Community Engagement and Response to First Time Code Contributors on GitHub

1st Author Name
Affiliation
Address
e-mail address
Optional phone number

2nd Author Name
Affiliation
Address
e-mail address
Optional phone number

Author Keywords
open source; online communities; software development

ACM Classification Keywords
K.6.3. Management of Computing and Information Systems: Software Management

ABSTRACT

We collect data for 13,383 first time code contributions from 45 projects on the website GitHub and analyze behavior of developers before submitting code as well as community response to code contributions. We find that most developers do not engage with the community on GitHub before attempting to submit code changes. We also find that most code submissions do not elicit much community response, and the metrics we use for community response can not predict whether or not a pull request is accepted. Our findings differ from previous research on open source software communities and social theories of learning in communities of practice.

INTRODUCTION

As Free/Libre and Open Source Software (FLOSS) continues to grow and more users become reliant on open source technologies, it is important to understand how this software is developed. Research of open source software can also contribute to existing research in a variety of other fields, including software engineering and CSCW. Recruiting and retaining project contributors is crucial to the success of open source projects, and this study examines the relationships among community engagement, community response, and whether or not code contributions are accepted. Much of the existing FLOSS research that focuses on individual developers discusses motivations for contributing [7, 10]. We, instead, focus on what happens immediately before and after an individual actually contributes. Early success is a strong indicator of future engagement, and our study helps us understand how a developer's social interactions and a community's response influence whether a developer will experience early success. As von Krogh points out, much of the work of socialization falls to the newcomer [17], and we examine the social work newcomers do in order to understand how their behavior influences whether or not their material contributions will be accepted.

Specifically, we examine the behavior of first time code contributors and the community response to their code contributions. We focus on code contributions because they mark the first time a user is attempting to participate in a core act of development. Code contributions can also be accepted or rejected, so our analysis allows us to see how social factors contribute to acceptance within the community. We provide evidence that the level of community engagement and response a user gives and receives does not influence whether his/her code contributions will be accepted. This finding indicates that FLOSS projects may not actually need to foster community engagement or emphasize social processes in order to succeed.

We studied projects hosted on GitHub, a social website that open source software developers use to host their software projects and to browse other developers' projects. It includes many features that are present on social networking sites, such as the ability to follow other users and leave comments on projects. GitHub provides a wealth of data for studying computer supported cooperative work, as it is a centralized location where many different tasks take place. For example, users can create bug reports, submit fixes, and engage in discussions about new features all in one place.

The rest of this paper is organized as follows. First, we provide an overview of GitHub terminology and a literature review on FLOSS and socialization. We then describe our data collection and analysis methods. The discussion of our results and how they differ from previous research in this area follow. We finish by situating our findings in relation to prior research and identifying areas for future work.

Terminology

A software project on the website is referred to as a *repository*. Any user on GitHub can *star* a repository. Users star repositories to be able to easily navigate to them and to receive updates on activity from the repositories. A repository can be private, meaning that it is only visible to the owner and anyone the owner grants access to, or public, meaning that anyone can view it. Our study focuses on public repositories. Many repositories use *issues* to organize workflow around bugs, features, enhancements, and other code developments. Issues can be labeled, assigned to developers, and associated with milestones. They can also be commented on and connected with commits (e.g., associated with the commit that fixes a bug). Issues are also marked ‘open’ or ‘closed’. If a developer wants to contribute to another one of developer’s repositories, he can *fork* the repository, which creates a copy of the project for him to work on. As the developer makes changes to this code, he *commits* his changes. A *commit* is a snapshot of the code at a certain point in time. When the developer is finished, he can submit a *pull request* to the owner of the project. All pull requests for a project are viewable on GitHub, and any user of the site can comment on them. A pull request can have a status of open or closed. A status of open indicates that that owner of the repository has not made a decision about whether or not to include the changes. If the owner of a repository wants to incorporate the changes the developer made, he can *merge* them into the repository. A pull request can be closed without being merged, which means that the changes the developer made were not accepted.

RELATED WORK

FLOSS Research

Research in the development of FLOSS has grown tremendously in the last several years. Crowston et al. [4] note the importance of understanding FLOSS development as it becomes a major social movement with many volunteers contributing to projects and many FLOSS projects becoming integral parts of the infrastructure of modern society (e.g., Apache, Python). Other researchers have emphasized the role that FLOSS research can play in improving current existing research of software engineering, particularly as the importance of understanding large scale software systems in science and industry increases [14].

Existing research approaches FLOSS from many different angles, including motivation of open source developers [**?**, 11, 15]; governance of FLOSS [**?**, **?**, **?**]; and knowledge sharing within FLOSS communities [**?**, 8, **?**]. Our study focuses on the behavior of first-time contributors to FLOSS projects and how the community responds to their contributions. We build on previous studies that describe the social processes of community joining [6, 9, 17]. Given the distributed nature of FLOSS development, our findings potentially contribute to our understanding of virtual work and distributed teams more broadly as well.

GitHub is a relatively new social FLOSS platform and has not been extensively studied. CSCW researchers provide some notable exceptions, however. Dabbish et al. [5] found that the transparency GitHub provides leads to inferences around commitment, work quality, community significance and personal relevance, which supports collaboration and learning. McDonald and Goggins [13] studied how different communities on GitHub measure success and found that most developers measure success in the number of contributors and contributor growth. They also found that developers believed the GitHub interface, in particular the use of pull requests, made communities more democratic and transparent. Choi et al. [3] leveraged data from a sample of GitHub projects to contribute to theories of developer coordination, finding that commits tend to happen in clustered events over time. In all these cases, the social features of GitHub, e.g. the ability to follow other users and view information about them, provide new ways to study social behavior in FLOSS projects. Our study investigates members’ participation in group discussions on the site. While previous studies have tried to combine data from mailing lists and version control [6], GitHub provides a centralized location to study communities in which discussions and code contributions occur in one place. At least with regards to user support, recent research suggests that developers may be moving away from mailing lists to social Q&A sites to respond to user requests for help [**?**]. By focusing on GitHub data, we contribute to understanding developer behavior on this new social platform.

Socialization and Joining Professional Communities

Our study focuses on the behavior of new code contributors and community response to their contributions. We use the theoretical framework of *legitimate peripheral participation* (LPP) [12] in our exploration community joining. LPP describes a process of learning in communities of practice in which newcomers join a community by participating in peripheral tasks and forming relationships to move towards the center of the community. LPP and communities of practice are also often leveraged in other studies of computer mediated communication. For example, Bryant and colleagues [1] note that members of Wikipedia initially become involved through peripheral activities – simple and low risk activities members can take part in to learn more about the community before trying to become major contributors.

Several studies of FLOSS development have used the LPP framework, e.g., to identify core and peripheral community members [9], to understand motivation in FLOSS communities [18], and to develop the concept of a *joining script* [17]. A *joining script* is a set of tasks for new developers to go through before being accepted into the community [17]. Similarly, Ducheneaut [6] found a pattern that resembles LPP in his study of contributors to the Python project.

GitHub’s tools (e.g., issues) enable potential developers to observe both the code and social conventions of a community before joining. Those tools also enable us to collect data to examine a number of legitimate peripheral activities in FLOSS development. For instance, opening and commenting on issues, submitting and commenting on pull requests are all low risk activities new developers can engaged in via GitHub. Existing members can also provide feedback about issues, comments, and code submitted through the site. Prior research relied on version control histories [9] and/or mailing list data [6, 17] to understand social and development activities, and using GitHub data allows us to explicitly connect social and code contributions and to examine their relationships.

Our study examines community response to a user’s first pull request. Research on Wikipedia has shown that community response to newcomers’ contributions affects their long term commitment to the project [2]. We expect a similar effect in FLOSS projects. Shibuya and Tamai [16] found that at least on factor that contributes to difficulty for newcomers to FLOSS projects is a lack of response from core developers.

We focus on response to a user's first pull request as this represents a critical moment in the socialization process of newcomers.

METHODS

Data Collection

Data was collected using the GitHub API.[1] We used a collection of node.js scripts to collect data from the API to store in a MySQL database.[2] In selecting which repositories to use for our analysis, we started with the top 100 most starred repositories on GitHub. We started with this list with the assumption that they were popular repositories that would be maintained by an active community. From these 100, we manually filtered out certain projects that we expected would follow different development patterns than a typical programming project, for example, collections of configuration files for text editors and shells, collections of icons, etc. We also excluded repositories that were used primarily for demonstration or documentation purposes, such as sample web applications to demonstrate use of a certain web framework. After filtering our intial list of 100, 45 repositories remained in our data set for analysis. We consider only pull requests with a status of closed. This resulted in approximately 44,400 pull requests. We further filtered this data by selecting only the first pull request a user submitted to a repository, leaving 13,383 pull requests. The distribution of these pull requests across repositories ranges from 10 to 1,489, with a median of 210. To find merged pull requests, we first filter all pull requests that are marked as merged by the GitHub API, meaning that the project maintainers used GitHub's merge feature to accept the pull request. In some repositories, project maintainers use a different workflow when accepting pull requests, wherein the code changes are accepted, but it is not reflected as merged on GitHub. In most of these cases, there is a standard way of reflecting this in the commit comments, so we use some naive heuristics for identifying these requests by searching commit comments for certain text patterns. For example, in many projects, the project maintainer will manually add the commits from the pull request, and create a new commit with a commit message that follows the pattern "Closes number" where number is the pull request ID on GitHub. Finding merged pull requests using both the status from the GitHub API as well as these text patterns results in finding 5,239, or 39.1% of first pull requests being merged.

Data Analysis

We primarily used logistic regression for our statistical analysis. In all cases, we use various aspects of developer engagement and community response to predict whether or not a pull request is merged. We also included controls for the changes and contributors in a repository, but those controls never produced an improvement in the models. Below we discuss the predictors we used.

Developer's Engagement

To measure developer engagement, we use four variables: the number of pull requests s/he commented on, the number of issues s/he commented on, the number of issues s/he submitted, and his/her reputation. Not all repositories in our data set use GitHub issues, so for those repositories, we only consider the number of previous pull requests a developer commented on. Previous studies indicate that users participate in technical discussions before submitting code [17]. Figure 1 plots the sum of the interaction variables (excluding reputation) by the number of repo.

Community's Response

In addition to measuring the activity of a developer in the community before submitting a pull request, we are also interested in measuring the community response to a given pull request and how that response relates to whether or not a pull request is accepted. We measure this in two ways.

First, we simply count the number of comments on a given pull request. This is used as a basic metric of how much attention a pull request receives. This variable is shown in Figure 2.

Our next analysis of community response focuses on the language of the comments on a pull request. To test whether or not the content of these comments is predictive of whether or not a pull request is merged, we collect the comments for each of our first pull requests. We ignore comments made by the user who submitted the pull request, since we are interested in what other users had to say about it. We also ignore the last comment associated with a pull request, since these often will explicitly say whether or not the maintainer is merging the pull request or not. We are interested in whether the type of language used in the discussion of a pull request is predictive of whether or not it is accepted. We ignore pull requests that only have one comment associated with it. This leaves 5,674 pull requests. Of these, 3,811, approximately 67%, were not accepted. We treat the remaining comments associated with the pull request as one document, and convert them into feature vectors representing the count of each unigram and bigram in the documents, after using a word list to remove stop words. We then train both a logistic regression and naive Bayes classifier using this feature set. The results of testing these classifiers is shown in Table 1. The results shown are the result of running 10-fold cross validation.

Table 1. Classifier results

	Logistic Regression	Naive Bayes
Accuracy	69.6%	70.4%
Precision	56.3%	56.3%
Recall	33.7%	44.4%

RESULTS

Table **??** summarizes the results of our logistic regressions using developer and community behaviors to predict whether pull requests are merged. Overall, we found that participating in technical discussions through pull request comments and issue comments made statistically significant difference on whether or not a pull request was merged, but the difference was so small as to be practically meaningless. We discuss the

[1] http://developer.github.com/

[2] These scripts are available at http://www.github.com/matthewheston/gh-collector.

Predictor	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**
Developer						
Reputation		7.82e-5**	8.17e-5**		8.05e-5**	7.98e-5*
		3.03e-5	3.06e-5		3.08e-5	3.16e-5
			1.00			1.00
Pull Request Comments	0.014**	0.001*	0.013**			0.014*
	0.005	0.0005	0.004			0.006
			1.01			1.01
Issue Comments				0.001	8.12e-4	-9.57e-4
				0.001	8.14e-4	0.001
						1.00
Issues Opened				0.016	0.016	0.016*
				0.008	0.008	0.008
						1.02
Community						
Pull Request Comments			-0.030***			-0.030***
			0.004			0.004
			0.97			0.97
Constant	-0.446***	-0.453***	-0.364***	-0.453***	-0.459***	-0.037***
	0.018	0.018	0.022	0.018	0.018	0.022

*Note: We report the coefficient and standard error for each predictor in each model. In the best fit models (3 and 6), we also report the odds ratios. Predictors' significance are indicated with asterisk where *, **, *** represent p <0.1, p <0.05, p <0.001, respectively.*

Table 2. Logistic Regression Results: Predicting whether a pull request will be merged

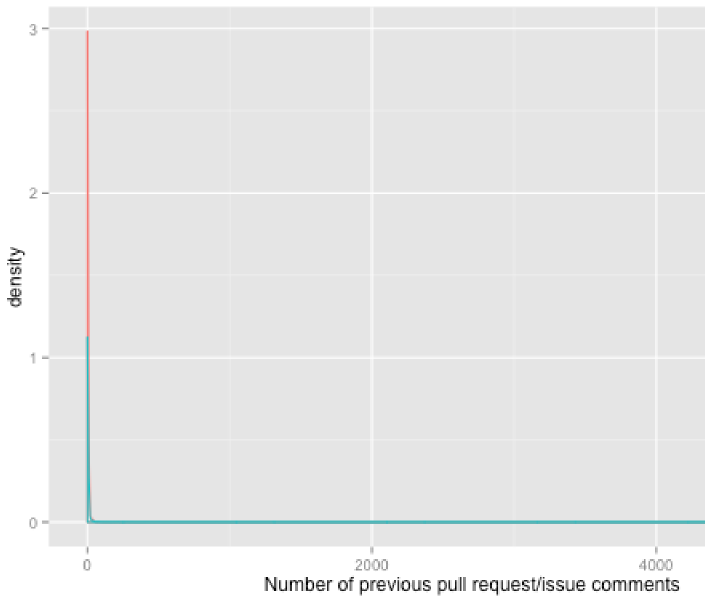

Figure 1. User participation density plots.

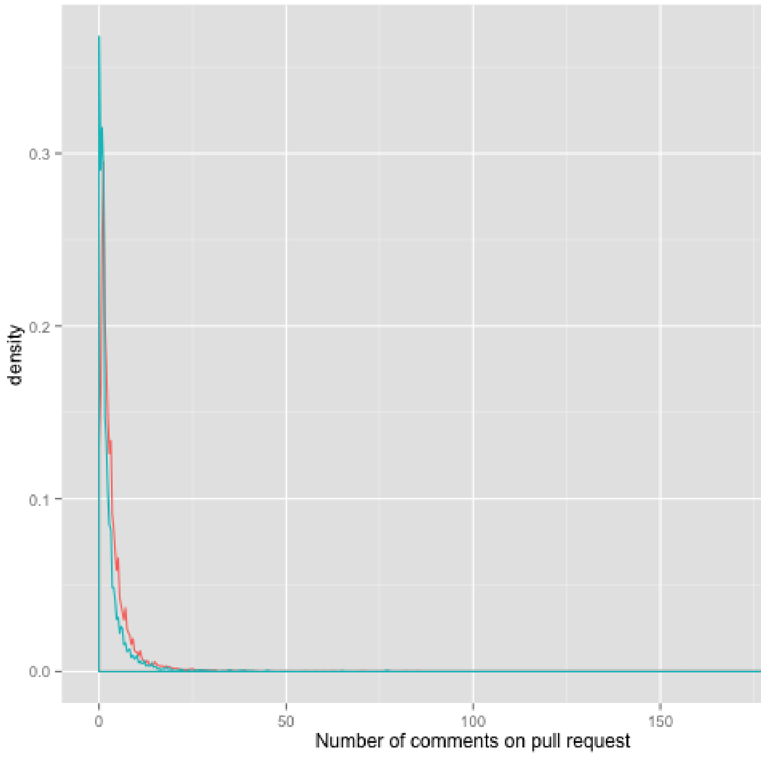

Figure 2. Attention pull request receives density plots.

results of these regressions in detail and provide additional descriptive statistics below.

Developer's Engagement

Pull Requests

We see that user participation for the majority of all first pull requests, both merged and not merged, is 0. This indicates that in general, most users are not attempting to engage in the peripheral activity of commenting on other pull requests before submitting their own. The GitHub interface makes it relatively easy for a user to fork a repository, make changes, and submit the changes for consideration. Previous studies on GitHub have shown that the number of contributions did increase for some projects that moved from other hosting options to GitHub [13]. It is possible this interface lowers the barrier of entry for a developer who wants to contribute to a project, and allows them to bypass participating in the joining script described by von Krogh et al. [17].

We also examine these variables for first pull requests by users who later submit another pull request. We want to see whether or not this "no engagement" pattern continues to hold for users who will become active contributors. Our intuition here is that some users might encounter a bug they fix or desire a feature that they implement, and then submit these changes back to repository. They may not comment on other pull requests as they are not interested in becoming long term members of the community, but rather are just interested in submitting a one time patch. Users who do plan on becoming active members, however, may participate in peripheral activities more. Figure 3 shows a visualization of the same first pull requests, but only for users who submit at least one other pull request at a later point in our data set, and Figure 4 shows the data for users who submit at least 5 more times. Looking at users who submit at least one other time cuts our number of observations from 13,383 to 5,207, indicating that approximately 61% of these pull requests come from users who will not contribute any others. Looking at users who will submit at least 10 more times gives us a total of 1,155 observations.

It is clear that in all these cases, regardless of whether or not they will be continuing to submit other pull requests later, at the time of submitting their first pull request, users are generally *not* participating socially in the community. The previous graphs only consider the number of pull requests a user commented on before submitting their first pull request, so we do not capture how users who submit multiple pull request over time comment on other pull requests over time. In Figure 5 we plot the total number of others' pull requests that a user commented on by how many pull requests they submitted themselves, considering only users who have submitted at least two pull requests. There is not a strong correlation between these variables (Spearman's ρ = 0.44), indicating that users do not necessarily participate in more commenting as they continue to submit more pull requests.

It's worth noting the one extreme outlier present in our data. One user-submitted pull request received 200 comments and was submitted by a user who had commented on 900 previous pull requests. This is an interesting case of a project maintainer, who has commit access and wouldn't typically need to submit a pull request to submit changes, creating a pull request for commits related to a major upgrade in the project. By creating a pull request, he was able to document all the changes associated with this change and allow community members to ask questions or comment on the changes. He has a high number of previous comments since he is in charge of accepting pull requests and often posts a comment such as, "Accepted" when accepting a pull request. This outlier likely received so many comments because many other developers were asking questions or voicing their opinions. This example demonstrates the different ways pull requests may be used in different projects, and how the way they are used may change depending on the type of user submitting them.

Issues

Community's Response

In Figure 1, we see more variance in the number of comments on first pull requests than we did with the number of pull requests users commented on before submitting. However, this variable does not seem to be a good predictor of whether or not a pull request is merged, since both the merged and not merged distributions follow a similar pattern. It seems just viewing the amount of activity a pull request receives is not enough to explain whether or not it gets merged.

Training classifiers using the comment text may help address this problem, as this can capture the valence of the comments, rather than just the raw number. However, the low recall rates we see in Table 1 indicate that the text data is not sufficient to distinguish positive cases. We list the top five features for each class in Table 3. Despite leaving out the last comment from each comment thread to avoid words that explicitly describe the action being taken, we still see these in our top features. Both "land" and "landed" are used in some repositories when a commit is merged, but not through the GitHub interface to indicate the git commit hash where the pull request commits were merged. We also see "closing" in the negative class, which shows explicit action being taken. Some of the larger projects also skew our results, as we see project names appear as the highest ranking features of the negative class. Some of our top features do seem fitting. The phrase "lgtm" which stands for "looks good to me" seems indicative of positive feedback from the community.

Our sample size of 5,674 is relatively small, but it is interesting to note that only 42% of the first pull requests in our data set have more than 1 comment associated with them. Our research question that drove these experiments was how community response affects whether or not a pull request is accepted. However, we see that the majority of pull requests don't attract any community attention.

Table 3. Classifier top features

Merged		Not merged	
land	0.981	bootstrap	-0.780
landed	0.894	spree	-0.875
cla thanks	0.874	closing	-0.641
fine	0.721	already	-0.600
lgtm	0.688	maybe	-0.597

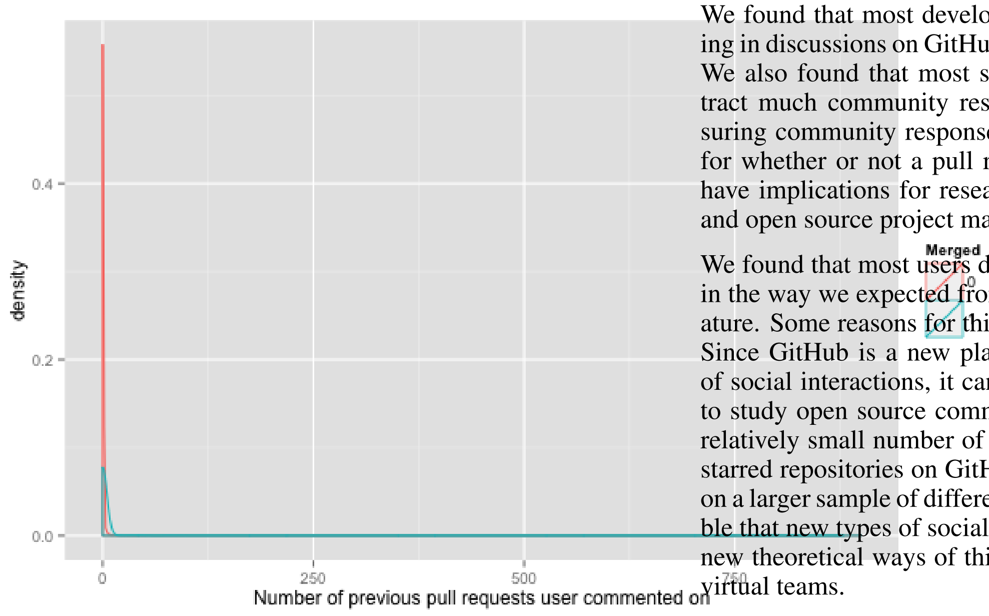

Figure 3. User participation density plots for users who submit at least one other pull request in our data set.

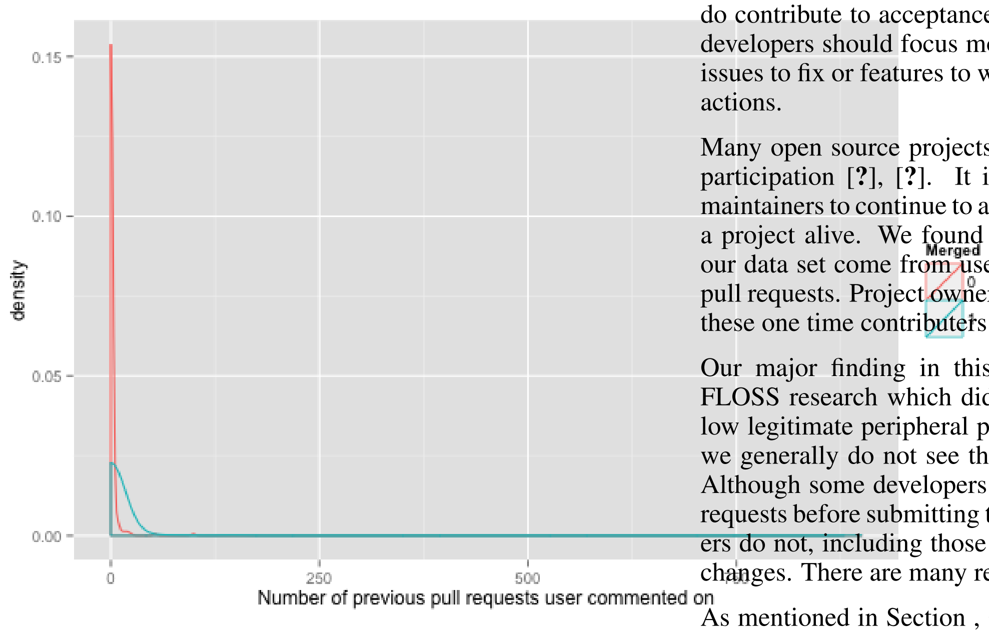

Figure 4. User participation density plots for users who submit at least ten other pull requests in our data set.

DISCUSSION

In this study, we analyzed community engagement and community response of first time code contributors on GitHub. We found that most developers do not engage by participating in discussions on GitHub before submitting code changes. We also found that most submitted pull requests do not attract much community response, and our attempts at measuring community response did not provide good predictors for whether or not a pull request is accepted. Our findings have implications for researchers, open source contributors, and open source project maintainers.

We found that most users do not engage with the community in the way we expected from previous FLOSS and LPP literature. Some reasons for this are discussed below in Section . Since GitHub is a new platform that encourages new types of social interactions, it can be used as a new source of data to study open source communities. Our study focused on a relatively small number of repositories pulled from the most starred repositories on GitHub. Further work should be done on a larger sample of different types of repositories. It's possible that new types of social websites like GitHub will require new theoretical ways of thinking about distributed work and virtual teams.

The majority of first pull requests in our data set were not accepted, but community engagement is not a good predictor or whether or not they are accepted, and most developers do not engage before submitting a pull request, even if they end up becoming active code contributors. The implication for developers who want to become involved in an open source project on GitHub is that they do not necessarily need to participate in peripheral activities before submitting pull requests. Although our study did not identify which factors do contribute to acceptance of pull request, it's possible that developers should focus more on things like finding relevant issues to fix or features to work on rather than on social interactions.

Many open source projects fail due to insufficient volunteer participation [?], [?]. It is therefore important for project maintainers to continue to attract volunteer developers to keep a project alive. We found that 61% of first pull requests in our data set come from users who will not submit any other pull requests. Project owners should consider ways to convert these one time contributers to regular active contributors.

Our major finding in this study is that, despite previous FLOSS research which did indicate social patterns that follow legitimate peripheral participation framework [6, 9, 18], we generally do not see this pattern in our GitHub data set. Although some developers do leave comments on other pull requests before submitting their own, the majority of developers do not, including those that will continue to submit code changes. There are many reasons this might be the case.

As mentioned in Section , GitHub's interface makes it fairly easy to submit changes. Users only need to click a button to create a copy of the repository that they can make commits on, and then click another button to submit those commits as a pull request when they are finished. This process may lower barriers to entry for new developers. While previous studies of FLOSS communities have focused on mailing lists and centralized version control repositories, the distributed nature of git may be altering the social patterns that take place in developer networks. This study may suggest we need to alter existing social theory as new interfaces change social interactions.

Limitations

This study only focused on data from GitHub, and our notion of community engagement by developers was limited to activity by developers on GitHub. While we have shown that this data is not sufficient to predict whether or not a pull request is merged, further studies should create joint data sets merging GitHub data with data from other sources, such as mailing lists, forums, or chat rooms to test whether or not developers engage with the community using these other platforms, and how those social interactions affect their acceptance in the community. Though GitHub is designed to house all of a project's information, many projects still use other channels for discussion that aren't included in our data. One difficulty in creating these types of data sets is identify merging, the process of matching different logins from different services to the same physical person, but a number of techniques have been proposed to assist in this process [**?**, **?**, **?**].

It is also important to note that the GitHub platform is used in different ways across projects. As described in Section , there was some work required to identify merged pull requests due to the different ways a project maintainer accepts the pull request, e.g. through the GitHub interface or not. In Section , we discuss an outlier in our data where a user who did not typically contribute code through the pull request mechanism did use this feature in order to allow community feedback and questions. There are many other examples of differences in use of the GitHub platform. Some projects, for example, may require all developers to submit pull requests for the purposes of code review, while others may grant commit access to certain developers, allowing them to bypass the pull request mechanism altogether. Although we did account for differences in accepting pull requests, there may be other nuances across projects that affect the data we collected that are harder to detect when working with data at scale.

Future Work

In examining developer engagement and community response, our study was primarily concerned with social factors within open source software development communities. It's possible that looking at only social factors is not enough to understand what contributes to acceptance on GitHub. Static analysis tools may be used in addition to the metrics we used in this study to further explore how the code itself affects whether or not a pull request is accepted.

In studying how communities respond to pull requests, we found in most cases, there was little community response. As mentioned above, attraction of new developers and retention of developers is important to project success. Future work should explore how community response can affect developer retention in GitHub projects.

In examining the top features for the classifiers we trained in studying community response, we found some issues. Our goal in filtering the text collected was to avoid words that explicitly described action being taken. We still found examples of these types of words in both our positive and negative classes. In the case of the positive (merged) class, the top feature was "land." This feature affects classifier performance since this word is used often in positive cases, but only in a few repositories. Similarly, project titles were often predictive of acceptance but those terms are not useful for predicting acceptance across projects. Future work studying linguistic features in this way should account for these types of project-specific vocabularies.

Our analysis focused only on behavior of developers leading up to the submission of their first pull request and the community response to that pull request. We consider this a critical moment in the the socialization process since this task is often considered as a core task that a developer can participate in. Future studies should invetigate how community response at this moment affects continued contribution and commitment to the project by developers.

Finally, to better understand the way that the GitHub interface affects developer behavior will require further qualitative study. While there have been surveys of GitHub developers previously [13], there remains a lot of work left to do in this area. The traces of behavior visible in GitHub data tell us only part of the story.

CONCLUSION

In this study, we measured developer engagement and community response of first time code contributors to GitHub and tried to predict whether code contributions would be accepted based on that data. We found that most users do not participate in social GitHub features before submitting code. In attempting to measure community response, we found that most code contributions do not receive much community response, and that both the raw amount of comments on a GitHub pull request as well as the language in those comments were not good predictors of whether or not the pull request is accepted. These findings suggest that social interaction is not a necessary component of successful FLOSS projects.

REFERENCES

1. Bryant, S. L., Forte, A., and Bruckman, A. Becoming wikipedian: Transformation of participation in a collaborative online encyclopedia. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '05, ACM (New York, NY, USA, 2005), 110.

2. Choi, B., Alexander, K., Kraut, R. E., and Levine, J. M. Socialization tactics in wikipedia and their effects. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, ACM (2010), 107–116.

3. Choi, J., Choi, J., Moon, J. Y., Hahn, J., and Kim, J. Herding in open source software development: an exploratory study. In *Proceedings of the 2013*

conference on Computer supported cooperative work companion, CSCW '13, ACM (New York, NY, USA, 2013), 129134.

4. Crowston, K., Wei, K., Howison, J., and Wiggins, A. Free/Libre open-source software development: What we know and what we do not know. *ACM Comput. Surv. 44*, 2 (Mar. 2008), 7:17:35.

5. Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, ACM (New York, NY, USA, 2012), 12771286.

6. Ducheneaut, N. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW) 14*, 4 (Aug. 2005), 323–368.

7. Hann, I.-H., Roberts, J., Slaughter, S., and Fielding, R. Economic incentives for participating in open source software projects. In *ICIS 2002 Proceedings* (2002).

8. Hemetsberger, A., and Reinhardt, C. Collective development in open-source communities: An activity theoretical perspective on successful online collaboration. *Organization Studies 30*, 9 (Sept. 2009), 987–1008.

9. Huang, S.-K., and Liu, K.-m. Mining version histories to verify the learning process of legitimate peripheral participants. In *Proceedings of the 2005 international workshop on Mining software repositories*, MSR '05, ACM (New York, NY, USA, 2005), 15.

10. Krishnamurthy, S. On the intrinsic and extrinsic motivation of free/libre/open source (FLOSS) developers. *Knowledge, Technology & Policy 18*, 4 (2006), 17–39.

11. Lakhani, K., and Wolf, R. G. Why hackers do what they do: Understanding motivation and effort in Free/Open source software projects. SSRN Scholarly Paper ID 443040, Social Science Research Network, Rochester, NY, Sept. 2003.

12. Lave, J., and Wenger, E. *Situated learning: Legitimate peripheral participation*. Learning in doing: Social, cognitive, and computational perspectives. Cambridge University Press, New York, NY, US, 1991.

13. McDonald, N., and Goggins, S. Performance and participation in open source software on GitHub. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, ACM (New York, NY, USA, 2013), 139144.

14. Scacchi, W. Free/Open source software development: Recent research results and methods. In *Advances in Computers*, Marvin V. Zelkowitz, Ed., vol. Volume 69 of *Architectural Issues*. Elsevier, 2007, 243–295.

15. Shah, S. K. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science 52*, 7 (July 2006), 1000–1014.

16. Shibuya, B., and Tamai, T. Understanding the process of participating in open source communities. In *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS'09. ICSE Workshop on*, IEEE (2009), 1–6.

17. von Krogh, G., Spaeth, S., and Lakhani, K. R. Community, joining, and specialization in open source software innovation: a case study. *Research Policy 32*, 7 (July 2003), 1217–1241.

18. Ye, Y., and Kishida, K. Toward an understanding of the motivation of open source software developers. In *25th International Conference on Software Engineering, 2003. Proceedings* (May 2003), 419–429.

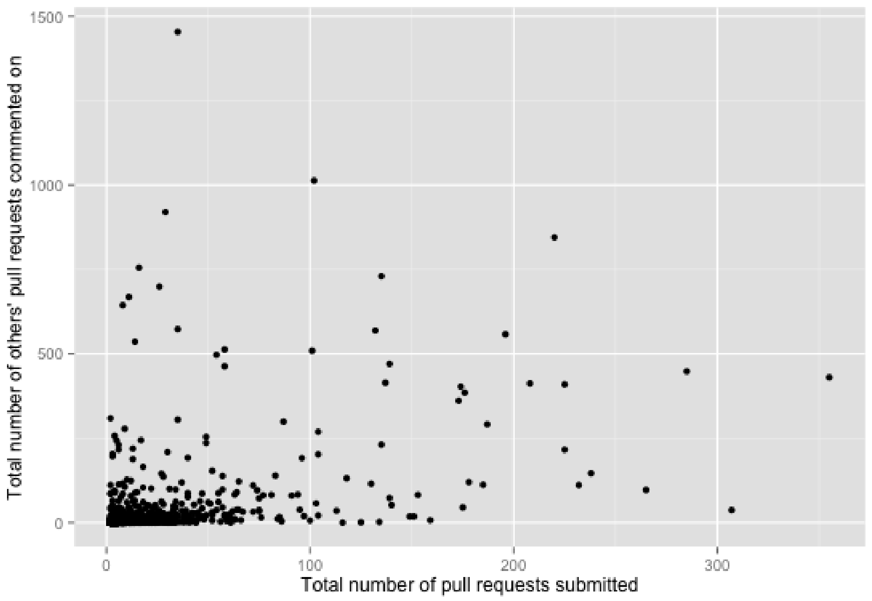

Figure 5. Total number of pull requests commented on and total number of pull requests submitted for each user.