ACCEPTANCE OF CODE CONTRIBUTORS ON GITHUB

BY

MATTHEW HESTON

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Information Architecture
in the Graduate College of the
Illinois Institute of Technology

Approved ______________________
Advisor

Chicago, Illinois
May 2014

ACKNOWLEDGMENT

This dissertation could not have been written without Dr. X who not only served as my supervisor but also encouraged and challenged me throughout my academic program. He and the other faculty members, Dr. Y and Dr. Z, guided me through the dissertation process, never accepting less than my best efforts. I thank them all.

(Don't copy this sample text. Write your own acknowledgement.)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Your Abstract goes here!

CHAPTER 1
INTRODUCTION

GitHub is 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 on one website. In this study, we use statistical and machine learning methods on data from GitHub repositories to explore how different factors may affect the acceptance of code changes from first time contributors.

The rest of this paper is organized as follows. The rest of this chapter provides a literature review on the subject, establishing the importance of studying open source software development, situating it within a context of virtual work and computer mediated communication, and reviewing a theoretical basis we use to inform our empirical methods. Chapter 2 describes our data collection and data analysis methods. The results of our experiments are discussed in Chapter 3. We conclude in Chapter **??** and discuss opportunities for future work.

1.1 Related Work

1.1.1 FLOSS Research. Research in the development of free/libre open source software (FLOSS) has grown tremendously in the last several years. Crowston et al. [3] 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 societ 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. Other studies 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 insustry increases [15].

Existing research approaches FLOSS from many different angles, including motivation of open source developers [7], [11], [16]; governance of open souce projects [9], [14], [**?**]; and knowledge sharing within FLOSS communities [6], [8], [17]. Our study focuses on the behavior of first time contributors to FLOSS projects and community response to their contributions. We build on previous studies that describe the social processes of community joining [5], [10], [20]. Given the distributed nature of FLOSS development, our findings contribute to current descriptions of virtual work and distributed teams.

Previous studies have used version control histories to verify learning processes of new members in FLOSS projects [10]. GitHub, however, has not been extensively studied as it is a relatively new social platform. Dabbish et al. [4] study how GitHub as a social application provides transparency and how that transparency affects collaboration and learning. McDonald and Goggins [13] study how different communities on GitHub measure success. Choi et al. [2] study a sample of GitHub based projects to contribute to theories of developer coordination. 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 [5], GitHub provides a centralized location to study communities in which discussion and code contribution all 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 [18]. By focusing on GitHub data, we contribute to understanding developer behavior on this new social platform.

1.1.2 Communities of Practice. 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. Several studies of FLOSS development have used the LPP framework. Huang and Liu [10] mine version control history to construct developer networks and identify core and peripheral community members. Ducheneaut [5] finds a pattern that resembles LPP in his study of contributors to the Python project. Ye and Kishida [21] use LPP to ground their theory of motivation in open source communities. This concept has been explored in other studies of computer mediated communication. In their study on members of Wikipedia, [1] note that members initially become involved through peripheral activities. These are simple and low risk activities members can take part in to learn more about the community before trying to become major contributors. Similarly, [20] from observing open source communities generate the construct of a *joining script*, where each project has a set of tasks for new developers to go through before being accepted into the community.

Our study seeks to which factors contribute to the acceptance of code contributions of first time contributors. We use LPP as a theoretical framework to ground our experiments.

CHAPTER 2
METHODS

2.1 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 it and to receieve updates on activity from the repository. 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.

2.2 Data Collection

Data was collected using the GitHub API.[1] We collected pull requests from 45 different repositories. The repositories selected came from the top 100 most starred repositories on GitHub. We chose popular repositories with the assumption that they would be maintained by an active community. We only consider pull requests with a status of closed. This resulted in approximately 44,400 pull requests. For most of our experiments, we will use what we refer to as first pull requests, by which we

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

mean the first pull request a user submitted to a repository. Filtering for these pull requests results in a sample size of 13,383. 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. This results in finding 5,239, or 39.1% of first pull requests being merged.

2.3 Data Analysis

CHAPTER 3
RESULTS

3.1 Communities of Practice

In this section, we explore both how a user engages with the community before submitting their pull request, as well as try to measure the community response to the pull request. We consider the main peripheral activity a user can participate in on GitHub is commenting on other pull requests. For each first pull request in our dataset, we count the number of other pull requests the user commented on before submitting. Based on the these previous findings, we would expect to see a positive correlation between this number and the likelihood that a pull request is recieved. We also count the number of comments that each first pull request recieves, with the assumption that this variable can be used to measure the amount of community interest in a given pull request. We plot these variables in Figure 3.1.1.

3.1.1 User Participation. 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 [20].

We also examine these variables for first pull requests by users who later submit another pull request. 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. Figure 3.1.1 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 3.1.1 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 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 3.1.1 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 $\rho = 0.44$), indicating that users do not necessarily participate in more commenting as they continue to submit more pull requests.

3.1.2 Attention Pull Request Receives. In Figure 3.1.1, 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, there is clearly no linear separation of merged and not merged pull requests using this variable, so it seems just viewing the amount of activity a pull request receives is not enough to explain

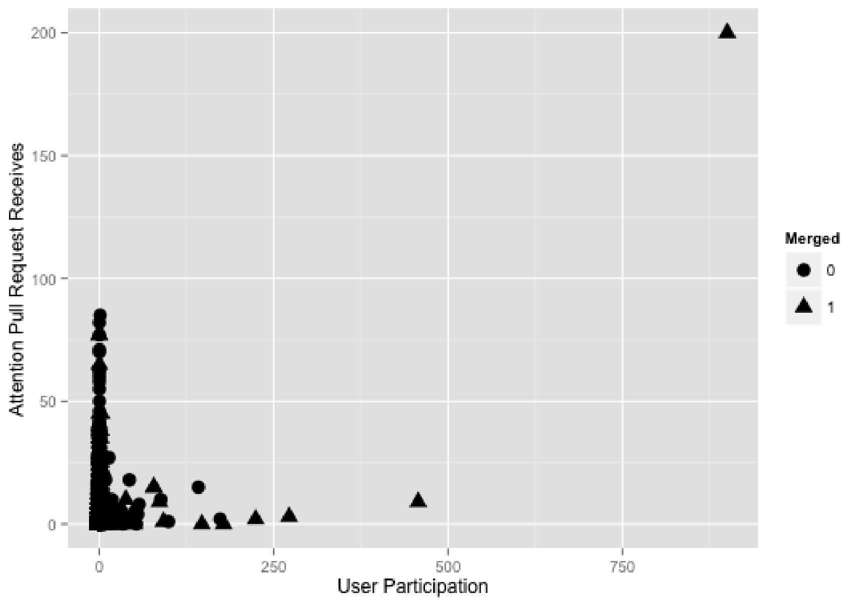

Figure 3.1. User participation and attention a pull request recieves for all first pull requests.

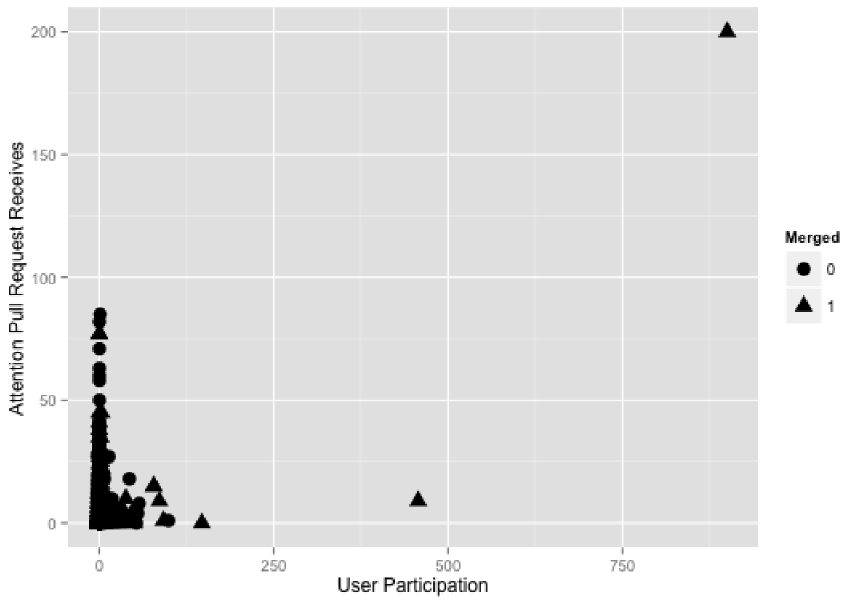

Figure 3.2. User participation and attention a pull request recieves variables for users who submit at least one other pull request in our data set.

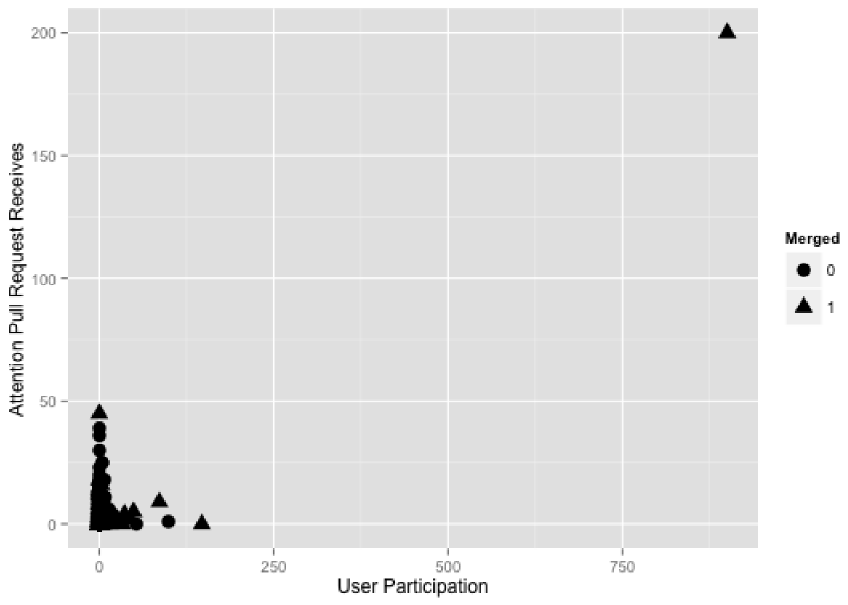

Figure 3.3. User participation and attention a pull request recieves variables for users who submit at least 10 other pull requests in our data set.

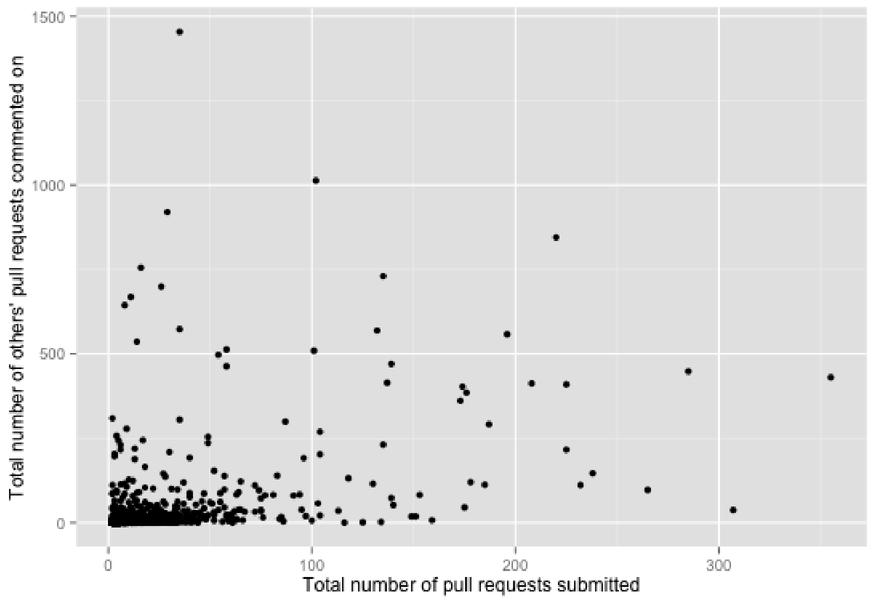

Figure 3.4. Total number of pull requests commented on and total number of pull requests submitted for each user.

whether or not it gets merged.

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 more interested in if 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 gives us a sample size of 5,674. 3,811, approximately 67% are unmerged. 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, and train both a logistic regression and naive bayes classifier using this feature set. The results of testing these classifiers is shown in Table 3.1.2. The results shown are the result of running 10-fold cross validation. The low recall rates indicate that the text data is not sufficient to distinguish positive cases. Of course, 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.

Table 3.1. Classifier results

	Logistic Regression	Naive Bayes
Accuracy	69.6%	70.6%
Precision	56.0%	60.3%
Recall	36.1%	30.7%

3.2 First Mover Advantage

[19] find what they call a *first-mover advantage* in the editing of Wikipedia

articles, wherein the first contribution to a page tends to survive longer and recieve less modifications than following contributions. In this section, we explore how the acceptance of pull requests changes over time.

In Figure 3.2, we plot the average number of both merged and unmerged pull requests over a 12 month period.

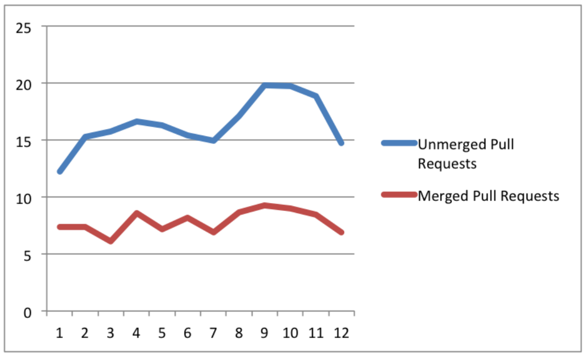

Figure 3.5. Average number of merged and not merged pull requests over 12 months.

BIBLIOGRAPHY

[1] Susan L. Bryant, Andrea Forte, and Amy Bruckman. 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, page 110, New York, NY, USA, 2005. ACM.

[2] Joohee Choi, Junghong Choi, Jae Yun Moon, Jungpil Hahn, and Jinwoo Kim. Herding in open source software development: an exploratory study. In *Proceedings of the 2013 conference on Computer supported cooperative work companion*, CSCW '13, page 129134, New York, NY, USA, 2013. ACM.

[3] Kevin Crowston, Kangning Wei, James Howison, and Andrea Wiggins. Free/Libre open-source software development: What we know and what we do not know. *ACM Comput. Surv.*, 44(2):7:17:35, March 2008.

[4] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 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, page 12771286, New York, NY, USA, 2012. ACM.

[5] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, 14(4):323–368, August 2005.

[6] Megan Lee Endres, Steven P. Endres, Sanjib K. Chowdhury, and Intakhab Alam. Tacit knowledge sharing, self-efficacy theory, and application to the open source community. *Journal of Knowledge Management*, 11(3):92–103, June 2007.

[7] Yulin Fang and Derrick Neufeld. Understanding sustained participation in open source software projects. *Journal of Management Information Systems*, 25(4):9–50, April 2009.

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

[9] Eric von Hippel and Georg von Krogh. Open source software and the Private-Collective innovation model: Issues for organization science. *Organization Science*, 14(2):209–223, April 2003.

[10] Shih-Kun Huang and Kang-min Liu. 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, page 15, New York, NY, USA, 2005. ACM.

[11] Karim Lakhani and Robert G. Wolf. 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, September 2003.

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

[13] Nora McDonald and Sean Goggins. Performance and participation in open source software on GitHub. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 139144, New York, NY, USA, 2013. ACM.

[14] Siobhn OMahony. Guarding the commons: how community managed software projects protect their work. *Research Policy*, 32(7):1179–1198, July 2003.

[15] Walt Scacchi. Free/Open source software development: Recent research results and methods. In Marvin V. Zelkowitz, editor, *Advances in Computers*, volume Volume 69 of *Architectural Issues*, pages 243–295. Elsevier, 2007.

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

[17] Sulayman K. Sowe, Ioannis Stamelos, and Lefteris Angelis. Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software*, 81(3):431–446, March 2008.

[18] Bogdan Vasilescu, Alexander Serebrenik, Prem Devanbu, and Vladimir Filkov. How social Q&#38;A sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work &#38; Social Computing*, CSCW '14, page 342354, New York, NY, USA, 2014. ACM.

[19] Fernanda B. Vigas, Martin Wattenberg, and Kushal Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, page 575582, New York, NY, USA, 2004. ACM.

[20] Georg von Krogh, Sebastian Spaeth, and Karim R Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, July 2003.

[21] Yunwen Ye and K. Kishida. Toward an understanding of the motivation of open source software developers. In *25th International Conference on Software Engineering, 2003. Proceedings*, pages 419–429, May 2003.