Which Contributions Predict Whether Developers Are Accepted Into GitHub Teams

Justin Middleton, Emerson Murphy-Hill, Demetrius Green,
Adam Meade, Roger Mayer, David White, Steve McDonald
North Carolina State University
Raleigh, North Carolina
{jamiddl2,ermurph3,dkgreen,awmeade,rcmayer,dlwhite5,sjmcdona}@ncsu.edu

ABSTRACT

Open-source software (OSS) often evolves from volunteer contributions, so OSS development teams must cooperate with their communities to attract new developers. However, in view of the myriad ways that developers interact over platforms for OSS development, observers of these communities may have trouble discerning, and thus learning from, the successful patterns of developer-to-team interactions that lead to eventual team acceptance. In this work, we study project communities on GitHub to discover which forms of software contribution characterize developers who begin as development team outsiders and eventually join the team, in contrast to developers who remain team outsiders. From this, we identify and compare the forms of contribution, such as pull requests and several forms of discussion comments, that influence whether new developers join OSS teams, and we discuss the implications that these behavioral patterns have for the focus of designers and educators.

ACM Reference Format:
Justin Middleton, Emerson Murphy-Hill, Demetrius Green, Adam Meade, Roger Mayer, David White, Steve McDonald. 2018. Which Contributions Predict Whether Developers Are Accepted Into GitHub Teams. In *MSR '18: MSR '18: 15th International Conference on Mining Software Repositories , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3196398.3196429

1 INTRODUCTION

With the Internet, developers today can organize and enjoy software development groups that span the world [1]. This is, in part, due to the spread of Open Source Software (OSS) philosophies, which bolster access to and redistribution of source code and of experimental software [2]. Many significant products, such as the Linux kernel and Mozilla Firefox, have developed under this approach, and their presence and value are still growing [3].

Despite the promise of accessibility in OSS philosophy, many OSS development teams struggle to attract and retain developers, and this puts consumers in danger. For example, consider the Heartbleed bug in OpenSSL, an open-source library for cryptography and security protocols.[1] When researchers uncovered this critical problem in 2014, at least 24% of HTTPS-enabled websites were at risk of exploitation, inducing severe costs in time and finances to fix [4]. Yet the maintenance team consisted mostly of developers working in spare time, dependent on donations, and often unable to respond to all bug reports.[2] Steve Marquess, financial officer for OpenSSL, reflected in blog post on how this problem of an inadequate work force was common throughout the OSS community, concluding that the mystery was "not that few overworked volunteers missed this bug; the mystery is why it hasn't happened more often." But the solution to this problem is more complicated than simply accepting every willing volunteer. Despite the need for developers, these projects must also guarantee quality by preferring sufficiently experienced developers [5].

So how will OSS projects survive? Aside from providing a hired workforce and material resources, as with the Core Infrastructure Initiative for example,[3] some companies have built websites for collaborative software development, providing the channels for interested developers to interact. On many of these websites, teams of developers can build projects and regulate access to these projects among volunteering contributors; individual developers, on the other hand, can navigate projects and contribute in several ways. As of this writing, the most popular of these websites is GitHub, connecting over 20 million registered users and over 70 million software projects,[4] including those of national governments and numerous industries

Given the influence of this website, along with the significance of OSS projects to the general public and the need for new developers to join them, we hope to see whether the data we can derive from these websites can lead us to greater insight and impact on the wellbeing of OSS projects. Therefore, we explore a central question: *on platforms for collaborative software development, what forms of activity most distinguish the developer who joins a software team from those who remain on the outside*? We answered this by gathering data on thousands of GitHub interactions between individual developer accounts, their projects, and GitHub teams. We organized activity types into a few categories, which include visible signs of attention (e.g. following developers, starring projects), textual discussion in issues and pull requests, and code contributions. From two snapshots of GitHub over a year apart, we then model the data to describe which interactions are most characteristic of developers who transition from roles outside the teams to roles inside.

MSR '18, May 28–29, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5716-6/18/05…$15.00
https://doi.org/10.1145/3196398.3196429

[1]https://www.openssl.org/
[2]http://veridicalsystems.com/blog/of-money-responsibility-and-pride/
[3]https://www.coreinfrastructure.org/
[4]https://github.com/about

Our analyses characterize several general ways of interacting with other developers on GitHub and indicates whether each form of interaction constitutes a positive or negative influence on the joining probability. Although we do not suggest hard boundaries which determine joining behavior, we note that the number of pull requests to a team constitutes the strongest positive influence of whether a developer joins that team, but we also note several forms of comment with moderate positive influence and many traits of teams that determines how receptive they are to new team members. Our work affirms the ways that developer activity makes impressions on teams and underscores the influence of social behavior in collaborative environments. Because the nature of team acceptance is important to both individuals and communities, we also describe principles of interactions that can be employed by designers of websites for OSS communities or for CS educators.

2 RELATED WORK

We organize the literature relevant to our question into three general categories: that which describe how developers are organized into structures in OSS communities; that which theorize how developers move into these communities; and that which establish GitHub specifically as a representative for modern OSS research.

2.1 The Structures of OSS Teams

Many researchers illustrate the distribution of developers in terms of an “onion" or core-periphery model, illustrated as concentric circles in which centrality implies more responsibility with less population. For example, Nakakoji and colleagues describe developers within four OSS projects using this model: outermost in the model are the passive users and readers, and innermost are the core members and project leaders. This community structure is so critical to these researchers that they conclude that any technical contribution not only changes the software but “also redefines the role of those contributing members and thus changes the social dynamics of the OSS community" [1]. Mockus, Dinh-Trong, and others perform case studies on large OSS communities and suggest that the balance of these layers is necessary to the healthy development of projects [6, 7]. However, some researchers, such as Ducheneaut in his case study of Python, argue that the onion model alone does not sufficiently capture the way developers move between layers [8]. Furthermore, the communities themselves understand that most participation is temporary [9]; Robles and colleagues point out that not even core members are exempt from moving outward [10]. Therefore, because current project members might eventually leave, the project teams must then consider how to keep new developers moving inward.

Furthermore, researchers have probed what it means to be a member of a project team, and the exact qualities of membership vary between studies. Vasilescu and colleagues conducted an online survey asking about whom an OSS team comprises; over 70% of respondents simply include everyone who does anything in the repository or anyone who makes good suggestions [11]. On the other hand, Shah and colleagues notes that, for hobbyist contributors, teams signal membership by granting a contributor the “committer" status, meaning the ability to directly change artifacts without another developer's permission [9]. Elsewhere, researchers like de Souza, Crowston, and others frame core developers as those who frequently contribute important code [12, 13], and Joblin and colleagues clarify that official designations as members or nonmembers within the projects often (but not always) aligns with the above metrics and developer perceptions [14].

In this project, we adopt the general core-periphery structure as a given in order to understand GitHub's project roles while also accepting the inward movement of some contributors over time. Bifurcating the developer population allows us to distinguish the activity of those in inner layers and those in the outer. However, rather than argue and define the number and types of roles as some of these studies have done, we take advantage of the roles that our research setting has already chosen and implemented into the website. This standard set of roles underlies an important difference between our work and previous researchers: whereas many of the previous studies cover only a few communities at a time as in-depth case-studies and design roles for them, we take advantage of our website as an aggregation of standardized communities to study thousands at a time.

2.2 The Theories of OSS Joining

The most relevant research is that which examines the act of joining a project team. von Grogh and colleagues propose the concept of a joining script, or general sequence of activity which newcomers must follow if they want join the group [15]. Teams, they say, expect both certain kinds of activities from and certain levels of it from peripheral contributors if they wish to officially join a project. For example, people who join their studied population often begin by reporting and fixing bugs, while those who didn't just asked questions about the project.

Other researchers build on the concept of joining scripts by hypothesizing which qualities of developers and teams influence joining. Sinha and colleagues found that they could explain the joining of 73% of their sample of contributors in Eclipse projects based on at least one of three hypotheses: they made bug reports or fixes, contributed to other OSS projects, or had social connections to core developers [16]. Furthermore, Bird and colleagues determined that a user's tenure on a project, contribution history, and social status influenced joining within some of their studied projects [5]. Casalnuovo and colleagues reaffirmed that social connections to a team are important for joining, at least on GitHub [17], and elsewhere, Sheoran and colleagues find that a lot of contributors begin as someone simply watching the project [18]. On the other hand, Herraiz and colleagues understand that not all open-source developers fall under the volunteer definition; some developers are hired by companies to contribute to critical OSS libraries, and these developers often end up following different joining scripts compared to volunteers [19].

As seen in these studies, our research question itself is not new—characterizing likely joiners has interested researchers for years. Although some of these also seek to create linear models for determining the influence of certain metrics on joining behavior, many of these emphasize social links that exist between developers. If discrete contributions are considered, it is often to see what the first type of interaction was, not to see how frequently or consistently the developers were making those contributions. Thus, we fit into our niche at quantifying the forms of contributions. And, as stated before, we benefit from the standardized forms of contributions across a single website that help us capture many communities at once.

2.3 Using GitHub for OSS Research

Many researchers study the dynamics of OSS projects on GitHub specifically, given its transparent interface. Dabbish and colleagues outline what it means to be transparent, or socially translucent, in that GitHub provides visible cues and artifacts from which other users can make inferences about someone's interest, activities, and skills [20]. Marlow and colleagues expanded that research by deconstructing how users form impressions through transparent traces of other users' activities in the act of evaluating contributions [21]. Pham and colleagues, on the other hand, looked into how these social factors hindered or help the spreading of community norms, particularly those of testing, throughout a project [22]. The studies set the groundwork for connecting the core-periphery models of community structures with concrete activities on the GitHub interface, as laid out in this research. Many of these studies, however, seek to define qualitatively, through interviews and otherwise, how GitHub's interface shapes the interactions on it, not to quantify the nature of team development and joining therein.

3 METHODOLOGY

In this section, we outline our three research questions first, and then we describe how we choose our research setting and collect the metrics by which approached our questions. Lastly, we describe our method for trimming the data of noise, transforming it, and then interpreting it.

3.1 Research Questions

As the influence of OSS projects grows and more consumers depend on their outcomes, the necessity of keeping these projects well-maintained becomes all the more critical. But we do not know of, nor is there likely to be, an exact and universal process to join development teams; the requirements for joining a software team varies between project types and team culture [1]. Furthermore, the focus on interactions only ignores the qualities about individual developers or team culture that distinguishes them from other developers or teams. Just as we can ask questions about the interactions between two parties, we can ask the same about the qualities of each party.

Therefore, we pose three research questions:

- *RQ1: What types and qualities of project interactions distinguish between developers who join an OSS team and developers who remain outside?*
- *RQ2: What activities or qualities of a developer independent from a specific team influence whether the developer joins that team?*
- *RQ3: What activities or qualities of a team influence whether the team accepts more people in?*

We motivate RQ1 through our interpretation on the previous literature of project migration. As von Krogh and colleagues set forth in their concept of the joining script, some behaviors are more likely to culminate in the acceptance of a newcomer by the group [15]. This script is not a checklist but a pattern of behavior that invested developers fulfill, often without any precisely definable thresholds. Building from this research, we assume the existence of patterns of behavior that characterize the relationship between an individual developer and an existing OSS development team and that predict, to some extent, the future of those relationships.

Furthermore, research on collaborative development websites motivates us to include RQ2 and RQ3 by considering activity beyond of this developer-team relationship. Teams on collaborative websites can often examine an individual's history with personal projects and with other teams on the individual's personal profile. Research suggests that this transparency into a developer's historical activity provides important insight into the individual's skills and interests [20, 21], which influences how the team evaluates their contributions. As such, we ask whether that influences the developer-team relationship, even when the developer's activity does not directly impact the team—for example, whether a developer's discussion habits in one project influences whether they join an entirely different project. From the other side, the activity of the team may likewise influence the likelihood that they accept a user.

3.2 Data Selection and Collection

To gather data, we looked among popular source code management (SCM) websites with team support, which includes GitHub, GitLab, Bitbucket, SourceForge, Assembla, and Google Code. At the time of writing, GitHub is the most popular of these websites, hosting everything from small projects by individuals to enormous, industry-effort OSS products. Furthermore, much of the website's activities are visible to third-parties, making this setting amenable to research. Therefore, we chose GitHub as a representative for general OSS activity and downloaded the August 2015 and January 2017 MySQL dumps of the GHTorrent database [23], which archives public GitHub activity like commits, projects, and users.[5]

However, we had to use other methods of data collection where GHTorrent lacked data. For example, GitHub allows a special kind of account called an organization account (which we refer to as "teams" in this paper)[6], wherein user accounts can be included as members of the organization account. However, GHTorrent does not maintain accurate data on whether developer accounts are members of teams on GitHub. In these cases, we downloaded webpages from GitHub to manually collect the relevant information.

3.3 Defining Membership on GitHub

To clarify what it means to be and become a member of a GitHub team, we assume the general applicability of the onion model which illustrates OSS communities as comprising several layers of population [1]. In essence, when we consider the "team" to be in general the group of individuals cooperating around a single goal or experiment, the inner layers of the onion model are the active developers and project leaders, and the outer layers are the observers and occasional contributers. To clearly define the border between inside and

[5]http://ghtorrent.org/relational.html

[6]Not to be confused with the GitHub feature of teams in organizations, which are a subset of organization members dedicated to a specific goal. However, this particular kind of "team" is not by default public to outsiders, so we cannot make claims about them. For more information, see the documentation: https://help.github.com/articles/organizing-members-into-teams/

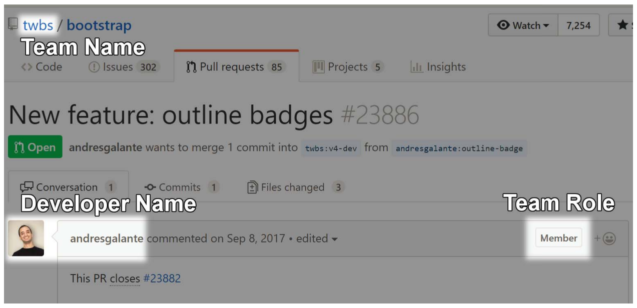
outside, we take inspiration from previous research that suggests that a critical transition in community standing is when project managers give an individual the ability to commit changes. We call this ability "write access". Not only is it often a remarkable increase in power for the new committer, but it also a signal from the project community that the new committer's contributions are valued and trusted [9, 24]. Therefore, we refer to people with write access as "insiders" of a development team, and we call people without as "outsiders." We use the word "contributor" to refer to anyone who interacts with a project, either from the inside or outside.

To examine how people transition from outsiders to insiders—in other words, how people join projects—we gathered two snapshots of GitHub wherein we could trace the joining behavior of users. The first snapshot occurred in August 2015 when we gathered data for an unrelated repository mining project; the second, in January 2017, when this study began. We focused on gathering the project roles of everyone who had initiated a pull request before our 2015 snapshot to reduce our scope because collecting information on every GitHub comment was time-prohibitive. However, this also means that everyone in our dataset had created at least one pull request, which is not necessarily representative of every type of developer on GitHub.

One way to identify members is through GitHub's interface. GitHub defines and displays its own set of team roles which teams across GitHub share. An individual developer's role in a team is visible whenever the developer participates in a discussion on GitHub, as in issues or pull requests, which we demonstrate in Figure 1. According to the documentation, some of the roles, such as "Collaborator" and "Member," indicate write access in the corresponding project.[7]

Unfortunately, GitHub, as an actively developed website, had changed the available project roles in the time between our snapshots. This change complicated our process of determining project movement: the 2017 role of "Contributor", for example, does not occur in 2015, just as "Owner" does not occur in our 2017 snapshot. As such, we had to manually find correspondence between 2015 and 2017 roles by investigating obsolete GitHub documentation and viewing archived versions of pull request discussions through the Internet Archive's Wayback Machine.[8] For example, by finding multiple versions of archived pull request discussions in which consistent project owners had participated, we could trace the labels which GitHub used over time to designate roughly the same permission levels. We repeated this tracing exercise multiple times until confident in our mappings, which we display in Table 1. For our analysis, we focus only on those who began as an outsider in 2015, marked without a label, and became a Collaborator or Member by 2017. We designate this population with the "join" keyword in the table.

Additionally, GitHub had also changed the default visibility of project roles between our snapshots. That is, developers on GitHub gained the choice to make their membership status private, visible only to other project members and not to third parties. This means



New feature: outline badges #23886

andresgalante wants to merge 1 commit into twbs:v4-dev from andresgalante:outline-badge

andresgalante commented on Sep 8, 2017 • edited

This PR closes #23882

Figure 1: How GitHub shows team membership.

Table 1: Movement in Teams

		2015		
		[No label]	Collaborator	Owner
2017	[No Label]	-	Leave	Leave
	Contributor	-	Leave	Leave
	Collaborator	Join	-	-
	Member	Join	-	-

that although we can still be certain that a public label corresponds to membership, we cannot be sure that the lack of a public label corresponds to the lack of membership. To overcome this obstacle, we consulted GHTorrent. There are a few actions that only project members can perform, particularly merging or closing somebody else's pull request or committing to the project without a pull request. Therefore, we looked for developers who had performed either of those two actions anytime before our snapshots according to the GHTorrent data; if they had, we considered them to have been a Member.

In summary, we identified people who joined projects by one of two ways. First, if they had a public role in a team's pull requests. Second, if they had performed member-exclusive actions in the project. Because we are studying how people join teams, we defined "joiners" as developers who fulfilled neither criteria at the 2015 snapshot and fulfilled at least one of the two at the 2017 snapshot. Developers who fulfilled neither criteria in 2017 are those that remained outsiders. We include the exact sizes of our population in Table 2, along with the population sizes after the other filters we use.

3.4 GitHub Contributions

To answer our research questions, we define and collect metrics for three different cases, given a particular user and a particular team: for RQ1, the activity that occurs between that user and team specifically; for RQ2, the activity that the particular user performs outside of that team; for RQ3, the activity that occurs within the team other than from that user. For each of these, since we have no information on exactly what time between 2015 and 2017 the individual users became members, we only included activity registered on the website before our 2015 snapshot.

3.4.1 Metrics for the Relationship Between User and Team. In GitHub's collaborative environment, a user can interact with teams

[7] Project owners can finely tune project permissions, meaning that write access is not guaranteed for everyone of a certain role. However, we assumed write access is usually the case. For more information, see the official documentation here: https://help.github.com/articles/repository-permission-levels-for-an-organization/
[8] https://archive.org/web/

and their projects in several ways. To characterize the relationship between an individual and a team, we focus on interactions that a project outsider can perform with default permission on GitHub. Furthermore, we aggregate all the interactions as simple quantities, counting the number of each interaction in our time window.

Attention is often the most basic form of contribution that outsiders give to a project, yet it is nevertheless important [1]. Sheoran and colleagues report that contributors who begin their interactions by watching a project are often more confident and versatile than other contributors [18]. Therefore, we count the ways that users can express interest in an team and their projects without directing affecting any of its contents. This includes the number of team projects that a user *watches*, which is asking to receive regular updates, and the number projects a user *forks*, which is making a project copy in order to experiment with the underlying code without necessarily making any permanent changes or suggestions.

For many users, the next step comes in discussions and bug reports. von Krogh and colleagues have already found that discussion engagement has a positive influence on an outsider's joining behavior [15], just as Sinha and colleagues established contributors participate in bug tracking systems before contributing code [16]. Therefore, we count the number of *issues* that an individual opens in a team's projects; issues are discussion threads for bug reports and suggestions pertaining to the corresponding project. Furthermore, we also count the number of subsequent *comments* that users add to issue discussions and remarks on specific lines of code.

As the user gathers confidence and expertise, the next step inward is often to offer code itself. On GitHub, each discrete bundle of changed code is referred to as a commit; however, because users can not contribute commits to projects with default outsider permissions, we do not count these. Instead, outsiders must submit commits in a *pull request* for evaluation and approval or rejection by the project members. We count the number of these that users submit to team projects instead. However, we must note that since we determined team standing by examining pull requests, we thereby artificially guaranteed that everyone within our dataset would have at least 1 pull request to their respective team, but other metrics would have no such guarantee.

We also took into account the research suggesting that pull request comments represent controversy in a contribution [25, 26]. This leads us to expect that the social dynamics of an individual making comments on their own pull requests would be different, perhaps more defensive or explanatory, from discussion on a different individual's pull request. For that reason, we split comments into four categories: *issue comments*, *commit comments*, *comments to pull requests that the individual created* and *comments to pull requests made by other individuals.*

Lastly, we included the amount of time in days between the developer's first interaction with the team and our first snapshot. We refer to this as *tenure (in days)* with the team. While we are interested in seeing what factor time plays in the development of affiliation, we also acknowledge that the breadth of our time window between snapshots—over a year—also makes our measure of tenure a rough estimate.

3.4.2 Metrics Describing the User. For RQ2, we collect the same metrics as in Section 3.4.1 but eliminate the need for the interactions to be with a specific team. At the same time, we exclude all the interactions between the developer and team in question as to not consider them twice within the same analysis. We include these metrics to untangle the effects of their general activity from the activity they target at one specific team.

In addition, we are able to add a few more relevant metrics to our analysis. For one, we can integrate the number of *commits* an individual has made to their own projects, because default permission allows individuals to make commits to their own projects. We also include the number of *non-forked projects* they themselves have created.

We also measure the examined user's social impact in light of the interest it stokes in other research [27]. To measure the impact that the user's projects have, we count the number of unique individuals who have *watched or forked* a project from our targeted individual, added into a single number. To measure the impact that the developer has as an actual identity on the website, we measure the people who are *followers* of this developer specifically, as well as the number of users that this individual is *following* themselves.

3.4.3 Metrics Describing the Team. To measure the qualities of a software team, we follow the research that draws connections between the size and maturity of teams and their inclination to accept new contributions [25], in addition to including the basic metrics in Section 3.4.1. For one, we count how many projects the team is responsible for as the *number of projects which the team owns.* Furthermore, we calculate how many people have previously contributed to the team's projects (i.e., discussions and code), which comprises the body of *contributors.* Lastly, just like in Section 3.4.2, we approximate the social impact of the team by counting the number of unique individuals who have *starred, watched, or forked* projects from this team.

3.5 Data Cleaning

We filtered our data to capture only the relationship in which we were interested: individual developers that interacted with teams of which they were not members. We did not include any relationships for which a developer contributed to an individually owned repository or for which the developer was already a member.

However, even among the valid developer–team relationships that we observed, many had been initiated several years before the first snapshot and there had been no interactions since. One problem is that any change in membership for relationships like this without any interactions visible to us could indicate meaningful activity in other channels of communication that we cannot account for. Another problem is that this would inflate the "Tenure" metric that we include in our analysis without accounting for long periods of inactivity. To avoid these problems, we considered only developer-team relationships for which we could confirm there were interactions in or after 2015, and we threw out all the observations that captured interactions only before our starting year.

By manual inspection of the outliers in the remaining data, furthermore, we discovered that many of our most outstanding records belonged to bot accounts. Many of these of following a common naming scheme: ending with "-bot." We then removed all observations including those accounts.

3.6 Analysis

To determine the most characteristic activities of joiners, we sought to create a logistic regression because it conduces to our binary outcome variable—whether someone joins an team or not as a 1 or 0 respectively—and makes fewer assumptions about the distribution of each of our metrics and their residuals. Additionally, we also log-scale all our explanatory variables in order to reduce the influence of our outliers, as the distributions for many of our metrics have a long right tail.

Furthermore, because the core concept in our observations are the developer-team interactions, we have many developers or teams which are represented in several different groups of interactions. Thus, rather than use all of our data in the analysis and overrepresenting any individual user or team who had interacted with a disproportionate number of other parties, we create two separate logistic analyses from random samples for which we will not have any user or team appearing multiple times within the same analysis. Our first analysis takes all valid teams and selects two individuals for each: one outsider who eventually joined the team, and one who did not. If we find a strong coefficient in this analysis, it is to say that the corresponding metric is a strong distinguishing characteristic from the point of view of the team; in other words, if metric x is significantly positive, then teams prefer developers who contribute many x's. Our second analysis flipped the first to find users who had interacted with at least two teams: one of which they joined and one of which they did not. Here, the focus was on controlling for differences in developers and examining the impact of different teams. In Analysis 2, if metric y is significantly positive, then users tend to contribute more y's to the teams that they join, or developers more often join teams that are responsible for more y's. As such, these two analyses are related but subtly different in perspective, and including both allows for better triangulate the source of the differences. If we observe differences in the effect sizes and significances for relationship metrics in the two analyses, then this result might suggest the lack of a coherent pattern of activity for joiners that applies to both team and developer.

To build these analyses, we used a modified maximum flow algorithm to ensure that we had no users overlapping between teams. For Analysis 1, we found 2,440 relationships that fulfilled these requirements, representing 1,220 unique teams. For Analysis 2, we found 1,036 observations, representing 518 developers. However, these two analyses share 281 observations about developer-to-team interactions, although these relationship metrics are combined with different baseline metrics in each analysis. A majority of the observations in each analysis are unique to that analysis, but some similarity between may be a result of this overlap.

4 FINDINGS

Under our research questions that ask how the activity of joiners differs from that of non-joiners over the same time period, we construct a logistic regression with the aforementioned explanatory variables. We present the odds ratios and their and standard errors in Table 3; the numbers with asterisks are those which were significant for a $p < .05$. For logistic regressions in general, the odds ratio represents the change in odds that an individual joins an organization when the explanatory variable has a unit increase. For our example, the estimate for the odds ratio for every unit increase in pull requests is 1.75 in Analysis 1 assuming all other metrics remain unchanged; since our data was log-scaled, this proportional increases happens at the milestones of 2^1, 2^2, 2^3, and so on.

Table 2: Data Parameters

Data Subset	N
Outsider Relationships in 2015	252,243
Relationships After Cleaning	247,589
Relationships with Contributions In 2015-7	64,270
Nonjoining Relationships	61,096
Joining Relationships	3,174
Unique Developers	47,352
Unique Teams	11,095
Analysis 1—Paired Developers by Teams	2,440
Analysis 2—Paired Teams by Developer	1,036

For our metrics concerning relationships, we found significant differences for three metrics in both analyses: the number of pull requests, comments on other people's pull requests, and the days since first interactions. All of these metrics have positive effects in both analyses. The number of issue comments was significant in Analysis 2 only. For metrics of the individual developer, the number of commits to personal projects and personal followers had a significantly positive impact on joining. For team metrics, the number of projects, pull requests, and overall number of contributors had a positive impact on joining, whereas the number of people watching or forking the team had a negative impact on joining and overall number of commits.

We include the descriptive statistics as well, first, in order to characterize our data more concretely, and second, to point out that joiners are, on the surface, more active in general across every kind of activity within the software communities. Furthermore, since each analysis is built on pairs of observations for joining and nonjoining behavior, we also included in each table the pairwise difference for each metric. That is, we subtracted the quantities of nonjoiners' metrics from the quantities of joiners. A positive result indicates that the joiner had more activity in that dimension than the nonjoiner. Done over all pairs that compose each analysis, we report the medians and interquartile ranges (IQRs, or the 25%th and 75%th percentiles) of these differences in the rightmost columns of each table. We preferred medians to means because they were more insensitive to the influence of outliers in our data. All results are statistically significant with the Wilcoxon rank-sum test with the Benjamini-Hochberg correction. These tables are available within this project's associated figshare.[9]

For each corresponding metric, the two analyses are generally consistent in odds ratios and medians. Furthermore, the quantities that characterized joiners tended to be higher, or at least not lower, to their nonjoining peers in every respect. An example applied to RQ1, by the time of our first snapshot, joiners in Analysis 2 had a median of 3 pull request submissions to their targeted team, with 50% of the population submitting between 2 and 10 pull requests. On the other hand, 50% of the nonjoining population had contributed

[9] https://figshare.com/s/896222feb33b06dd061a

Table 3: Odds Ratio (OR) per Unit Increase of Each Metric

	log_2(Metric)	Analysis 1		Analysis 2	
	Intercept	0.08***	[0.04, 0.14]	0.04***	[0.01, 0.14]
	Statistically Significant Results				
RQ1: What types and qualities of project interactions distinguish between developers who join an OSS team and developers who remain outside?	Pull Requests (PRs)	1.58***	[1.42, 1.77]	1.86***	[1.50, 2.33]
	Issue Comments	1.04	[0.95, 1.14]	1.20*	[1.04, 1.39]
	Comments on Other PRs	1.20***	[1.09, 1.32]	1.27**	[1.07, 1.51]
	Tenure (Days)	1.08*	[1.00, 1.16]	1.14*	[1.01, 1.29]
RQ2: What activities or qualities of a developer independent from a specific team influence whether the developer joins that team?	Commits to Personal Projects	1.07*	[1.00, 1.14]		
	Followers	1.15**	[1.05, 1.26]		
RQ3: What activities or qualities of a team influence whether the team accepts more people in?	Projects in Team			1.20**	[1.06, 1.35]
	Commits in Team			0.84**	[0.74, 0.95]
	PRs in Team			1.30**	[1.07, 1.57]
	Contributors to Team			1.21*	[1.00, 1.45]
	Watchers, Forkers in Team			0.84**	[0.73, 0.95]
	Statistically Insignificant Results				
RQ1	Issues	1.08	[0.96, 1.22]	0.93	[0.77, 1.13]
	Commit Comments	1.11	[0.93, 1.34]	0.94	[0.71, 1.25]
	Comments on Own PRs	1.00	[0.92, 1.07]	1.05	[0.91, 1.22]
	Team Projects Forked	1.11	[0.96, 1.28]	1.05	[0.79, 1.38]
	Team Projects Watched	0.93	[0.82, 1.06]	1.07	[0.88, 1.31]
RQ2	Personal Projects	0.99	[0.90, 1.09]		
	Other PRs	1.01	[0.92, 1.09]		
	Other Issues	0.95	[0.88, 1.04]		
	Other Commit Comments	0.98	[0.92, 1.06]		
	Other Issue Comments	0.98	[0.90, 1.07]		
	Other Comments to Own PRs	1.06	[0.98, 1.15]		
	Other Comments to Other PRs	1.00	[0.94, 1.07]		
	Following	0.99	[0.94, 1.05]		
	Watchers, Forkers for Own Projects	0.96	[0.90, 1.01]		
RQ3	Issues in Team			1.11	[0.88, 1.40]
	Commit Comments in Team			1.09	[0.99, 1.20]
	Issue Comments in Team			0.86	[0.71, 1.06]
	PR Comments to Org			0.97	[0.84, 1.13]

$^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$

only 1 or 3 pull requests. However, when taking these descriptive statistics along with the coefficients in the logistic regression, we see that not every form of activity influences the final joining decision equally. This distinction allows us to make finer discussion about the salient forms of activity even when the initial impression could be that more activity in general is more influential.

5 DISCUSSION

Given the results in the previous section, we are able to draw inferences from the significant metrics about what kinds of activity on GitHub influences joining behavior. We discuss the implications for each research question below.

5.1 How Do Developer-To-Team Interactions Influence Joining?

The number of pull requests submitted has the highest influence per unit in both analyses. Although we did not include an analysis of the rate of pull request acceptance, this confirms at least the attempted contribution of code as an important step within the community. Yet the influence does not end at the submission of pull requests; both models suggest that engagement in other people's pull requests through comments boosts the odds that a developer joins the team. Additionally, Analysis 2 further confirms that the influence extends to multiple forms of discussion, both issues and pull requests alike. Considering that each of these activities has a different “cost" of contribution—it could take longer to create the code for a pull request than the text for a comment—success in affiliation need not be the overwhelming success in one activity but rather the measured combination of several, given a developer's time and opportunity.

Interpreted from the onion model's layers, the positive influence of activity generally increases with the investment in the project. That is, external code contributors in projects, situated toward the inner layers, are fewer but more influential and invested than those who only participate in discussions or less, so this model supports an explanation that inner-layer activities help to explain the divide between joiners and nonjoiners. The significance of our Tenure metric, which captures how long the individual has been involved with the project, also underscores the importance of time in moving through the layers by the joining script, as underscored by von Krogh and colleagues [15]. However, without sufficient statistical significance to accept the estimated influence of metrics from every layer in that model, there remains room for future work that clarifies the relationship.

As in RQ1, pull requests come up again in RQ3, along with project forking. Since pull requests and project forking are closely related activities, our findings suggest that this area of software development is worthy of close consideration in subsequent research. Dabbish and colleagues have already discussed how GitHub's transparent interface assists organization development, allowing team members to see the community interest in the project, even to the point of recruiting the most active participants [20]. But our websites for collaborative development must also support fine-grained monitoring from the other direction: from interested developer over the team activity. Since joining developers often participate in many pull requests before joining, websites should signal the relationships and dependencies between projects so developers can better understand development across a software team and perceive the opportunities for contribution in context of the entire team or other forkers. Furthermore, since the act of proposing the changes in a fork is also a critical moment for a developer's impression on a team, visible community standards or integrated checklists and reminders in website design could intervene during the pull request creation process to reduce the need for copious comments after submission.

For interactions between developer and team, more is generally better. Pull requests influence a team's decision most and frequent commenting in discussions is also a positive influence. Time and familiarity is a general benefit as well.

5.2 How Does Independent User Activity Influence Joining?

When we look at developers without considering specific teams, only one form of contributions was significant influences to joining behavior: the number of commits to personal projects. Furthermore, one of the social metrics that we included, the number of followers to a developer's account, was significantly positive.

To some extent, these findings confirm some of the common wisdom and previous work concerning the impact of one's social standing on general approval. Tsay and colleagues, for example, had investigated the impact of the number of followers on whether a developers contributions will be approved [25]; here, we find that it further determines whether that developer will be more likely to be onboarded onto the team as a regular contributor. And as Dabbish and colleagues state, the more followers one has, the more likely it is that they could be a "coding rockstar" whose name precedes them [20]. As such, followers act as a form of short-circuited approval to a team, demonstrating that a number of other people already trust and value their work, making it more likely that even more will trust them in the future. The effect of followers, however, also has a flipside for those without; developers without followers will have to work more in order to prove their worth given the same starting context otherwise.

Therefore, intervening in website design could help both the team and the individual developer. As mentioned in Section 5.1, giving developers more control in designing their personal sandboxes can allow them to appropriate space devoted towards specific teams and projects whether or not they're official members; with better designed spaces, they can learn more specifically about those team projects in their respective contexts and communities. For the team perspective, there already exists some surface quantification of a developer's contributions to a specific project—the number of commits, for example. But quantifying their interactions with the team in context of the total website activity more clearly could better help organizations pinpoint developers with investment in that team among their peers.

The number of commits in personal projects and the number of followers on one's profile, both of which are readily visible on a user's profile page, constitute positive influences on an individual's joining odds.

5.3 How Does Independent Team Activity Influence Joining?

Our second analysis suggests statistically significant effects for five of our team metrics: the numbers of projects, commits, pull requests, along with the number of unique contributors and the number of watchers and forkers. Furthermore, the odds ratios in Table 3 suggest influences in opposite directions: positive for projects, pull requests, and total contributors, negative for commits and watchers.

Since these statistics are calculated from the perspective of the individual developer choosing between projects, we interpret these differences as the factors that determine the receptivity of projects. From this perspective, and especially considering the opposite directions that similar metrics push the developer, we infer specifics tension in team management between maintaining quality through exclusivity and ensuring support through inclusivity that has also been addressed by other researchers [12]. That is, more active projects require more people to maintain them, incentivizing the team to accept more people. In other words, more team projects indicates, in general, more opportunities for contribution for interested developers, and so developers are more likely to join these teams. At the same time, the more outsiders who pay attention to a team's work as measured through watchers and forkers, the more pressure there is on the team to ensure quality to their audience [20]. As such, teams with more impactful projects are harder to get into.

Just as teams navigate a balance of skill and free time when negotiating new developers as discussed in Section 5.2, so too are developers negotiating a balance of opportunity and promise when seeking teams that will both accept and benefit them. While some teams assuage this search by posting explicit openings for new developers, communities and websites can do more to support the matchmaking problem and help developers discover and focus on receptive projects.

The more projects and pull requests a team has, the more likely it is that developers are accepted by that team. However, when those projects are popular with watchers and forkers or comprise many commits, the joining odds decrease.

5.4 Implications

Our analyses suggest that many kinds of online interactions are significant for the eventual joining of a software team. Given the behavioral patterns that our analyses support, we can then make suggestions, for example, about the design of websites, communities, and educational curricula.

5.4.1 Community and Website Design. The populations most directly affected by our work here are the online OSS communities themselves. With our outline of influential activities for team interactions, we reinforce for project newcomers the importance of continued social interaction in addition to technical contributions. Furthermore, given that this work more strongly suggests the existence of joining patterns, teams can further facilitate the attraction and participation of newcomers by publishing guidelines for contributions near their projects. Some already do, but spreading this behavior can further standardize the behavior of contributors into more effective patterns.

To some extent, the websites hosting OSS communities can implement interface features to support this standardization. For example, with a progression akin to the onion model in mind, websites can assist new users with tutorials or checklists to outline the best patterns and practices for community involvement built out of the data from previous successful joiners. For one, allowing developers to see clear quantifications for not only how many lines of code they have contributed but also how much they have commented in a team's projects or how many people they know within in it could help the newcomer into valuable reflection with their involvement so far. Furthermore, adding reminders and checklists for common actions not yet taken—for example, not yet participating in any issue discussions or pointing out when pull requests descriptions are likely insufficient—could provide concrete guidance for bettering the newcomer's impression on the team.

5.4.2 Educators. As the influence of OSS communities grows, the importance of teaching less experienced developers how to interact with these communities will grow as well. For the educator looking to emphasize and incorporate certain kinds of behaviors into their curricula on modern OSS communities, these patterns provide a starting point. Although this research does not provide hard boundaries which determine, for example, how many pull requests are necessary to ensure that a team accepts a developers, it does allow us to see where the students should focus their work.

For example, while issues and bug reports are nevertheless an important part of collaborative software development, the results from our work suggest that this behavior alone might not be enough to make a lasting impression on a software team. The stronger behavior, a teacher might point out and encourage, is the continued interaction within the discussion and peripheral decision-making processes carried out in discussions across the community.

In other words, education and preparation for OSS involvement is different than the simple mastering of a particular language or toolset. Instead, our results suggest a pattern of community involvement and social skills that are nonetheless important for successful collaboration in the long term.

6 LIMITATIONS

A primary limitation of our study is in the incompleteness of its data. Our information on membership came from a previous project in which we were primarily interested in pull requests. As such, all of our observations come from people who had made at least one pull request (accepted or rejected) to a team by mid-late 2015. However, pull requests are by no means mandatory for all teams, which means we missed any new members who became such without a pull request as well as every user who had not created a pull request by that time. Some of our chosen metrics are also conceptually similar, meaning that collinearity effects could reduce the influence of our findings here. We also did not have data specific enough to pinpoint exactly when an individual joined a team and instead narrow it to a window of more than a year. Because of the width of this window, we lost valid data for each user's interactions with a team before joining which we cannot precisely quantify.

Another significant gap in our data is the intentions of developers in their interactions with teams. In this study, all that we have been able to say is, to some certainty, whether an individual has joined a team or not. But, if they did not join, we cannot say whether it was because they were not accepted based on their contributions or because they simply didn't want to join and did not accept an invitation. Without controlling for the numerous reasons why someone might or might not join, we can only discern patterns that exist across intentions, meaning that there might be important distinctions in behavior that we cannot broach here.

Furthermore, we had no data on any of the interactions that happens off of GitHub. As Kalliamvakou and colleagues note, many projects do not conduct all of their work on GitHub, opting instead for mailing lists, forums, and other platforms. Furthermore, we can access only a user's public data [28].

Despite our theoretical underpinnings in models of OSS communities, we acknowledge that these are not a perfect fit. For one, GitHub does not perfectly follow the core-periphery model in terms of their explicit roles and populations; in our 2017 snapshot, for example, we deemed "Collaborator" as a role between outsider and core team "Member," yet Members outnumber Collaborators for many teams. Furthermore, not all developers will follow the same pattern of inward movement, as Herraiz and colleagues point out differences between volunteers and hired developers [19]. In the absence of a way to generally discern between volunteers and hired work, we clustered them all together under the assumption that most organizations on GitHub were defined by volunteers; if this assumption is incorrect, then our findings only reflect the behaviors that are true of the majority population and not necessarily the behaviors unique to volunteers.

7 FUTURE WORK

First, as we pointed out in Section 5, our results contain many seeming tensions between related metrics that might be worth investigation. This includes the following:

- What explains the discrepancies in impact between the analyses?
- Why would the number of team contributors positively influence joining behavior whereas team watchers and forkers negatively influence the same behavior?

We proposed hypotheses as to explain these tensions in our data but did not design our analysis to answer them specifically. By choosing any of these or related questions and focusing on the specific metrics contained in them, we could perform a closer analysis of the contents of those contributions to advance stronger suggestions about these seeming contradictions.

Furthermore, as we have discussed in Section 6, our analysis does not take into account the different motivations developers have for joining or not joining development teams. Additionally, we also do not consider other forms of team movement like the departure from teams, made more significant by previous papers that note that a high turnover rate is expected by the managers of teams [9]. In these ways, this paper represents a surface scan across the dynamics of team development on GitHub that leaves room for deeper analysis. By including analyses of how behavior varies with intention or how behavior on GitHub changes over the lifecycle of project membership, we can gain a more nuanced understanding of collaborative behavior online.

Lastly, the study here is based on discrete, visible contributions without considering the content. One extension is to supplement these contributions with content analysis. We can break down commits based on the code or text that they add to a project, and we can analyze the sentiment of comments to see how different attitudes might be indicative of the bond between a contributor and the team. To add to this, there is also the chance to examine the contents of social networks—that is, how developers have interacted materially before and how that influences joining behavior as well.

8 CONCLUSION

In this study, we deconstruct the relationship between individual developers and teams on GitHub by their visible contributions across the open source website, including commits, issues, pull requests, and comments. While we indeed find support for the idea that increases in activity correlate with a higher probability for membership, we also found the particular cases for which more activity can reduce the probability. Our logistic regression suggests that joiners are often defined not by a disproportionate amount of one particular form of activity but by more activity across several dimensions of project involvement. This underscores the notion that software collaboration is much more than the code itself and that the social components of software should not be undervalued by software teams.

With our findings, we seek to emphasize the full range of collaborative processes to software communities and the platforms that support them. The methods to attract and onboard new developers may change slightly with each different interface and paradigm of software collaboration, but we believe there are patterns in human behavior itself that we can suss out through this work and in future work. After all, the wellbeing of OSS projects determines to a large extent the future of not only software engineers but software consumers in general.

9 ACKNOWLEDGMENTS

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

REFERENCES

[1] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in *Proceedings of the international workshop on Principles of software evolution.* ACM, 2002, pp. 76–85.

[2] O. S. Initiative, "The open source definition," https://opensource.org/docs/osd.

[3] B. Fitzgerald, "The transformation of open source software," *Mis Quarterly*, pp. 587–598, 2006.

[4] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer *et al.*, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference.* ACM, 2014, pp. 475–488.

[5] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu, "Open borders? immigration in open source projects," in *Proceedings of the Fourth International Workshop on Mining Software Repositories.* IEEE Computer Society, 2007, p. 6.

[6] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.

[7] T. T. Dinh-Trong and J. M. Bieman, "The freebsd project: A replication case study of open source development," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 481–494, 2005.

[8] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 4, pp. 323–368, 2005.

[9] S. K. Shah, "Motivation, governance, and the viability of hybrid forms in open source software development," *Management Science*, vol. 52, no. 7, pp. 1000–1014, 2006.

[10] G. Robles, J. M. Gonzalez-Barahona, and I. Herraiz, "Evolution of the core team of developers in libre software projects," in *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on.* IEEE, 2009, pp. 167–170.

[11] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of diversity on git hub: A user survey," in *Cooperative and Human Aspects of Software Engineering (CHASE), 2015 IEEE/ACM 8th International Workshop on.* IEEE, 2015, pp. 50–56.

[12] C. De Souza, J. Froehlich, and P. Dourish, "Seeking the source: software source code as a social and technical artifact," in *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work.* ACM, 2005, pp. 197–206.

[13] K. Crowston, K. Wei, Q. Li, and J. Howison, "Core and periphery in free/libre and open source software team communications," in *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, vol. 6. IEEE, 2006, pp. 118a–118a.

[14] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: an empirical study on count and network metrics," *arXiv preprint arXiv:1604.00830*, 2016.

[15] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[16] V. S. Sinha, S. Mani, and S. Sinha, "Entering the circle of trust: developer initiation as committers in open-source projects," in *Proceedings of the 8th Working Conference on Mining Software Repositories.* ACM, 2011, pp. 133–142.

[17] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in Github: the role of prior social links and language experience," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.* ACM, 2015, pp. 817–828.

[18] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding watchers on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 336–339.

[19] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. González Barahona, "The processes of joining in global distributed software projects," in *Proceedings of the 2006 international workshop on Global software development for the practitioner*. ACM, 2006, pp. 27–33.

[20] L. Dabbish, C. Stuart, J. Tsay, and J. 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*. ACM, 2012, pp. 1277–1286.

[21] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in github," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 117–128.

[22] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider, "Creating a shared understanding of testing culture on a social coding site," in *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013, pp. 112–121.

[23] G. Gousios, "The ghtorent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 233–236.

[24] C. Jensen and W. Scacchi, "Modeling recruitment and role migration processes in ossd projects," *ProSim05*, vol. 39, 2005.

[25] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th international conference on Software engineering*. ACM, 2014, pp. 356–366.

[26] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 2015, pp. 367–371.

[27] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," *arXiv preprint arXiv:1407.2535*, 2014.

[28] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.