Influence of Social and Technical Factors for Evaluating Contribution in GitHub

Jason Tsay, Laura Dabbish, James Herbsleb
School of Computer Science and Center for the Future of Work, Heinz College
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213 USA
{jtsay, dabbish, jdh}@cs.cmu.edu

ABSTRACT

Open source software is commonly portrayed as a meritocracy, where decisions are based solely on their technical merit. However, literature on open source suggests a complex social structure underlying the meritocracy. Social work environments such as GitHub make the relationships between users and between users and work artifacts transparent. This transparency enables developers to better use information such as technical value and social connections when making work decisions. We present a study on open source software contribution in GitHub that focuses on the task of evaluating pull requests, which are one of the primary methods for contributing code in GitHub. We analyzed the association of various technical and social measures with the likelihood of contribution acceptance. We found that project managers made use of information signaling both good technical contribution practices for a pull request and the strength of the social connection between the submitter and project manager when evaluating pull requests. Pull requests with many comments were much less likely to be accepted, moderated by the submitter's prior interaction in the project. Well-established projects were more conservative in accepting pull requests. These findings provide evidence that developers use both technical and social information when evaluating potential contributions to open source software projects.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments - *Integrated Environments*; H.4.3 [**Information Systems Applications**]: Communications Applications.

General Terms

Management, Measurement, Human Factors

Keywords

GitHub; transparency; open source; social computing; signaling theory; social media; contribution

1. INTRODUCTION

Traditionally, open source software (OSS) communities have been characterized as meritocracies [26] where "code is king" and decisions are based solely on technical merit. Multiple popular open source software foundations such as the GNOME Foundation [10], Apache Software Foundation [14], and Mozilla Foundation [21] officially describe themselves as meritocracies. For example, in the case of Mozilla, “authority is distributed to both volunteer and employed community members as they show their abilities through contributions to the project” [21]. These “abilities” are generally assumed to be technical expertise brought to the software project by various developers.

Previous studies on open source software suggest that there are many more factors that influence contribution evaluation beyond technical merit. In fact, prior work suggests that there exists a complex social structure around contribution in open source software [8]. New contributors to traditional open source projects are expected to “lurk” or monitor project mailing lists before even attempting contributions. These projects have complex socialization processes that need to be undertaken before accepting technical contributions [17].

With the advent of social media and distributed version control systems, many open source software projects operate with an unprecedented degree of transparency. With social media attached to the development activity, relationships between developers and actions on code are made visible, and developers use these as signals from which they infer important but hidden qualities [6]. For example, project managers are able to view all of the prior projects that a newcomer might have participated in and evaluate them as signals of developer skill before deciding whether or not to accept a contribution. Newcomers looking for open source projects to join might investigate what prior contribution attempts look like and how they have fared, as signals of openness and project norms. Studies on these kinds of transparent environments suggest that developers make complex inferences about other developers and projects using these kinds of information [5][19].

With information available from potentially millions of developers and millions of repositories in these transparent work environments, then what information do software developers use when evaluating software contributions? Conventional wisdom on open source projects suggests that technical merit of the contribution itself should be all-important [26]. Prior literature on traditional open source suggests that prior interactions with a project and project culture should also have an important effect. In a transparent environment, the visible relationships between users may also have an important effect on contribution decisions. These environments also make explicit the relationship between users and work artifacts such as repositories information that project managers may be making use of. We aim to better understand how different signals are used by software project managers in order to evaluate contributions in open source projects.

ICSE'14, May 31 – June 7, 2014, Hyderabad, India

With this work, our goal is to deepen our understanding of how information in transparent open source software environments is used to evaluate contributions. In particular, we investigate signals and cues associated with contribution acceptance. By better understanding how developers in transparent work environments use information available to evaluate contributions, we can inform tool design for similar work environments. For example, site designers may choose to make important signals more salient to developers. To answer these questions, we performed an analysis of contribution from thousands of projects on the social open source project-hosting site GitHub. For our unit of analysis, we use the pull request, one of the primary methods for code contributions in GitHub, where a developer submits a code change to a project that notifies the project core members for a review of the code contribution to evaluate whether or not to accept the change. GitHub is an example of a new class of work environment that offers greater transparency of work actions and social relationships.

We found that both technical and social signals had strong associations with contribution acceptance. In particular, social signals of the connection between the submitting user and the user managing the contribution were especially associated with contribution acceptance. Contributions with many associated comments were much less likely to be accepted, perhaps due to contention between the submitter and the core project team. The negative influence of comments was moderated, however, by the submitter's prior interaction in the project. In the following sections we consider related research on the contribution practices of open source software projects and online communities in order to generate hypotheses, describe our multi-level logistic model of pull request acceptance, report the results of our analysis, and discuss the implications of our findings.

2. TRANSPARENCY AND OPEN SOURCE

Previous work on open source software projects and developers suggests that there is a complex social and technical structure around the concept of making contributions to projects. We position our work in the literature on newcomers making contributions in online communities more generally. We also examine new software development environments that make use of transparency and social media features, which sheds light on the kinds of signals that are visible and the inferences developers make. Informed by previous work, we generate hypotheses to test in our analysis of contributions in GitHub.

2.1 Open Source Software

As open source software often relies on the volunteer efforts of software developers, the survival and well-being of open source software projects often depends on attracting contributions from the larger community [4]. Developers choose to offer contributions to open source software projects for a variety of reasons. Hars and Ou [13] performed a survey on the motivations of open source developers. They found that developers would contribute their time and effort for various reasons, with self-determination and developing human capital the most highly ranked. Other highly ranked motivations include peer recognition, self-marketing, and a developer's personal need for the software project. Lakhani and Wolf [18] and Roberts et al. [25] found that open source developers are motivated by both intrinsic motivations such as intellectual stimulation, and extrinsic motivations such as improving skills.

Literature on the contribution process for open source software projects suggests that accepting contributions, especially from unknown developers, is a complex process. Krogh et al. [17] found in their study of the contribution process in the Freenet open source project that there are "joining scripts" that successful newcomers follow before offering contributions. These joining scripts involve participating in prior activity such as lurking on the project's mailing list, participating in technical discussions, and reporting bugs. Developers that offered technical contributions without following this joining script tended not to have their contributions accepted into the project. Ducheneaut [8] made a similar observation in the Python project, noting a progressive socialization process that requires both displaying technical skills and creating the right social relations. In order for contributions to be accepted into the project, the contribution must both be technically sound and be vetted by core members of the project. For successful and complete socialization (becoming an "insider"), a developer needs to recruit core members of the project as a network of "allies".

Shah [27] observed that this contribution process also leads to evolution for a developer's level of participation in the project. Most developers make simple initial contributions such as bug fixes in order to fulfill some need. A number of these developers choose to continue to participate in the project, evolving from a need-based participation to a hobbyist. Often, these developers will also gain committer rights or the right to freely commit their changes directly into the project.

As open source software projects evolve, their contribution needs tend to change as the project matures. Nakakoji et al. [22] observed that as open source projects evolve, the communities around the project co-evolve along with the open source software system. Contributions to the project influence the transformation of both the software system and the community. Nakakoji et al. also defined at least three different classes of open source projects that evolve into each other. These project classes each have their own unique contribution needs and selection criteria. For example, a Service-Oriented OSS system like PostgreSQL tends to be very conservative in terms of accepting contributions due to a need for stability. Stewart and Gosain [28] also found that project maturity in open source software projects on SourceForge moderates both objective and subjective performance outcomes. For example, the effect of task completion on perceived effectiveness is more positive for more mature projects.

2.2 Contribution in Online Communities

Open source projects are a form of online community. To survive and thrive, online communities face the challenge of attracting and evaluating contributions. Kraut and Resnick [16] claim that when dealing with newcomers, successful online communities must meet a number of challenges: attracting newcomers, selecting among the newcomers, retaining newcomers, socializing newcomers, and protecting existing members from potential problems newcomers may bring. When evaluating newcomers, communities will often screen potential members by using signals of whether or not a newcomer is a good fit. In order to gather information about these signals, diagnostic tasks are often used such as solving CAPTCHAs to screen automated attackers or acquiring experience points and weapons to signal character prowess in the online game World of Warcraft. An open question is whether transparent work environments already provide these kinds of signals by virtue of making past work activity visible.

In the community of Wikipedia, Bryant et al. [3] found that some newcomers will transition from making peripheral contributions to specific articles into core users that help maintain Wikipedia and its community as a whole. Newcomers learn the conventions and contribution rules of the Wikipedia community through observation (lurking) and direct mentoring from more experienced users. Related to this concept of mentoring is a community-wide norm of "don't bite the newcomers". The nature of users' contributions also tends to change as newcomers become more socialized, from purely making edits in articles to also participating in community discussions, administrative duties, and "meta" tasks.

Iriberri and Leroy [15] found that online communities have multiple lifecycle stages with different contribution needs. For example, during the earlier Growth stage, communities are more concerned with attracting new members and supporting interactions while the later Maturity stage, communities may prefer to recognize contributions and increase visibility of certain members.

2.3 Transparent Work Environments

We use GitHub as an example of a new class of transparent software environments that incorporate social media features to make work more visible. Previous qualitative research on GitHub by Dabbish et al. [5] showed that developers are able to make a variety of subtle inferences about other developers and projects using the social media cues. They use these inferences in practical ways, for instance to help manage their projects, discover user needs, and recruit developers. This research also suggested that the feed of project activity generated by social media strongly influences who and what developers attend to. Developers in this study used signals of community attention to a project or event in the feed to determine if a project was worth using or a discussion was worth reading. Project managers, especially those in popular projects that received many contributions (pull requests) per day, would make inferences about the quality of code contributions and submitter competence based on these signals. In some cases, project managers would directly communicate with submitters about contributions in order to solicit or negotiate changes. In some cases, multiple rounds of comments were necessary in order to establish shared understanding. The transparent nature of GitHub also led developers to become acutely aware that their work actions had an audience. These audience pressures encouraged developers to create more legible contributions with “snappy” commit messages and clean code because users would potentially later use the commit history to make inferences such as developer intention and competence.

Marlow et al. [19] found that when GitHub developers engage in information-seeking behaviors, they use signals in the environment to form impressions of users and projects. For example, impressions of general coding ability could be gleamed from the contents of a GitHub user's profile. Signals of whether or not a developer possesses specialized project-relevant skills were embedded in the user's activity log. Project managers would often account for uncertainty when evaluating contributions, straightforward and easily verifiable changes were often accepted "as is" whereas complicated, uncertain changes would require discussion before acceptance. In these cases, project managers would often engage in discussion with the submitter in order to negotiate the change. In these cases, where the value of the contribution was uncertain, project managers would make use of both code-based factors and person-based factors. For example, a project manager may weigh the cost of fixing a contribution against the benefit of recruiting a new member to the project.

Pham et al. [23] found in their study of the testing culture in GitHub that project managers would demand that contributions include tests in certain cases. For example, contributions that introduced new features were expected to include tests. On the other hand, contributions that involved existing code, especially if the change was small like a bug fix, may or may not require tests. Also, if the project manager trusted the submitting developer, the contribution tended to be evaluated more leniently. Project managers also saw an urgent need for automatic testing in their projects due to the large group of peripheral developers and general issues of scale. Many submitters would include tests as a method for highlighting the value of their contribution to the project manager. The transparent nature of GitHub helped lower the barrier for including tests in contributions. For example, if tests for existing parts of the codebase were prominent and visible, submitters would be much more likely to make use of these existing tests and include tests in their own contributions.

2.4 Research Questions and Hypotheses

Our examination of literature on contributions in open source software and online communities suggests a number of potentially important factors in evaluating contributions in transparent work environments such as GitHub.

2.4.1 Technical Contribution Norms

We see in prior work about GitHub that there are certain contribution norms that signal a technically well-prepared contribution. For example, project managers see an urgent need for automatic testing in their projects in order to maintain quality as the number of peripheral developers scales [23]. So, project managers tend to value contributions that include test cases more highly. Another example of such a signal is the community norm of having legible, easy-to-evaluate pull requests [5]. Contributions that display these signals of technical value may indicate a well-thought out technical submission that is also easier for a project manager to evaluate [5].

H1: Contributions that show signs of following technical contribution norms are more likely to be accepted.

2.4.2 Social Connection

In traditional open source software projects, newcomers often need to “recruit” core members of a project in order to have their contributions accepted [8]. This process involves knowing who the key core members are and being able to convince them of the usefulness of the contribution, especially if the code contribution is complex. Often, these key members expect newcomers to have previously participated in technical discussions and other peripheral actions in order to learn project-specific norms and prove suitability before submitting contributions [17]. In GitHub, these kinds of social connections are visible and made explicit, perhaps making social connections between submitters and project managers more salient.

H2: Contributions from submitters with a stronger social connection to the project are more likely to be accepted.

2.4.3 Highly Discussed Contributions

Certain contributions raise uncertainty about their value for a project and subsequently generate more discussion [19]. Changes that required high amounts of discussion tend to be more closely scrutinized by more members of the site, as GitHub users would look at discussion on a contribution as a signal of controversy. These contributions may be less technically sound, more complicated to evaluate, or simply controversial in terms of project direction or implementation strategy. Due to the high degree of uncertainty, project managers may then be less willing to accept the contribution.

H3: Contributions with a high amount of discussion are less likely to be accepted.

2.4.4 Decision-Making for Highly Discussed Contributions

When the value of a contribution is uncertain, project managers may employ different standards when evaluating the contributions [19]. In the cases of contributions with high amounts of discussion, we expect both the tone of the discussion and the degree of uncertainty to change depending on differences between the technical nature of the contribution and the social relationship between the submitter and the core project team. These different social and technical factors should then moderate the uncertainty in highly discussed contributions.

H4: Acceptance of highly discussed contributions will be moderated by both social and technical factors.

2.4.5 Submitter's General Community Standing

Previous research on GitHub has found that developers often use inferences about developers and software projects to evaluate them [19]. This research suggests the identity of the submitter and/or the software project may affect how contributions are evaluated. Members of the GitHub community regard certain members as being at a higher standing. Some prolific developers are even considered "coding rockstars" by the overall community [5]. Project managers who receive contributions from higher standing submitters may then be more willing to accept them based on the submitter's status.

H5: Contributions from submitters with a high status in the general community are more likely to be accepted.

2.4.6 Submitter's Status in Project

With open source software projects, there often is a structure of "core" and "periphery" developers, with core developers being the few central developers who implement most of the code changes and make important project direction decisions and peripheral developers being the "many eyes" of the project that make small changes such as bug fixes [20]. Core developers who make contributions to their own project may then be more likely to have their contributions accepted by fellow project managers.

H6: Contributions from submitters that hold higher status in a specific project are more likely to be accepted.

2.4.7 Project Establishment

As open source software projects progress through their lifecycle, their needs tend to differ from less mature projects [22]. The development stage of an open source project also tends to moderate its performance outcomes [28]. As projects evolve, their contribution needs may also co-evolve [22]. More established projects may be more service-oriented with many downstream dependencies. Project managers are often aware that their projects are depended on by other, perhaps more high profile projects. For example, certain popular websites may depend on a particular library on GitHub, so a broken release may also break the popular website [5]. Project managers of established projects may then be much more conservative when accepting contributions in light of these dependencies.

H7: Contributions to established projects are less likely to be accepted.

3. METHODS

To answer our research questions, we created and analyzed a dataset from the social open source software hosting site GitHub [12]. We selected a sample of pull requests on GitHub and gathered information on the pull requests, the submitting users, and the project the pull request was submitted to. From this dataset, we fit a statistical model that associates social and technical contribution measures with the likelihood of pull request acceptance. In this section we present descriptions of the GitHub setting, our data collection procedures, measure calculation, and analysis technique.

3.1 Description of the Research Setting

GitHub is a project-hosting site started in 2008 that brands itself as "Social Coding." The site offers both free open source project hosting and paid private hosting and is home to over ten million repositories [1]. Some of the more popular open source software projects that GitHub hosts include *Ruby on Rails* and *jQuery*. We selected GitHub as our research setting because it implements many of the social networking features found in well-known social networking sites such as Facebook and Twitter to improve collaboration between software developers through transparency.

GitHub allows potential project contributors to "fork" or make a personal copy of any public project where they can make changes to, add, or alter functionality, without disturbing the code in the original branch. This potential participant can then request that code changes in their personal copy be merged into the project's main repository. This can be accomplished by creating a "pull request" to the original project. The project manager has several options to "close" the pull request, including accepting the offered contribution and merging it into the project's code base or rejecting the contribution. At the same time, managers and other interested users may comment on the pull request, perhaps to suggest improvements or negotiate over the code change. Of course, project managers may also ignore the contribution, leaving the pull request "open".

GitHub also provides a set of social networking features. These include the ability for developers to "follow" other members in the community and to "star" the repositories of different projects. Following directs events about actions by a developer to the participant's news feed. Examples of such events include the creation of a new project or the starring of an existing project by a followed participant. Much of the followed participants' social activity is also visible in the feed, including changes to the set of users that person is following. Starring a repository works similarly to a bookmarking system, adding the starred project to a list of projects for a particular user. In addition, participants have a profile page that lists personal information as well as activity-related information such as the repositories they own and watch as well as the participants that they follow.

3.2 Pull Request Selection

We create a dataset of pull requests and the users and repositories associated with each pull request through sampling for active, collaborative projects on GitHub. Our dataset comprises information gathered from the GitHub Application Programmer Interface (API). First, we drew a sample of repositories from the GitHub Archive dataset [11] on July 17, 2013 with the following sampling criteria:

1) Excluded forks, developer-specific copies of repositories often meant for interim development work, in order to avoid double-counting contributions in our model.

Table 1. Descriptives of measures pre-transformation.

Measure	mean	median	stdev	skew
Test Inclusion*	0.151	0.000	0.358	1.950
Commit Size (lines)	1456	25.000	27799	61.876
Files Changed	13.265	2.000	165.460	67.691
Social Distance*	0.096	0.000	0.295	2.740
Prior Interaction	200.583	22.000	566.388	8.184
Comments	2.664	1.000	6.656	19.198
Followers	35.972	7.000	177.082	22.965
Collaborator Status*	0.435	0.000	0.496	0.261
Repo Maturity**	2.104	1.956	1.188	0.568
Collaborators	20.203	8.000	42.808	6.063
Stars	1981	293	4095	2.977
Pull Req Acceptance*	0.723	1.000	0.447	-0.999

*Dichotomous variables
**In years as of July 17, 2013

2) Excluded repositories that have not had at least one event of activity within one week prior to data collection, July 10, 2013 in order to avoid inactive projects.

3) Excluded repositories that do not use the GitHub issue tracker, as we also use the issue tracker as a source of data.

This selection included 185,342 repositories. We further refined the selection using the GitHub API to retrieve more detailed information about each repository with the following criteria:

1) Removed each repository that did not contain at least one closed pull request due to using closed pull requests as a base unit of analysis.

2) Excluded repositories with less than three unique contributors in order to ensure that the project has received some outside contributions.

After this second phase of filtering, our sample included12,482 projects.

We used pull requests as a base unit of analysis. From these 12,482 projects, we extract all closed pull requests from the API. As we were interested in the decision of whether or not to accept a pull request, we excluded all open pull requests. In total, this includes 659,501 pull requests across the 12,482 projects. For this dataset, we also gathered information about each unique GitHub user associated with the set of pull requests. This set of user information includes 95,270 unique GitHub user accounts. We also used the API to gather information on all issues and comments for each repository.

3.3 Measures

From our created dataset, we generated contribution measures for our analysis based on prior literature on GitHub, traditional open source software communities, and online communities (see Table 1 for a descriptive summary of the measures).

3.3.1 Outcome Measure

Our main outcome measure was whether or not a pull request is accepted. Pull request acceptance in this context means that the code contributions included in the pull request were merged into the project's code base. Pull request acceptance is a dichotomous variable.

3.3.2 Pull Request-level Measures

For our base level of measurement, we collected information unique to each closed pull request in our dataset. Each signal for the pull request represents a social or technical attribute about the contribution that may factor into the acceptance decision.

3.3.2.1 Technical Contribution Norms

We use three measures to operationalize different dimensions of valued technical contribution norms for a pull request.

Test Inclusion – This measure was a dichotomous variable indicating whether or not the pull request included test cases. From the prior work of Dabbish et al. [5] on GitHub, we know that when core members evaluate pull requests, they look for the inclusion of test cases as a signal of the thoroughness of the contribution. To measure this, we looked at the file pathnames in each pull request and looked for the word "test". If the pull request included such a pathname, then the pull request is labeled as including tests. This is due to most test cases either residing in a test folder (i.e. project/test/…) or the filenames including the word "test" (i.e. test_numberformat.java). To verify, a simple spot-check was performed on forty randomly chosen pull requests, twenty labeled as having tests, twenty labeled as not having tests. All checked pull requests were found to be correctly labeled. Of course, this measure is probably conservative, with unfound false negatives.

Commit Size – This measure is the number of lines changed in the pull request. Along with number of files changed, we included the number of lines changed in a pull request as a signal of a pull request's legibility. Pull requests that change large portions of the code base at a time are much harder for project managers to understand and evaluate.

Number of Files Changed – This measure is the number of files changed in the pull request. Along with the commit size, we use these measures to indicate how legible a particular pull request is. Pull requests that touch a large number of files tend to be much harder to understand and evaluate for project managers [19].

3.3.2.2 Social Connection

To represent two different dimensions of the social connections in GitHub, we used a measure for social distance and another for prior interaction.

Social Distance – This measure was a dichotomous variable indicating whether or not the submitter follows the user that closes the pull request. We use this as a proxy of the social closeness between the submitter and the closer in a particular pull request.

Prior Interaction – Prior work on GitHub by Dabbish et al. [5], indicates that core members for a project, especially when attempting to recruit new members, use prior contributions as a signal of the trustworthiness of a contributor and contribution. To measure prior interaction, we counted the number of events before a particular pull request that the user has participated in for this project. Events include participating in issues, pull requests, and commenting on various GitHub artifacts.

3.3.2.3 Highly Discussed Contributions

Comments on Pull Request – Marlow et al. [19] found that uncertain pull requests tended to require negotiation and/or explanation. Pull requests with lots of comments also tended to signal controversy [5]. To measure the level of discussion, we counted the number of comments in the closed pull request.

3.3.3 User-level Measures

As each pull request has a submitting user that may submit multiple pull requests to a project, we grouped pull requests by the submitting GitHub user account and collected information about each GitHub submitter.

3.3.3.1 Submitter's General Community Standing

Followers – This measure is the number of followers a GitHub user has at time of data collection. The number of followers a GitHub user possesses is used as a signal of standing [5] within the community. For example, users with lots of followers were treated as local celebrities.

3.3.3.2 Submitter's Status in Project

Collaborator Status – This signal is a dichotomous variable for the user's collaborator status within the project. In GitHub, a collaborator for a project has direct commit access to the repository. Therefore, they do not need to perform the pull request process in order to merge code contributions into the project. However, interviews with GitHub users indicate that many collaborators opt to create pull requests for code contributions despite having commit status. Often, this is done to allow other users to review changes before accepting the code contribution.

3.3.4 Repository-level Measures

We further grouped the dataset by grouping each set of submitters into a repository and collected information about each repository.

3.3.4.1 Project Establishment

We used three different measures to represent three dimensions of establishment for the project receiving the pull request is.

Repository Age – This measure is a continuous variable representing the project's age how long a project has existed on GitHub since the time of data collection. We use this as an indicator of the repository's maturity.

Collaborators – This measure is the number of collaborators on a project. We use the number of collaborators as a proxy for the relative size of the development team involved in a particular GitHub project.

Stars – This measure is a continuous variable for the number of stars on a project. When evaluating projects, GitHub users make use of the number of stars as a signal for community interest in the project [5]. As stars were indications of attention from a user to a particular project, more stars indicate more users interested in the project. Measures such as the number of forks and the number of contributors to a particular GitHub project were highly correlated with this measure, so were omitted to avoid collinearity.

4. RESULTS

Our analysis suggests that both technical and social contribution measures are highly associated with acceptance. First, we examine our hypotheses and how each predictor variable associates with acceptance. We also consider factors that cut across pull requests such as user-level and repository-level measures. A summary of the models is presented in Table 2.

We report measure associations with contribution in odds ratios, which are the increase or decrease of the odds of acceptance occurring per "unit" of the measure. In this case, a "unit" of each measure is one standard deviation from the log-transformed for continuous variables or the presence of a dichotomous variable.

4.1 Analysis

Using these pull request-level, submitter-level, and repository-level measures, we create a model that predicts the likelihood of pull request acceptance. We fit a multi-level mixed effects logistic regression model to our data because our outcome variable (acceptance) is dichotomous and our dataset nested in multiple levels. We chose a logistic regression approach in order to better predict our dichotomous outcome variable. To account for the three-level nesting of the dataset from pull requests to users to repositories, we created a mixed model where our contribution measures are fixed effects and the unique user and repository intercepts are represented as random effects. We used a R [24] package [2] that accounts for cross-classification of data, as 28,880 out of 95,720 users appear in multiple projects in our dataset. None of the measures had pairwise correlations above 0.6 suggesting no multicollinearity problems [7]. To ensure normality, each of the continuous variables in the model was log transformed and then centered such that the mean of each measure is 0 and standard deviation is 1.

4.2 Pull Request-Level Measures

4.2.1 Technical Contribution Norms

H1: Contributions that follow technical contribution norms are more likely to be accepted.

We tested H1 by examining the association of test case inclusion, commit size, and files changed with contribution acceptance. The inclusion of test cases was positively associated with pull request acceptance, with acceptance likelihood increased by 17.1% when tests are included. Lines changed had a stronger effect but negative, with each unit of lines changed decreasing the chance of acceptance by 26.2% compared to 7.3% with each unit of files changed.

As we expect contributions that include test cases and are more legible are more likely to be accepted, so we find support for H1.

4.2.2 Social Connection

H2: Contributions from submitters with a stronger social connection to the project are more likely to be accepted.

We tested H2 by examining the association of social distance and prior interaction with contribution acceptance. We find support for H2 as both of our social connection measures were positively associated with pull request acceptance. Our measure of social distance had the strongest influence on likelihood of acceptance as compared with other pull-request level factors, increasing acceptance by 187% when the submitter follows the project manager. Prior interaction was also positively associated with acceptance, increasing acceptance likelihood by 35.6% per unit.

4.2.3 Highly Discussed Contributions

H3: Contributions with a high amount of discussion are less likely to be accepted.

To test H3, we examined the association between pull request comment count and acceptance. Pull requests with longer discussion, as indicated by higher counts of comments, were less

Table 2. Multi-level mixed effects logistic model for pull request acceptance

			Model I	Model II	Model III	Model IV
			Pull Request Level		Pull + Submitter Level	Pull + Submitter + Repo Level
	Factor	**Variable**	**Odds Ratio**	**Odds Ratio**	**Odds Ratio**	**Odds Ratio**
Pull Request Level		(Intercept)	2.934 ***	2.898 ***	2.845 ***	3.925 ***
	Technical Contribution Norms (H1)	Test Inclusion	1.059 ***	1.023 *	1.114 ***	1.171 ***
		Commit Size	0.849 ***	0.834 ***	0.736 ***	0.738 ***
		Number of Files Changed	1.165 ***	1.152 ***	0.970 ***	0.927 ***
	Social Connection (H2)	Social Distance	1.345 ***	1.461 ***	3.636 ***	2.870 ***
		Prior Interaction	1.423 ***	1.362 ***	1.207 ***	1.356 ***
	Highly Discussed Contributions (H3)	Comments	0.481 ***	0.480 ***	0.414 ***	0.454 ***
	Decision-Making for Highly Discussed Contributions (H4)	*Test Inclusion x Comments*		1.057 ***	1.092 ***	1.106 ***
		Commit Size x Comments		1.101 ***	1.166 ***	1.169 ***
		Files Changed x Comments		1.017 ***	1.043 ***	1.035 ***
		Social Distance x Comments		0.806 ***	0.792 ***	0.796 ***
		Prior Interaction x Comments		1.106 ***	1.246 ***	1.142 ***
Submitter Level	Status in General Community (H5)	Followers			1.060 ***	1.181 ***
	Status in Project (H6)	Collaborator Status			3.904 ***	1.636 ***
Repo Level	Project Establishment (H7)	Repository Age				0.820 ***
		Collaborators				0.954 **
		Stars				0.648 ***
AIC:			633600	630879	506850	461077

likely to be accepted, supporting H3. This is our second strongest effect among the pull request-level factors, with the likelihood of acceptance decreasing by 54.6% with each unit of comment count.

4.2.4 Decision-Making for Highly Discussed Contributions

H4: Acceptance of highly discussed contributions will be moderated by both social and technical factors.

To test H4, we added an interaction term to the model, interacting number of comments with each pull request-level measure in order to investigate how social and technical factors moderated the decision-making process for highly discussed contributions. We found that all five interactions with social and technical factors were significant, indicating support for H4.

We provide charts detailing the direction of the interactions in Figures 1, 2, and 3. The associations of test inclusion, number of files, commit size, and social distance all significantly moderate the influence of discussion on contribution acceptance, though with a small effect. Prior interaction most strongly moderates the relationship between discussion and acceptance, with number of comments having almost no influence on acceptance for previous contributors. We discuss later the implications of these interactions for how evaluating highly discussed contributions may differ from more standard contributions.

4.3 User-Level Measures

4.3.1 Submitter Status in General Community

H5: Contributions from submitters with a high status in the general community are more likely to be accepted.

To test H5, we examined the association of follow count with pull request acceptance. We find a positive association, supporting H5. Having followers increases the likelihood of acceptance by 18.1% per unit of followers. This suggests that submitters with higher community standing are more likely to have their pull requests accepted.

4.3.2 Submitter Status in Project

H6: Contributions from submitters that hold higher status in a specific project are more likely to be accepted.

We tested H6 by examining the association of collaborator status with contribution acceptance. Perhaps unsurprisingly, when submitters with commit access choose to create pull requests instead of directly merging code, their pull requests are more likely to be accepted than non-collaborators, supporting H6.

Being a collaborator on a project increases the likelihood of contributions being accepted by 63.6%.

4.4 Repository-Level Measures

4.4.1 Project Establishment

H7: Contributions to established projects are less likely to be accepted.

We test H7 by examining the association of our project establishment measures (the age of the project, number of users with commit status, and popularity of the project) with contribution acceptance. All three of our project establishment dimensions have negative associations with pull request acceptance, so we find support for H7. Number of collaborators, used as a proxy for project team size, has the smallest influence on acceptance likelihood out of the three establishment measures, decreasing acceptance by 4.6% per unit of collaborator count. Somewhat surprisingly, this suggests that project "size" does not have as strong an influence on pull request acceptance as compared with age or popularity. The older a project, used here as a proxy for maturity, the less likely it is to accept pull requests, with acceptance decreasing by 18.0% per unit of project age. Popularity had the strongest negative influence on acceptance, with projects 35.2% less likely to accept pull requests per unit of increase in stars.

5. DISCUSSION

In this section, we summarize the results and discuss the implications of the hypotheses in terms of prior literature.

5.1 Technical Norms and Social Connection

From conventional wisdom on open source software projects, we expect to see some evidence for a "meritocracy", in that technical contribution norms should reign over other signals when considering contributions [26]. However, in the environment of GitHub, which is both transparent and equipped with social media functionality, we also expect contributors to make use of the social connections that the environment makes salient. Our analysis suggests that while following technical contribution norms for pull requests is associated with acceptance, the social connections behind pull requests have even stronger associations.

In terms of technical contribution norms, we found that pull requests more consistent with community-wide pull request practices like inclusion of test cases and small commit sizes [5] were more likely to be accepted. Code contributions that did not follow technical norms were less likely to be accepted, perhaps due to the higher assessment costs required by the project manager.

We also find that social connections increase likelihood of contribution acceptance, even when controlling for compliance with technical contribution norms. In traditional open source software projects, contributors are often expected to participate in more social aspects of the project such as participating in mailing list technical discussions before making code contributions in order to learn project-specific norms and ease socialization [17]. In the case of GitHub however, this expectation may be less prevalent because the pull request system standardizes the contribution process. The pull-request process also lowers the barriers for contribution, meaning many developers will make one-off contributions to projects or "drive-by commits" [23]. However, we still found that a contributor that has prior interaction with a project also has a higher likelihood of pull request acceptance. We also find that submitters socially closer to project managers tend to have their contributions accepted. This social distance association is also the strongest in the model. Similar to evaluating technical contribution norms, stronger social connections may indicate qualities such as trust, which may lower the project manager's assessment cost. For example, if the submitter is trusted to make good contributions, project managers may be more lenient in their evaluations [23].

While both technical contribution norms and social connections were associated with pull request acceptance, our measures for social contribution had much stronger associations than our technical contribution norm measures. One possible explanation is that when project managers are evaluating pull requests, when the evaluation cost is too high, they may decide to outright reject the contribution. Whereas pull requests that follow technical norms such as legible code changes and test cases make the pull request

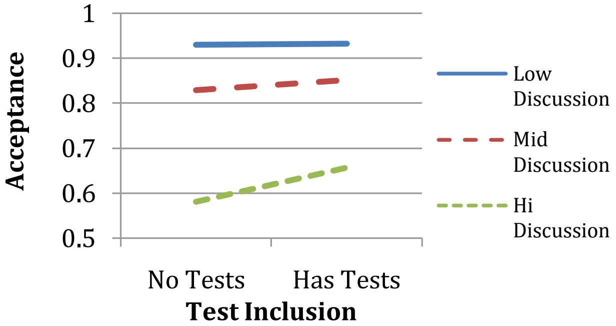

Figure 1. Interaction plot of test inclusion and contribution contention

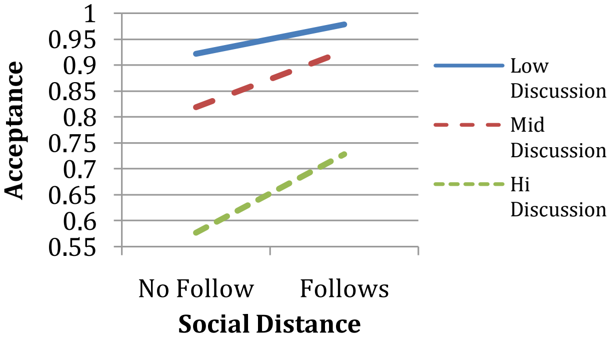

Figure 2. Interaction plot of social distance and contribution contention

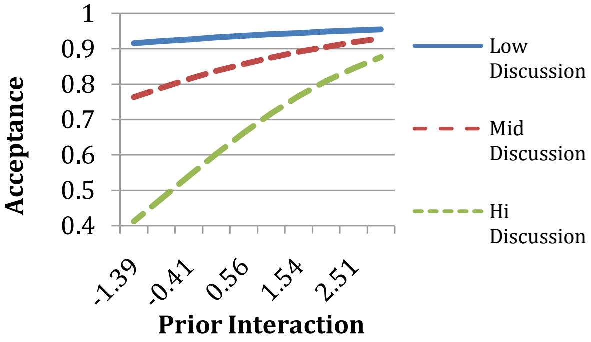

Figure 3. Interaction plot of prior interaction and contribution contention (prior interaction values are standardized)

much easier to evaluate, a strong social connection between the project manager and submitter may allow the project manager to bypass much of the evaluation process. Pull requests from unknown developers may be subject to much more thorough and costly evaluations from project managers than pull requests from known contributors [23]. For example, a familiar developer may be expected to already have run the contribution through the test suite, allowing for a project manager to bypass that phase of the evaluation, increasing the likelihood of acceptance. Similarly, we see that members of the project with commit rights, perhaps an explicit form of trust, also have positive associations with acceptance. This may be similar to the effect of familiarity in distributed software development, where team familiarity is associated with team performance, especially for geographically dispersed teams (as many GitHub projects are) [9]. One explanation for the familiarity finding is that teammates with high familiarity know whom to contact for queries and resources, making coordination much more efficient. Perhaps submitters with a strong social connection also lower the coordination costs required to use the contribution. For example, project managers familiar with the submitter may not bother to look for project-specific coding style norms, knowing that the submitter already should know them. All of these factors that lower assessment costs reduce the chances that flaws are found, or that the project manager will opt to reject the pull request rather than perform and expensive evaluation.

Future research should examine in more detail how technical and social signals influence evaluation cost during pull request acceptance. If technical norm signals are harder to evaluate, then perhaps future collaborative software tools should focus on lowering the evaluation cost of a software contribution. At the same time, if developers are using social signals to evaluate contributions, then perhaps those signals should be made more visible during evaluation tasks, assuming that these signals are optimal for decision-making. Future research should also examine whether these evaluation decisions are optimal or what leads to optimal acceptance decisions.

5.2 Decision-Making and Highly Discussed Contributions

Next to social distance, the amount of discussion around a pull request had the strongest influence on likelihood of acceptance. The more highly discussed the contribution, the less likely the contribution would be accepted. This by itself is not too surprising, given the high degree of uncertainty present in such pull requests [19]. However, we also hypothesized that highly discussed pull requests differ in both the tone of the discussion and the degree of uncertainty in the contribution being discussed. These differences in the nature of the discussion around evaluating a pull request also reflect differences in the decision-making process for project managers. When discussing a pull request in order to evaluate the value of the contribution, project managers may be using different kinds of information. We found in our model (see Table 2) that both technical contribution norms and social connection measures moderated the effect of discussion on contributions.

For highly discussed contributions, our social and technical pull request-level measures moderate the negative association of discussion amount on acceptance. For most of our factors, however, regardless of being social or technical in nature, the moderating effect is too small to affect the very negative influence of having a large amount of discussion in a contribution. In Figure 1 for example, the negative effect of high discussion overwhelms the positive technical effect of test inclusion, reducing the likelihood of acceptance by about 30% regardless of test inclusion. Even for the variable with the largest association with acceptance, social distance, having a high amount of discussion still reduces the likelihood of acceptance by about 25% regardless of whether or not following occurs as seen in Figure 2. This small moderating effect suggests that for most pull requests, project managers are much less willing to accept the contribution, regardless of whether or not technical contribution norms are followed or a social connection of the submitter to the project manager exists. This may indicate that regardless of the tone of the discussion or nature of the contribution, high amounts of discussion on a pull request indicates a high degree of uncertainty for the value of the contribution.

However, a submitter's prior interaction on the project significantly changes the influence of discussion on acceptance as seen in Figure 3. Surprisingly, there is a positive association between discussion and acceptance likelihood for participants with prior interaction. This may indicate that when experienced submitters are working on a project, the nature of the discussions around their pull requests is different in some way than submitters who do not have this prior experience. Discussions where the submitter has high amounts of prior interaction may be less focused on evaluating a contribution's value and more focused on optimizing the code. Conversely, the discussion around a contribution from a submitter with no prior interaction on the project may focus more on evaluating whether the pull contribution is worth accepting. For example, a submitter with no prior interaction may be unaware of a project's submission practices and the resulting discussion would be focused on ensuring the pull request matches the project's standards.

Interestingly, the moderation effect of prior interaction is at odds with the effect of social distance despite both variables being used for our social connection measure in the analysis model. This may suggest that when discussing contributions, project managers will turn to prior interaction rather than social distance as a signal to use during evaluation of pull requests. Perhaps this occurs because prior interactions are a more trustworthy signal than the social distance signal of the submitter following the project manager. To demonstrate prior interaction, a user has to actively participate in discussions, bug reports, and other forms of contribution on the project. Prior interaction may act as an assessment signal, where the signal of prior interactions cannot easily be generated without actual participation [6]. Prior interaction is a reliable signal of social connection because participation cannot be easily faked. On the other hand, social distance via following may indicate a social connection between two users through convention. This signal is less reliable because a submitter can follow a project manager without actually creating a social connection with the project manager. When discussing how to evaluate contributions, the convention of following users does not replace familiarity built from actual prior interaction.

Future research should examine how we can design tools that assist in deliberation by highlighting certain information. Future tool design may assist developers during software change evaluation discussions by making certain signals more or less visible. Future tools may even dynamically change the visibility of different signals depending on the tone of the discussion.

5.3 Audience Pressures

While social and technical features of pull requests had important associations with acceptance, our model also suggests that the type of submitter and the type of project that the pull request is submitted to also influences acceptance likelihood.

Pull requests from submitters who have commit rights, known as collaborators in GitHub, were associated with acceptance. Pull requests from collaborators seem to be special cases of contribution because these users are not required to undergo the pull request process in order to have their changes merged into the project, unlike other developers.

Well-established projects were negatively associated with acceptance on all three dimensions. In particular, the popularity of a repository has the strongest negative association out of the three. Number of stars, our proxy for project popularity, is used by members of the GitHub community as a signal for project quality, which project managers are aware of [5].

The contrasting associations between popular projects and collaborators may indicate that audience pressure is a factor when project managers evaluate pull requests. For popular projects, the transparent nature of GitHub means project managers are aware, at least in part, of the identity of users of their project [5]. Knowing that hundreds or thousands of users, some highly visible, depend on a particular project may discourage project managers from accepting risky or uncertain code contributions. Conversely, collaborators, who possess the ability to accept pull requests into the project, may be immune to these audience pressures.

The effect of audience pressure on software contribution evaluation is not well understood. Future research may investigate more thoroughly how audience pressures affect both contributors and core members of projects. Signals used to evaluate contributions may differ depending on whether or not core members feel pressure from the audience. For example, core members may be much more concerned about managing uncertainty when they are aware that millions of potential users are watching and depending on the project being stable.

5.4 Limitations

One of the main limitations of our study is that most of our data is of a cross-sectional nature. At the same time, some of our measures are more robust to reverse-causality because of timing inherent in the pull request process. Prior interaction, test inclusion, and number of lines and files changed, are all variables whose value is determined prior to any consideration of acceptance of the pull request. Other variables, however, are cross-sectional at the time of data collection, such as follower count. Without performing a true longitudinal analysis, we cannot be certain about the direction of causality for these latter variables using our dataset. Future work should perform longitudinal analyses on contribution measures in order to make stronger inferences about causality.

6. CONCLUSION

In this work we examined how social and technical information in the transparent open source software environment of GitHub is used to make contribution decisions. We created a statistical model analyzing the association of different pull request, submitter, and repository measures of contributions with the likelihood of the contribution being accepted. We found that project managers made use of information involving both the technical contribution practices of a pull request and the strength of the social connection between the submitter and project manager when evaluating pull requests. Highly discussed pull requests were much less likely to be accepted, however, the submitter's prior interaction in the project moderated this effect. Well-established projects were more conservative when evaluating pull requests, perhaps due to audience pressures. Our findings inform how software developers and project managers make use of information in social work environments such as GitHub and imply a variety of ways that social features in work environments can support software development. Future research may investigate how developers use signals in other work environments, transparent or not. Future tool design may use our findings to identify signals to make more visible for project managers when making evaluation decisions. Our findings may also inform how project managers should change their evaluation policies based on what signals are important.

7. ACKNOWLEDGMENTS

This material is supported by the Center for the Future of Work at Carnegie Mellon University's Heinz College and by the National Science Foundation under Grant No. IIS1111750.

8. REFERENCES

[1] 10 Million Repositories - GitHub: *https://github.com/blog/1724-10-million-repositories*.

[2] Bates, D., Maechler, M. and Bolker, B. 2013. lme4.0: Linear mixed-effects models using S4 classes.

[3] Bryant, S.L., Forte, A. and Bruckman, A. 2005. Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia. *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work* (New York, NY, USA, 2005), 1–10.

[4] Crowston, K., Wei, K., Howison, J. and Wiggins, A. 2008. Free/Libre open-source software development: What we know and what we do not know. *ACM Comput. Surv.* 44, 2 (Mar. 2008), 7:1–7:35.

[5] Dabbish, L., Stuart, C., Tsay, J. and Herbsleb, J. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (New York, NY, USA, 2012), 1277–1286.

[6] Donath, J. 2005. *Signals, truth, and design*. MIT Press.

[7] Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., Marquéz, J.R.G., Gruber, B., Lafourcade, B., Leitão, P.J. and others 2013. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*. 36, 1 (2013), 27–46.

[8] Ducheneaut, N. 2005. Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work (CSCW)*. 14, 4 (2005), 323–368.

[9] Espinosa, J.A., Slaughter, S.A., Kraut, R.E. and Herbsleb, J.D. 2007. Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development. *Organization Science*. 18, 4 (2007), 613–630.

[10] Foundation/Charter - GNOME Wiki: *https://wiki.gnome.org/Foundation/Charter*. Accessed: 2013-08-31.

[11] GitHub Archive: *http://www.githubarchive.org/*. Accessed: 2013-08-31.

[12] GitHub: *http://github.com*. Accessed: 2013-08-21.

[13] Hars, A. and Ou, S. 2002. Working for Free? Motivations for Participating in Open-Source Projects. *Int. J. Electron. Commerce*. 6, 3 (Apr. 2002), 25–39.

[14] How the ASF works: *http://www.apache.org/foundation/how-it-works.html*. Accessed: 2013-08-31.

[15] Iriberri, A. and Leroy, G. 2009. A life-cycle perspective on online community success. *ACM Comput. Surv.* 41, 2 (Feb. 2009), 11:1–11:29.

[16] Kraut, R.E. and Resnick, P. 2012. *Building Successful Online Communities: Evidence-Based Social Design*. MIT Press.

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

[18] Lakhani, K.R. and Wolf, R. 2005. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *Perspectives on Free and Open Source Software*. J. Feller, B. Fitzgerald, S. Hissam, and K.R. Lakhani, eds. MIT Press. 3–22.

[19] Marlow, J., Dabbish, L. and Herbsleb, J. 2013. Impression formation in online peer production: activity traces and personal profiles in github. *Proceedings of the 2013 conference on Computer supported cooperative work* (New York, NY, USA, 2013), 117–128.

[20] Mockus, A., Fielding, R.T. and Herbsleb, J.D. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.* 11, 3 (Jul. 2002), 309–346.

[21] Mozilla - Governance - mozilla.org: *http://www.mozilla.org/en-US/about/governance/*. Accessed: 2013-08-31.

[22] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. and Ye, Y. 2002. Evolution patterns of open-source software systems and communities. *Proceedings of the International Workshop on Principles of Software Evolution* (New York, NY, USA, 2002), 76–85.

[23] Pham, R., Singer, L., Liskin, O., Figueira Filho, F. and Schneider, K. 2013. Creating a shared understanding of testing culture on a social coding site. *Proceedings of the 2013 International Conference on Software Engineering* (Piscataway, NJ, USA, 2013), 112–121.

[24] R Core Team 2013. R: A Language and Environment for Statistical Computing.

[25] Roberts, J.A., Hann, I.-H. and Slaughter, S.A. 2006. Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*. 52, 7 (2006), 984–999.

[26]Scacchi, W. 2007. Free/Open Source Software Development: Recent Research Results and Methods. *Architectural Issues*. M. V Zelkowitz, ed. Elsevier. 243–295.

[27] Shah, S.K. 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Manage. Sci.* 52, 7 (Jul. 2006), 1000–1014.

[28] Stewart, K.J. and Gosain, S. 2006. The moderating role of development stage in free/open source software project performance. *Software Process: Improvement and Practice*. 11, 2 (2006), 177–191.