time contributors can take up bugs that would require minimal lines changed (5-20) and/or focus on configuration issues and semantic issues such as typographic errors and adding missing cases.

4.3 How many first-time contributions are supported by documentation and what is the nature of documentation added?

Motivation: Documentation is an integral part of software development and it is important to understand the impact first timers can have on it. It is reasonable to expect many first timers to contribute to documentation and therefore understanding the nature and size of changes they make will help OSS moderators tailor tasks to first timers. Understanding how first timers document the changes they make also gives insight into how code is documented and how complex the changes made are.

Approach: We further classify the 55 commits classified as documentation in 4.1 into the types described below, using the same methods as in 4.1. We also investigate the 345 commits that were classified as changes to code and see how many are supported by documentation explaining the changes made.

1) Documentation - Changes and additions made to documentation files such as READMEs and/or comments explaining code. Note: Only commits where the majority change is documentation is classified a documentation change.

1.1)Release Notes - Updates on new features added, bugs fixed or the work done

1.2)Comments - Information added in between code to explain its functionality

1.3)Readme - Installation or usage instructions and/or information about the software being developed.

1.4)License - Adding/modifying licensing information.

Results: Table 6 describes the classification of contributions that changed only documentation. Almost 17% of these contributions were adding only comments to code and had a median of 8 lines changed. 62.26% of these contributions were edits to README files with a median of 12 lines changes.

In studying the 345 commits that were classified into categories other than documentation, we find that 16% of these commits were accompanied by edits to documentation files such as READMEs that describes the functionality added/changed. Specifically, only 25% of small features are accompanied with documentation and 42% of major features are accompanied with edits to documentation files.

Takeaway 4: It is not unreasonable to expect first-timers looking for an easy first task to enter the world of open source to want to contribute to documentation. Riehle[9] recommends OSS moderators to list a series of simple tasks that are designed specifically for newcomers as they "provide a mechanism to engage volunteers" and adding documentation changes to that list could be a good idea. As the median number of lines changed while editing READMEs was only 12, OSS moderators could also look into breaking up larger tasks into smaller ones while adding them to the list

5 LIMITATIONS AND THREATS

Some limitations of the method used to mine data are-

1.The GitHub API limits the number of data units returned per request to 1000. This means that if any of the queries made for a user have more than 1000 data units, then that user will have to be dropped. Here we define a data unit as the smallest unit of data the API returns for a request.

2.The user has complete control over which repositories they want to list under their 'owned' repositories section and therefore if the user has decided to remove their first contribution from their page, then this script will not pick up that contribution. However, we know of no reason why a developer would do so.

3.OSS projects have existed long before GitHub - our approach picks up only a user's first GitHub OSS contribution.

4. Our initial source of users was the Apache Software Foundation. This means that all the users we studied have, at some point, contributed to this one organization. This could possibly reduce the representativeness of our data.

5. It is reasonable to expect the list of repositories we collected to contain 'non-software' repositories such as home work assignments, informal toy projects etc. Munaiah et al. have developed a framework that can differentiate between those and repositories containing engineered software projects. When we ran our list of 3500 projects through that framework, we found that only 30 (0.86%) were non-software projects.

A user has to put some effort in creating a fork of a parent project, making a pull request to the parent and then have that pull request accepted to be a part of our list (and not just push upstream). We argue that this gives these contributions some 'value' and that most 'non-software' repositories would not follow this workflow.

6 CONCLUSION

In this study we examine quantitatively and qualitatively the first contributions of developers to OSS projects on GitHub. From our results, OSS moderators can understand which kind of tasks were more popular with first time contributors. While we cannot say if moderators should curate more of such tasks, we can say that such tasks are not impossible for first time contributors to do.

The scripts used to mine the data and the data itself is available at https://bit.ly/2UZTehj

REFERENCES

[1] [n.d.]. Confidence Intervals. surveysystem.com/sscalc.htm#one

[2] [n.d.]. GitHub API v3. https://developer.github.com/v3/

[3] 2019. The Apache Software Foundation. https://github.com/apache

[4] 2019. The State of the Octoverse. https://octoverse.github.com/

[5] A. Alali, H. Kagdi, and J. I. Maletic. 2008. What's a Typical Commit? A Characterization of Open Source Software Repositories. In *2008 16th IEEE International Conference on Program Comprehension*. 182–191.

[6] Andrea Capiluppi and Martin Michlmayr. 2007. From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. *International Federation for Information Processing Digital Library; Open Source Development, Adoption and Innovation;* 234. https://doi.org/10.1007/978-0-387-72486-7_3

[7] Brian Fitzpatrick and Ben Collins-Sussman. 2008. How Open Source Projects Survive Poisonous People. *Talk presented at Google I/O* (2008).

[8] L. P. Hattori and M. Lanza. 2008. On the nature of commits. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*. 63–71.

[9] Dirk Riehle. 2015. The five stages of open source volunteering. In *Crowdsourcing*. Springer, 25–38.

[10] I. Steinmacher, C. Treude, and M. A. Gerosa. 2019. Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects. *IEEE Software* 36, 4 (July 2019), 41–49. https://doi.org/10.1109/MS.2018.110162131

[11] Harold Valdivia-Garcia. 2016. Understanding the Impact of Diversity in Software Bugs on Bug Prediction Models.

[12] Jianguo Wang and Anita Sarma. 2011. Which bug should I fix: helping new developers onboard a new project. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2011, Waikiki, Honolulu, HI, USA, May 21, 2011*, Marcelo Cataldo, Cleidson R. B. de Souza, Yvonne Dittrich, Rashina Hoda, and Helen Sharp (Eds.). ACM, 76–79. https://doi.org/10.1145/1984642.1984661