Performance and Participation in Open Source Software on GitHub

Nora McDonald
Drexel University iSchool
3141 Chestnut St.
Philadelphia, PA 191041 USA
norakmcdonald@gmail.com

Sean Goggins
Drexel University iSchool
3141 Chestnut St.
Philadelphia, PA 191041 USA
outdoors@acm.org

CHI 2013 Extended Abstracts, April 27–May 2, 2013, Paris, France.
ACM 978-1-4503-1952-2/13/04.

Abstract
A few studies have attempted to provide metrics of success in open source software (OSS) projects but the role a code hosting workspace plays in how performance is viewed and measured is little examined. We conducted qualitative, exploratory research with lead and core developers on three successful projects on GitHub to understand how OSS communities on GitHub measure success. These results were obtained in connection with a larger project that is designed to understand the structure of code hosting platforms in relation to participation and performance. We report two main findings. First, lead and core members of the projects we interviewed display a nuanced understanding of community participation in their assessment of success. Second, they attribute increased participation on their projects to the features and usability provided by GitHub.

Author Keywords
Open Source Software; social computing; performance

ACM Classification Keywords
K.4.3: Organizational Impacts (Computer-supported collaborative work)

Introduction

Measuring the relationships between platform design, participation and performance in open source software (OSS) poses a number of challenges for researchers, since each factor influences the others. GitHub, a code hosting repository based on the Git version control system, makes the level of participation on OSS projects readily visible. Understanding how this increased awareness of participation influences the structure of virtual software organizations that use GitHub will shed light on the relationship between platform design, participation and performance measures.

Our goal is to provide much-needed insight on those cross-relationships in order to further development of theories that explain the formation, development and dissolution of OSS organizations. However, "success" in OSS development is not well understood from the standpoint of participants or consistently operationalized across projects. As a first step to understanding how OSS developers themselves think about and measure success, we conducted interviews with 10 lead and core developers on three open source projects.

Background and Literature

Several studies have presented metrics for measuring project success or performance. Mockus and Fielding [6] measure defect density (the number of defects per thousand lines of code) and response times to problem reports contributed by users. Similarly, another study compares the rate of bug resolution on three OSS projects with three commercial projects [5]. One study found a positive correlation between contributor output and OSS projects with a less restrictive license [2]. Others have looked at the role that division of labor has in performance (survival and level of activity) [3]. To date, how OSS community leaders evaluate the success of their projects has been little examined, though one study considers the role of participation transparency in collaborative work on GitHub [1].

GitHub Interface

GitHub allows users to set up a public repository that anyone can fork and use for their own code and/or make changes that they may contribute back as a pull request. Pull requests are a way in which code from one developer is contributed back to a GitHub repository publically. A "fork" is a clone or copy of a repository. Forks are made for two main reasons: first, to use the code in some derivative way; and second, as a precursor to contributing back to the original project through a pull request, which can then be merged by those with access. All of these activities are published in an open, visible stream on GitHub and content and discussion associated with issues, commits and pull requests are also public. Users of GitHub can receive alerts (via desktop clients and email) about code changes, pull requests, comments, issues, etc. for any public project.

Methods & Data

We conducted semi-structured interviews with 10 lead and core members of three large OSS projects hosted on GitHub. Contributors to each project were recruited via email and were given no incentives. All projects have had over a hundred contributors; two, the JavaScript Library and the Web-application framework, have switched from another code hosting infrastructure to GitHub. Our goal was to interview projects that varied in terms of the type of software produced, the project's maturity, and organizational structure.

Interviews lasted approximately 45 minutes and were conducted over Google Hangouts and Skype (and one through email). In addition to asking about respondent's current and past roles and how their responsibilities had changed, we focused on how the project had evolved on the platform, how the respondent/community uses its features and coordinates work and how they measure "success" of the project. Interviews were recorded and transcribed.

Software	Project Organization	Participant
JavaScript Library (JS)	Volunteer & paid developers with support from non-profit	JS1, JS2*, JS3, JS4, JS5, JS6
Web-application Framework (WF)	Volunteer & 1 paid developers with support from non-profit	WF1, WF2*
Server Side Software System (SS)	Volunteer & paid developers with sponsorship from a software & services co.	SS1*, SS2

Table 1. Interview Sample (*Project Lead)

Findings

The most commonly mentioned measures of success on the OSS projects we examined are the number of contributors and contributor growth (WF1, JS1, WF2, JS4, JS5). Our respondents are clear and specific in their comparisons of GitHub with other platforms they have used, and describe GitHub's interface and tools as a significant contributing factor for increasing participation on their projects. Specifically, they note that key forms of early contribution, like contributing a patch to a piece of software, have low barriers on GitHub (WF1, JS1, WF2, SS1, JS2, SS2). Community WF talks about how the level of participation has doubled since moving from their prior source code infrastructure to GitHub (WF1, WF2).

Developers also say that the user experience on GitHub (particularly pull requests) has allowed them to become more "democratic" (WF2, SS2, JS6) and "transparent" (WF2), which leads to greater participation and more opportunities for review and feedback from the community as well as greater "visibility for potential contributors" (JS2).

Developers are also using GitHub's visible metrics of contribution (commits, pull requests, forks, etc) to measure success. Release quality and bug fixing are rarely mentioned as measures of success. This suggests that process measures may be more salient than product quality measures in distributed source code management systems like GitHub.

Number of Contributors

The number of contributors and new contributors are most commonly cited as measures of success among the projects we examined (WF1, JS1, WF2, JS4, JS5). Some developers mention that within the span of the last year the number of new contributors doubled and they attribute this to the GitHub interface:

> "[Moving to GitHub] was probably one of the best moves we ever did for the project. The number of contributions has gone up from – it was in the low twenties; it's over 150 now people who have contributed, so it's much better." (WF2)

> "It's one of the most popular projects on GitHub, and the number of stable contributors has at least doubled since we were on Subversion." (WF1)

Most of our respondents explicitly note that GitHub lowers barriers to participation (WF1, JS1, WF2, SS1, JS2, SS2) and this has led to better code and greater participation. Specifically, contributors don't have to ask

"permission" to submit a patch (SS2). Another key advantage for GitHub is that everyone knows how it works, making it very easy to contribute (JS1, WF2, SS2). "There's no mystery on how to support a patch" (WF2). The consistency across projects means that users don't have to "learn how it works" because all projects "kind of work the same" (WF2). Having a "common language of contribution" has meant that projects don't have to dedicate months figuring out the best way for people to contribute; in the past (on previous platforms) "everyone had their own way of doing things" (SS2). Developers say that GitHub is "lowering the barrier to contributing" because one can simply "push a button" to make their own fork and edit (WF2). They also note that more people seem to be participating (WF1, WF2, SS2).

> "They've also designed the interface around GitHub to lower the barrier to contributing quite – it's extremely low now, where you just have to push a button to make your own fork, do an edit." (WF2)

> "Like, I don't have to fiddle around with their project and then ask them permission to get it in. Like, I can push it and it's saved … And then if I want them to merge it into their project, I just send them a pull request." (SS2)

Another developer talks about their development into a more "egalitarian" project with "super low-value entry" (JS4). He gives an example of a contributor who doesn't know JavaScript, but who can contribute to their website repository (on GitHub) because that individual knows HTML and CSS.

Democratizing through Pull Requests

Developers see considerable value in making code submissions public through pull requests – even if that code is not submitted – because it provides better "visibility for potential contributors" and makes it easier to "bring them onboard" (JS2) and also allows for public discussion to direct the decision, which leads to better outcomes (WF2, JS2, SS2, JS6).

With pull requests, people can make comments "in line" and that means that everyone can see the "whole thing," which provides more "transparency" and gives the person who is submitting greater feedback (WF2), which can lead to individual success that supports continuing contributions. "Transparency" also means that users can see what other users "intentions" are and can discuss changes "before they happen"(WF2). This allows for people who don't want to contribute back to the project to provide input and participate "before the change happens and kind of shape it" (WF2). Developers point to the fact that because GitHub provides this "visibility" (JS2) and allows conversations to take place "in public" it has allowed them to become more "democratic" (in their own words) (WF2, SS2, JS6).

> "For the software development side, the tools that GitHub provides give the project better visibility for potential contributors, and the ability for them to file pull requests that can be reviewed by the team makes it easier to bring them onboard." (JS2)

> "And the whole system of pull requests … it gives you such great tools to talk about lines of codes … it makes the conversation about that contribution a lot more civil … It seems very sort of democratized in that everybody's opinion is equally weighted." (SS2)

Across the projects we examined nearly every significant contribution, regardless of whether members have the ability to directly commit code, is submitted as a pull request (WF1, JS1, WF2, SS1, JS3, JS4, JS5). In other words, even in cases where a user's privileges enable them to avoid the distributed community process, the practices they follow adhere to the more democratic process our respondents describe. These practices enable core members to seek feedback from the community by submitting new code for review (even by new contributors), which results in the code developing with the discussion. It's also a way that core members ensure that the process of integrating new features is more transparent. Community WF says that maintaining community involvement in this way is vital to their project:

> "So when I'm building new features, I like to get some sort of feedback and some sort of like sanity check that what I'm doing isn't totally crazy. So if I do a pull request, it could relate to kind of – create some transparency, about what's gonna be happening and get some community involvement. We have no money, so we have to maintain that transparency and kind of maintain that kind of community involvement for it to stay kind of functional." (WF2)

Visible Metrics of Activity – GitHub Interface
Developers see visible activity on GitHub's interface as important indicators of the success of the project. Specifically, developers mentioned number of commits and committers, which GitHub provides for projects on the "Graphs" tab (JS1, JS5); and number of forks (WF1, JS1) and stars (JS1), which is visible in the upper right hand side of the project's page on GitHub. Another indication is if there are a long list of "closed" pull requests and/or pull requests, which GitHub makes visible on project's "Pull Requests" tab, or the date of the last commit (one way to view this is in the "commit" tab) (JS1).

> "If you look at a project JS commit[s] you see lots of different icons, you know, that there – that there are a lot of people involved with it and that's – that's my bet of how [successful a project is]." (JS5)

> "GitHub immediately facilitates that sort of knowledge right on the main page of any given project. So if you open up a repo, and it says, "Last commit was two years ago," the likelihood of you actually pulling that … slim to none. If I haven't seen activity in – even like the last month – I get a little concerned." (JS1)

External Metrics – Are They Talking About Us?
The health or vibrancy of the community is measurable from the number of people in the IRC channel at any given time or through email list activity (WF2). Another metric community WF tracks is the number of people who visit the API documentation, "because if people are using the documentation, that means they are probably using the code" (WF2). The number of downloads is also mentioned as an important measure of success (WF1, WF2, JS3, JS4). One community has adopted a tool (Splunk) to measure basic statistics like the number of downloads and the number of uses (JS2, JS3). Only one developer mentions concern with quality of the code: "did it break anything, did it slow anything down majorly?" (JS3).

A few developers mention that they are looking for community feedback from Twitter, blogs and conferences (SS1, JS2, JS4). One developer says that if there are blogs getting written or people talking on Twitter about how the project is "great or terrible" that means they are still "relevant" (SS1). Another says that

conference can be a way to maintain contact with the community and get "direct feedback" (JS2).

One developer we interviewed talks about success in terms of having a finished product, with no new features or API changes. They see "value" has having a piece of software that is "not a moving target" and suggest that "the community on top of it can stabilize and grow even faster if the ground isn't moving underneath them" (SS2).

Discussion

Our preliminary findings suggest that contributor growth, community involvement and visible activity are key metrics for success and that code quality, while indirectly related to contribution, is, at best, secondary (or certainly not top of mind). Moreover, it seems clear that the GitHub platform plays a major role not only by providing visible means for judging community involvement and activity, but also, in turn, making it easier to grow pools of participants.

These findings also suggest that greater weight be placed on project process measures, with growing new leadership as indicated by members moving from the periphery toward the core being one key example. Based on these results, we see new opportunities for exploring the concept of distributed leadership, where leadership is shared among members. We suggest future studies to work toward characterizing and quantifying aspects of organization's structure that focus on how the platform and community support growing pools of "newbies" through spontaneous collaboration [4].

References

1. Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. Social coding in GitHub: transparency and collaboration in an open software repository. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ACM (2012), 1277–1286.
2. Fershtman, C. and Gandal, N. The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination. *Available at SSRN:http://ssrn.com/abstract=515282*, (2004).
3. Giuri, P., Ploner, M., Rullani, F., and Torrisi, S. Skills and Division of Labor In An Ecology of Floss Projects: Implications for Performance. *DRUID 10th Anniversary Summer Conference*, (2005).
4. Gronn, P. Distributed leadership as a unit of analysis. *The Leadership Quarterly 13*, 4 (2002), 423–451.
5. Kuan, J. Open-Source Software as Consumer Integration into Production. *Available at SSRN: http://ssrn.com/abstract=259648*, (2001).
6. Mockus, A., Fielding, R.T., and Herbsleb, J.D. Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol. 11*, 3 (2002), 309–346.