Restrictions in Open Source: A Study of Team Composition and Ownership in Open Source Software Development Projects

Completed Research Paper

Poonacha K. Medappa
HEC Paris
1 Rue de la Libération, Jouy-en-Josas, France 78350
poonacha.medappa@hec.edu

Shirish C. Srivastava
HEC Paris
1 Rue de la Libération, Jouy-en-Josas, France 78350
srivastava@hec.fr

Abstract

Grounding our study in the theories of coordination and network governance, we examine the influence that restricting the access to the source code has on the likelihood of survival of open source software projects. Furthermore, given the increasing shift of organizations towards adopting the open source development approach, we investigate the changes brought to the coordination mechanisms when open source projects are owned by organizations. The results from our analysis of about 6500 open source projects indicates the presence of a mixed relationship between the proportion of contributors who have write access and the survival of the project. Further, we find that ownership type (organization vs. individual) creates a crossover interaction that changes the nature of relationship between proportion of contributors who have write access and the survival of the project. The findings provide researchers and practitioners with insights for designing effective open source project teams that can sustain contributions.

Keywords: Open source software, software development, team composition, access restrictions, coordination, network governance, survival analysis

Introduction

In the current digitally enabled collaborative environment, Free (Libre) and Open Source Software (FLOSS) projects have become ubiquitous. Moreover, the evolved coordination and improved motivational mechanisms employed in FLOSS projects, like that seen in its work structures (Howison and Crowston 2014; Lindberg et al. 2016) and social practices (von Krogh et al. 2012), have allowed commercial organizations to look towards FLOSS as a viable mode of software development. These mechanisms, when correctly leveraged can allow project owners to tap into the vast reserves of programming skills spread across the globe, enabling them to create software that surpasses proprietary software in both quality and functionality (Coverity 2013). The potential that this model of development offers to Information Technology (IT) and non-IT industries has attracted considerable attention from the academic community, which has tried to better understand the antecedents to FLOSS project success (Crowston et al. 2012). The examined antecedents of project success include: the choice of license types, project sponsorship (Stewart, Ammeter, et al. 2006), user and developer base, project complexity (Midha and Palvia 2012), network embeddedness of the project (Fershtman and Gandal 2011; Grewal et al. 2006), and software components (Crowston et al. 2012). Within the large body of research that looks at FLOSS success, relatively less work has tried to examine the influence of contributor access restrictions and team composition on the success of projects (Rullani and Haefliger 2013). Since the effective organization of teams can often be the difference between profiting from the advantages vs. succumbing to the challenges of distributed work, there is a need to better understand the intricacies of team composition (Sagers 2004).

FLOSS project contributors are often classified as belonging to either the core or the peripheral group of developers (Crowston, Wei, et al. 2006). Studies that have looked at the contributor groups from the nature of their participation and type of contribution have already established the importance of both the core and peripheral groups as antecedents to the success of the project (Setia et al. 2012). Further, the duality between the different groups of contributors has been found to be essential for FLOSS projects to grow and innovate (Rullani and Haefliger 2013). At the project team level, teams comprising the right mix of contributor roles can support the teams' organizational, intra-personal and inter-personal processes thereby influencing the performance of the team (Licorish and MacDonell 2014). By offering a unique perspective on contributor types and team composition in FLOSS projects, the aforementioned studies have unearthed three interesting avenues for expanding the theory. First, most studies have treated the two contributor groups independently without investigating the interplay between them owing to their relative composition in the project team. Second, most extant researches, have examined the influence of contributors on project success in nonorganizational settings that are characterized by high levels of openness and limited management (Crowston, Wei, et al. 2006; Rullani and Haefliger 2013; Setia et al. 2012). It is unclear if and how these mechanisms get altered when organizations own the project and introduce formal project management approaches coupled with different motivational practices (Capra et al. 2011). Third, the emergence of Distributed Version Control Systems (DVCS) based development platforms like GitHub, allows a rather unique definition of core and peripheral contributors based on their rights of access to the main project code (Rullani and Haefliger 2013). This classification differs from the prior research that has adopted a classification based on percentage of code contributions (Crowston, Wei, et al. 2006; Setia et al. 2012) and the relative communication density (Licorish and MacDonell 2014). The classification of contributor groups based on the permissions afforded to them provides an updated understanding of the socio-technical influence of access restrictions on the survival of projects.

Beyond the three aforementioned gaps, FLOSS researchers have constantly been interested in better understanding the constituents of FLOSS project success (Crowston, Howison, et al. 2006). Surprisingly, very few studies have tried to examine success from the perspective of project survival – i.e. in terms of the project's ability to sustain itself and remain active for prolonged periods of time (Chengalur-smith et al. 2010; Gamalielsson and Lundell 2014; Samoladas et al. 2010). Unlike traditional software development which often comes with specific goals and delivery dates, a FLOSS project starts as an 'itch worth scratching' and if the project manages to engage the open source community it can go on to become a large project that continuously attracts community attention and contributions (Raymond 1999). Thus, the FLOSS project's ability to withstand the test of time and remain active for a longer duration can be considered as an important measure of its success, which differentiates it from traditional software development. With organizations and individuals experimenting with FLOSS as a source of sustainable software projects, understanding the antecedents of FLOSS project survival is valuable for both practitioners and researchers (Gamalielsson and Lundell 2014).

Driven by the need to establish a deeper understanding about the mechanisms associated with FLOSS team composition and its influence on the survival of the project, and also to identify the implications that these mechanisms have for project owners (both individuals and organizations), our research addresses the following two questions: a) *What role does access restrictions on contributors have in influencing the survival of FLOSS projects? b) How does organizational ownership moderate this relationship?* By answering these research questions, we aim to make the following contributions to theory and practice. First, we advance the understanding of the contributor types and team composition in FLOSS environments and their influence on the survival of the FLOSS projects. By grounding our research in coordination theory (Malone and Crowston 1994) and the network governance (Jones et al. 1997; Wasko et al. 2004), we attempt to offer a more nuanced conceptualization of the mechanisms operating between the access restrictions to the source code, team composition and the survival of the project. Second, we contribute to the theoretical understanding of the different ways in which, the type of contributor and their composition in a project influences project survival in individual and organization owned FLOSS projects. Prior research has examined many important aspects related to organizational participation in FLOSS projects and has laid the groundwork for a deeper theoretical inquiry (Capra et al. 2011; Fitzgerald 2006). Building on this prior research, we show that the influence of access restrictions and team composition on the survival of the project is sensitive to the nature of ownership of the project. We believe that cognizance of these influences is crucial for facilitating the success of FLOSS projects. Lastly, the results from this study helps advance theoretical understanding on FLOSS success and offer practical insights for project owners. In particular, by looking at project success from the perspective of sustenance and project survival, we shed light on a less understood dimension of FLOSS projects (Chengalur-smith et al. 2010). Management scholars have expressed considerable concern about the failure of academic research to penetrate the practitioner community (Rynes et al. 2001). We believe that a better understanding about the influence of the nature of contributors on the project survival can lead to better management practices for planning potentially successful FLOSS projects.

The rest of this paper is organised as follows; in the next section, we provide a brief literature background on of the important concepts relevant for this study. We then develop our theoretical arguments surrounding the concepts and present our hypotheses. In the methodology section, we provide details of our research setting and measures. We follow this with a discussion of the empirical models used and an analysis of the results. Finally, in the discussion and conclusion section we discuss the implications of our study for theory and highlight the major limitations in our research.

Theoretical Background

To understand the influence that team composition has on the survival of individual and organization owned projects, three important concepts need to be developed: a) contributor type and team composition, b) FLOSS project success and survival and, c) organizational ownership of FLOSS projects. In this section, we set the context for each of the concepts and provide a brief background on the related literature.

Contributor type and team composition

In DVCS systems like Git which is used in the GitHub platform, contributors can clone latest version of the project files, creating a full mirror of the project (including all of its prior versions) on their local machines. This feature enables the creation of several remote branches of projects, which allows concurrent work among different groups of people within the same project (Gousios et al. 2014). Once development work is complete in any of the local branch, the changes can be submitted back to the main project branch by creating a request for merging (called as "pull request" in GitHub). In GitHub, a change to the main project branch can be accomplished through either a push or a pull request event. A push event is created when a change is made to the project directly by a contributor who has push (write) access to the project. The project owner can grant write access to any contributor by identifying the contributor as a "collaborator" for the project. In contrast, any contributor (regardless of whether they have push access) can propose a change to the project by issuing a pull request. In this case, a contributor with write access can choose to merge the proposed changes into the project if they are found to be satisfactory (Gousios et al. 2014).

In communities of practice such as FLOSS, it is possible to differentiate the core of the community comprising individuals who are deeply engaged with the community activities, from the group of individuals who are more loosely linked forming the periphery of the community (Rullani and Haefliger 2013). From this perspective, it is possible to classify core-periphery groups based on four characteristics (a) access restrictions to the main project code, (b) percentage of contributions made by the contributors, (c) typology of contributions made by the contributors (i.e. degree to which their activities belong to critical tasks), and (d) communication density associated with the contributors (Crowston, Wei, et al. 2006; Licorish and MacDonell 2014; Rullani and Haefliger 2013). Because our theoretical framing relies on the coordination challenges incurred when offering access to the source code, and to account for the distributed nature of the workflow adopted in Git based DVCS systems, we use the extant of "access restrictions" to classify the core and periphery. That is, we consider the core as —the group of contributors who are given 'write access' to the main project code and the periphery as —the contributors who are *not* given 'write access' but still participate in development activities and contribute code by issuing pull requests. Further, this operationalization is aligned to Wenger (2005) conceptualization of communities of practice, where periphery is defined as the group of individuals who orbit the deeply engaged core group of contributors, participating sporadically in activities that are directly related to the core's ongoing activity. Apart from contributing to the code, the core group of contributors is given the added responsibility of overseeing the merge process, providing feedback, conducting tests, requesting changes, and finally accepting the contributions (Gousios et al. 2014). On the other hand, the periphery provides the core with a large workforce that can perform activities such as bug reports, patches, production, or useful feature identification (Rullani and Haefliger 2013). Further, the heterogeneity of skills allows some peripheral members to engage with the core and to enter the "center of the action," providing valuable out-of-the-box ad hoc solutions (Rullani and Haefliger 2013, p 945).

Most research on FLOSS contributor roles focuses on the differences between contributor roles and the process of role definition (Crowston et al. 2012). For example, Crowston, Wei, et al. (2006) provided one of the early works on understanding contributors in terms of core and periphery groups. In their research, the authors studied the distribution of contributions in the bug tracking systems identifying three different methods of classification of core-periphery groups. Subsequent research has tried to build on this understanding of the core-periphery groups by trying to understand their influence on different aspects of the project. For example, Setia et al. (2012), unearth the important role that peripheral contributors have in terms of enhancing product quality and product diffusion. On the other hand, Licorish and MacDonell (2014) focused on the core contributors attitudes, knowledge sharing behaviors and task performance. Similarly, based on psychometric text analysis conducted on the top developers of Apache httpd server project, Rigby and Hassan (2007) teased out the difference in personalities between the top and other developers. While these robust studies describe the nuances in the nature and type of contributions, as a next step, there is a need to understand the interplay between the core and peripheral groups, and the impact that their composition in a team can have on the project success parameters. In contrast, other studies have looked at contributor roles from the perspective of social practices associated with different contributor groups. In particular, Sagers (2004) in a survey based study of 38 FLOSS projects unearthed the impact of restricting access to the code base on improving coordination and safeguarding exchanges. Rullani and Haefliger (2013) described the division of labor between core and periphery groups in online communities of practice and unearthed the process through which standards of practice, emerged and propagated across the peripheral group of contributors. The aforementioned studies, by identifying the mechanisms that operate within core and peripheral group of contributors, offer an excellent groundwork on which we can expand our understanding of FLOSS team composition under different ownership forms, particularly in terms of their influence on survival of the project.

Project success

Despite the widespread use of FLOSS projects by individuals and organizations, measuring the success of such projects can be challenging because of the open source nature of these projects, which precludes their association with the usual monetary measures such as prices, revenues, and sales (Fershtman and Gandal 2011). Furthermore, FLOSS projects are freely available in the public domain, where there is no need to request for any permission to use them. Nonetheless, researchers have continually attempted to describe the meaning of success in the context of FLOSS projects. Crowston, Howison, et al. (2006) identified seven measures of FLOSS project success: system and information quality, user satisfaction, use, individual and organizational impacts, project output, process, and outcomes for project members. Similarly, Lee et al. (2009) developed a FLOSS project success model in which they identified five measures of FLOSS project success: software quality, community service quality, use, user satisfaction, and individual net benefits. Both early models accentuate the complexity and multidimensional nature of the FLOSS success construct. Most empirical studies that examine the mechanisms associated with FLOSS project success have operationalized success along one or more of the described dimensions (Crowston et al. 2012).

Although a diverse set of measures have been used to operationalize FLOSS project success (e.g. Fershtman and Gandal 2011; Grewal et al. 2006; Stewart, Ammeter, et al. 2006), we note that understanding success from the point of view of a projects' ability to sustain itself and remain active for a long duration of time is fairly limited (Chengalur-smith et al. 2010). Successful FLOSS projects are expected to see continuous enhancement and maintenance of the code, which results in multiple stable versions delivered one after another (Raymond 1999). Thus, sustenance or survival of the project can be considered as an important measure of success which often differentiates FLOSS projects from more traditional projects that may come with specific end goals and milestones. Further, in their empirical analysis of FLOSS projects success, Crowston, Howison, et al. (2006) found that the common measures used for success, such as popularity and quality were correlated to each other but the project lifespan was unique as it did not correlate with the other measures of success. Given this finding, survival analysis of the FLOSS projects can provide an independent but important view on project success not yet fully examined in literature.

Organization ownership

By early 2000, organizations began to realize the commercial potential of FLOSS. For example, in November 2001, IBM embraced the open source approach by opening the source code of several of its software tools (estimated at $40 million) to the public domain (Lohr 2001). This led to a transformation of FLOSS communities, which had mainly been volunteer driven, to communities that attracted organizations and commercial interests (Wagstrom 2009). The concept of OSS 2.0 introduced by Fitzgerald (2006) captured the transformation that led to FLOSS becoming a mainstream, commercially viable form of software development. With organizations entering the FLOSS arena, researchers began to study the benefits that organizations could accrue by participating in FLOSS projects (Stewart, Ammeter, et al. 2006). Over time, the focus of research shifted towards understanding the new governance and control mechanisms that organizations needed to orchestrate in FLOSS projects. For example, O'Mahony's (2005) research on the Linux open source community provides an understanding of the governance mechanisms that emerge when organizations enter the mix of open source communities. Dahlander and Magnusson's (2005) study of four Nordic FLOSS organizations found that organizations adopt different approaches for handling organization–community relationships and that, depending on the type of approach they adopt, organizations face different managerial challenges and operational means of exerting control. Wagstrom (2009) examined the Eclipse open source project to provide a detailed account of how FLOSS communities with multiple organizations develop effective governance structures to facilitate the interaction between the different organizations and the individual contributors in the community. Capra et al. (2011) described organizational involvement in terms of not only providing developmental support by involving employees but also in terms of the nondevelopment support the organization provides and the management and coordination practices that it brings in. From these studies, it is clear that organizations introduce unique governance and project management mechanisms when they own FLOSS projects. These findings leads us to question how the network governance mechanisms that emerge to sustain collaboration in communities of practice (Jones et al. 1997; Sagers 2004) transform when organizations own FLOSS projects and introduce more formal governance and project management practices (Capra et al. 2011). Driven by the need to understand this influence, our research tries to examine the moderating influence of organizational ownership and the project management practices that it brings on the relationship between team composition and project survival.

Theory and Hypotheses

Relationship between Team Composition and the Survival of the Project

To study the impact of team composition on the survival of the project, we analyze the process of building code in FLOSS environments, using process as the focus of our analysis. Coordination Theory (CT; Malone and Crowston 1994) offers a good approach for analyzing processes and can help us understand how changes to team composition can create dependencies resulting in new coordination challenges which in turn can influence the survival of the project. According to coordination theory, actors in organizations face coordination challenges arising from the dependencies that constrain the ways in which tasks can be performed. These coordination challenges stem from dependencies that emerge between tasks (which includes goals and activities) and resources used or created by tasks (which includes the effort of the actors) (Crowston 1997). CT identifies three types of dependencies that lead to coordination challenges: (a) task-task dependencies emerge when tasks share a common input or output or when the output of one task is the input to another, (b) task-resource dependencies emerge when a task requires a resource for completion, and (c) resource – resource dependencies emerge when one resource depends on another (Crowston 1997).

To overcome these coordination challenges, actors must perform additional activities, referred to as coordination mechanisms (Crowston 1997). The theory of network governance (Jones et al. 1997) provides the theoretical framework to understand the social mechanisms that emerge in communities of practice in-order to overcome coordination challenges and safeguard interactions between actors in the community (Sagers 2004; Wasko et al. 2004). Building on CT and the theory of network governance, we contend that as the number of core contributors increase, new coordination challenges emerge because of two factors (a) an increase in task-task dependencies (b) variance in the core contributors' expectations, skills and goals. Each of these factors are detailed below.

Increased task level dependencies: Because the core developers contribute a significant portion of the code and have the autonomy to make changes directly to the main project code (Setia et al. 2012), an increase in the number of core contributors results in an increase in the task level dependencies in the project. These task level dependencies manifests as two challenges - (i) challenges associated with effective prioritization and labeling of tasks (task triaging), and (ii) challenges associated with managing task conflicts. Triaging of work and identifying the priorities to process bug reports is an important element in FLOSS projects which can crucially affect product quality, project reputation, user motivation and thus the long-term success of a project (Zanetti et al. 2013). One part of the triage process is the assignment of a task to a developer with the appropriate expertise. As the core contributors of a project become numerous and distributed, finding a contributor with a specific expertise can become difficult (Matter et al. 2009). This problem is exacerbated by the fact that contributors are often sporadically engaged and work on multiple different projects simultaneously (Howison and Crowston 2014). Triaging work therefore becomes more time consuming and risky as the core contributors increase (Matter et al. 2009).

Task conflicts not only arise as the number of task contributions increase but also due to different opinions about how a specific task should be performed (Jehn and Mannix 2001). While task conflicts can be beneficial in small numbers they can lead to dysfunctionalities and failures in group performance as they increase (van Wendel de Joode 2004). Because of issues related to task triaging and conflicts that increase as the number of core contributors increase, we posit that coordination challenges related to task level dependencies increase as the number of core contributors increase.

Variance in the core contributors' expectation, skills and goals: Being a community of practice, FLOSS projects can be conceptualized as requiring network governance and informal social mechanisms to coordinate tasks and safeguard exchanges (Sagers 2004; Wasko et al. 2004). Accordingly, allowing a greater access to the main project code (by increasing the number of core contributors) can increase the variance in expectations, skills, and goals that core contributors bring to the project (Jones et al. 1997). This variance can manifest as conflicts and/or misunderstandings between core contributors which may have a negative influence the survival of the community of practice. Presence of these conflicts in FLOSS projects were highlighted by van Wendel de Joode (2004), in his interviews of FLOSS project contributors. In these interviews, a programmer from the OpenOffice community indicated how cultural difference between important contributors would escalate into conflicts and misunderstandings: "People who have different cultural or language skills cause problems. Native speakers would understand things that non-native didn't understand and they would get pissed off and became counter-productive." (van Wendel de Joode 2004; p 107). In contrast, when the number of core contributors is small, it can enable continued interactions between core contributors and may permit exchange partners to learn each other's systems to develop communication protocols, and to establish routines for working together — all of which enhance coordination (Jones et al. 1997; Wasko et al. 2004). In addition, since the core contributors have the autonomy to make changes to the project: change licensing and governance policies and influence the direction of the project —the project may be exposed to opportunistic behavior exhibited by core contributors. To reduce the variance in expectations and limit the projects' exposure to opportunistic behavior, it is in the interest of the project to have fewer core contributors who interact more often so that they develop stronger ties between each-other and ensure that their interests and needs are aligned to that of the project (Jones et al. 1997; Wasko et al. 2004).

The challenges that emerge from the aforementioned factors can prove risky for the project and when not controlled, can influence the survival of the project negatively. Sagers (2004), in his survey-based study of FLOSS projects was one of the first to find that access restrictions to the development code can have a positive influence on the success of the project, giving evidence to the presence of network governance mechanism in FLOSS projects. It is not just the core contributors who influence the survival of the project but also the contributors at the periphery, who through their large numbers can ensure higher overall activity in the project and provide the workforce to complete new changes/patches initiated by the core contributors (Rullani and Haefliger 2013). The importance of peripheral contributors for a project has been well established by Setia et al. (2012), who found that the number of peripheral contributors in a project can not only have a positively influence on the awareness and adoption of the FLOSS product but can also help during the development of the code by testing the code and identifying bugs. Therefore, while core contributors enable the design and evolution of the project by determining new changes and fixes, it is the sub-unit comprising core and peripheral contributors that works together to accomplish the envisioned changes and adds functionality to the project. Based on this understanding, we can say that the team composition of the FLOSS project measured in terms of the proportion of core contributors influences the survival of the project. As the proportion of core contributors increases in a FLOSS project, there is – (1) a higher burden of coordination because of task level dependencies and variance in core contributors' expectations, skills and goals, and (2) a lesser number of peripheral developers per sub-unit resulting in a smaller chance that the envisioned changes and fixes is accomplished and accurately tested. These challenges can prove risky for the project and reduce the probability of survival of the project. Hence, we hypothesize that an increase in the proportion of core contributors negatively influences the survival of the project:

> *Hypothesis 1: A greater proportion of core contributors in a project will lead to a lower chance of survival of the project*

Moderating Influence of Organizational Ownership on the Relationship between Team Composition and Project Survival

When organizations own FLOSS projects, they bring in strategic planning initiatives and introduce project management practices for the efficient coordination and management of development activities (Fitzgerald 2006). Using the bazaar metaphor (Raymond 1999)[1], it can be said that the organizational ownership of FLOSS projects leads to the introduction of management practices transforming the adopted model of development from a bazaar to a cathedral-like form. On the other hand, individual owned projects tend to retain the bazaar model of FLOSS development, with less formal management practices as compared to organization owned FLOSS projects (Medappa and Srivastava 2016). The introduction of management practices when organizations own FLOSS projects have been empirically observed. For example, a study of 83 Eclipse projects found that FLOSS projects initiated by organizations employed both leadership and resource deployment control, as compared to projects that are initiated by individuals belonging to the FLOSS community (Schaarschmidt et al. 2015). Through these management practices, organizational ownership promotes mechanisms for coordinating and managing FLOSS development activities. These activities play an important role in mitigating the challenges emerging from the increase in proportion of core contributors (Crowston 1997). The mechanisms through which these challenges are overcome can be classified as those that help manage task level dependencies and those that help manage variance in the core contributors' expectations, skills and goals.

Mechanisms for managing task level dependencies: Organizations often participate in FLOSS projects by identifying new requirements and functionalities, planning and designing the application, and coordinating development activities (Capra et al. 2011). Further, the organization owner tends to implement release management practices, easing the process of creation of distribution packages and prioritizing the functionalities for release to end user (Capra et al. 2011). These findings have been observed empirically in the GNOME open source project, where the GNOME Foundation acts as a centralized release coordination authority, creating release schedules and coordinating modules for release (O'Mahony 2005). At the task level, the aforementioned activities act as coordination mechanisms that can help resolve task level dependencies, helping overcome task conflicts and allowing effective triaging of tasks. Hence, when organizations own FLOSS projects, the project management and coordination activities that they bring to the project can act as mechanisms for mitigating the challenges that emerge because of task level dependencies.

Mechanisms for managing variance in the core contributors' expectation, skills and goals: As the number of core contributors increases, there is a greater variance in their expectations, skills and goals leading to difficulties in coordinating their activities and ensuring there is a common direction to the project (Sagers 2004). The difficulties in managing their expectations and coordinating their activities are complicated by the distributed nature of the community, where, contributors are spread across the globe and are unlikely to meet each other. We contend that, organizational ownership of FLOSS projects can help mitigate this challenge, by establishing a macroculture between the contributors. The macroculture creates a system of widely shared assumptions and values, comprising organization-specific, occupational, or

---

[1] Eric S. Raymond is credited with coining the terms cathedral and bazaar to describe software development work that occurs through centralized and distributed efforts respectively. In the bazaar model of development, code development occurs over the internet with many people tinkering with the source code without central control.

professional knowledge, that guide actions and create typical behavior patterns among independent contributors (Jones et al. 1997; Wasko et al. 2004). The specificities of the macroculture emerge from the fundamental assumptions about user and organizational needs, existing projects, and the FLOSS community distilled across time from several projects and contributors (Gordon 1991). Within this macroculture, organizations often establish best practices and guidelines specifying roles, role relationships, and conventions to be employed by participants (Jones et al. 1997). The existence of the macroculture allows core contributors to have a better understanding of the culture of the community and can allow them to a priori align their expectations to that of the community. Because of these influences, we hypothesize that organization ownership mitigates the negative influence that the increase in core contributors has on the survival of the project.

Hypothesis 2: Organizational ownership mitigates the negative influence that the proportion of core contributors has on project survival

The different coordination mechanisms, communication protocols, project management activities, that organizations bring to the FLOSS project creates new governance related activities in the project (Fitzgerald 2006). These governance related activities include suggesting requirements and functionalities to be added to the software, planning and designing the application, or simply coordinating development (Capra et al. 2011). Core contributors are often assigned these governance activities, shifting their nature of work from pure code contribution to governance of the project (Rullani and Haefliger 2013). For example, in the Android open source project, experienced contributors take on the role of approvers and project leaders who not only participate in the code-review process but also lead all technical aspects of the project, including the project roadmap, development, release cycles, versioning, and quality assurance [2]. Considering these activities, core contributors are found to work across multiple roles, and are crucial to their teams' organizational, intra-personal and interpersonal processes (Licorish and MacDonell 2013). This can lead to a decrease in the number of code contributions from core contributors in organization owned projects as they devote some of their time for the governance and coordination activities of the project. Hence, we hypothesize,

Hypothesis 3: The average code contributions per core contributor decreases in the case of organization owned project as compared to individual owned projects.

Methodology

To understand the mechanisms through which team composition influences FLOSS projects, we conducted an empirical analysis of FLOSS projects hosted on GitHub. GitHub's popularity among programmers, its developer-focused environment, its integrated social features, and the availability of detailed metadata makes it a popular environment for FLOSS research (Kalliamvakou et al. 2014). Our adopted methodology comprised three steps: data collection, measurement, and analysis. In the data collection step, we collected detailed project log data of all FLOSS projects started in early 2014 from the GH Archive database (https://www.gharchive.org/). In the measurement step, using the collected project log data, we measured the dependent, independent, and control variables. Finally, in the analysis step, we carried out survival analysis (Hosmer et al. 2008) at the project level to test hypotheses 1 and 2 and adopted a Hierarchical Linear Modeling (HLM) approach (Raudenbush and Bryk 2002) for testing hypothesis 3. By improving the data collection, measurement, and analysis methods over the course of several months, we tried to ensure that the methodology we adopted was exhaustive and robust.

Data Collection

We employed Google's bigquery tool to query the archived project log data available in the GH Archive database. Because this database is large (441 GB, with 134 million rows for the year 2014; 488 GB, with 212 million rows for the year 2015), we needed the bandwidth provided by a tool like Google's bigquery to run queries and export the results. We restricted our analysis to all projects that were started during the first five months of 2014 to reduce the number and size of queries. We analyzed the events of the project for the

[2] https://source.android.com/setup/start/roles

first two years of its lifecycle (years 2014 and 2015). In all, we ran more than 60,000 queries over a period of 40 days.

While GitHub provides a rich dataset, care needs to be taken to overcome common perils in using this dataset (Kalliamvakou et al. 2014). For example, projects that do not involve software development, are too small, or are mirrors/personal stores should be avoided. After filtering the dataset to address the perils[3], we were left with a sample of 6431 FLOSS projects, of which 3027 are individual-owned and 3404 are organization-owned that we finally considered for our analysis.

Measurement

GitHub offers a good environment for measuring the proportion of core contributors and studying its relationship to project's survival. More specifically, two features of GitHub make it ideal for this research. First, the granularity of the data and the availability of time stamps and contributor information for all events enabled us to clearly identify the contributors and their contributions, which allowed us to operationalize the proportion of core contributors. Second, the availability of detailed contributor- and project-specific data allowed us to measure the dependent variable and operationalize the theorized control variables. The different variables that we used in this research and their measures are detailed in the following subsections.

Dependent variables

As mentioned in the section "FLOSS Project Success," we consider the survival or sustenance of the project to be an important measure of FLOSS project success which is not yet well understood. To understand what determines the survival of the project and test hypotheses 1 and 2, we conduct survival analysis using Cox proportional hazard model (Hosmer et al. 2008). In this model, we consider the event to have happened (project failure) if the project became inactive within the first two years of its inception. A project was deemed inactive if no code changes were made on the source code after the year 2015. The choice of this timeline for project inactivity is in line with Stewart, Darcy, et al. (2006), who found that a significant proportion of FLOSS projects cease to have activity after the first year. In survival analysis, we estimate the likelihood that a project becomes inactive at time *t* by calculating the hazard function (described in the Model and Results section of this paper).

To test hypothesis 3, we adopted the *number of code contributions of the contributor* as the dependent variable of interest. To measure this construct, we summed up all the contributions made by each contributor in terms of commits added to the main project code during the years 2014 and 2015. Note that while hypotheses 1 and 2 are tested at the project level, hypothesis 3 is tested at the contributor level using HLM (Raudenbush and Bryk 2002).

Independent variables

To study the influences of the team composition and ownership on the survival of a FLOSS project, we employed two independent variables (a) proportion of core contributors and, (b) organization owner flag. The *proportion of core contributors* was measured as the ratio of the total number of *core* contributors to the total number of contributors in the project. To identify the core contributors in a project we studied the nature of contributions made by each contributor to identify those contributors who have write access to the project (Rullani and Haefliger 2013). If a contributor has written directly to the main project code[4] at any point during the years 2014 and 2015, we identified that contributor as a core contributor. Table 1 provides the means and standard deviations of the contributions made by the core and peripheral contributors in our sample of 6,431 projects. From table 1, we note that the mean contributions of the core contributor is much higher than that of the peripheral contributor (Crowston, Wei, et al. 2006).

[3] The complete list of perils identified by Kalliamvakou et al. (2014) and a summary of the methods we adopted to address these can be provided on request.

[4] Only contributors who are given write access can implement a "push" event on a project in GitHub. By studying all the push events made on the project in the years 2014 and 2015, we identified the contributors who have write access to the project.

	Number of Contributors	Mean Contributions	Std. Deviation Contributions
Core contributors	22,930	77.10	202.87
Peripheral contributors	61,734	2.34	6.19

Table 1. Mean and Standard Deviations of Contributions Made by Contributors

To study the moderating influence of project ownership on the relationship between the team composition and the success of a FLOSS project, we used a flag, *organization owner flag*, which takes the value 1 if the project is owned by an organization and 0 if it is owned by an individual. Organizations in GitHub are shared accounts that can be used to centralize a group's code and adopt a workflow that is suitable for business (Neath 2010). This workflow provides multiple levels of permission controls that enables companies to create nested teams with hierarchical access to the code, allowing them to replicate their organization structure on GitHub. Most companies hosting projects on GitHub, use their own organization accounts to consolidate monitoring and management of their FLOSS projects. GitHub recognizes projects that are owned by organizations' and makes this attribute publicly available through its API. By accessing this GitHub determined project attribute, we identified if a project is owned by an organization or an individual.

Control variables

We controlled for two kinds of variables in our analysis: contributor/owner characteristics and project characteristics.

<u>Contributor/owner characteristics:</u> We identified two measures to control the influence of contributor and project owner characteristics on the relationship between the proportion of core contributors and the survival of a project. First, we controlled for the *experience of the project owner* in terms of the number of GitHub FLOSS projects the owner has created. The existence and density of prior ties between the project owner and contributors has been found to positively influence the probability that the project will attract more individuals (Rebeca and García 2009). Based on this finding, we would expect that the experience of the project owner plays a role in enhancing the popularity of the project and its chance of survival. Second, an increase in the *number of contributors* is often associated with an increase in the number of task contributions and consequently an increase in the success of the project (Stewart, Ammeter, et al. 2006; Subramaniam et al. 2009). To control these influences, we included the fixed effects for the number of contributors by creating a dummy variable for each value of number of contributors in the project.

<u>Project characteristics:</u> We identified seven measures to control for different project characteristics. First, we controlled for project popularity measured in terms of the *number of stars* that the project has received. Project popularity has been found to be correlated to the number of bugs, the number of participants in the bug trackers (Crowston, Howison, et al. 2006), and the number of contributors (Krishnamurthy 2005). Further, popular projects attract a larger community of users who not only identify bugs during use but also lead to improved ideas (Subramaniam et al. 2009). Because popularity can influence both the survival and the number of contributors, we controlled for the effect of popularity in our model. Second, we controlled for the number of forks that have been created from the project. The distributed nature of GitHub environment enables a pull-based development model, where changes are offered to a project repository through a network of project forks (Gousios et al. 2014). Hence the number of forks can determine how distributed and complex the development work is and can influence the number of core contributors required to effectively manage the project. Third, we controlled for the *average task size*, measured as the average number of commits made within the project's pull-request events. As the tasks within a project increase in size and complexity, the average number of commits per task increases. It is important to control for the average task size in the project because projects with a greater number of small tasks may require different coordination mechanisms than projects with small number of large tasks. Fourth, we controlled for *project size* measured as megabytes of code. Smaller projects are usually associated with fewer contributors and contributions than are larger projects. Thus, projects of different sizes are expected to differ in terms of their capacity for attracting contributors and coordinating new tasks (Setia et al. 2012). Fifth, we controlled for the *number of programming languages*. While projects that involve multiple languages may have greater functionality, coordinating contributions across different languages can be challenging. Sixth, we controlled for the type of license attributed to the project. We classify licenses as being either restrictive or permissive. FLOSS software that adopts a restrictive license follows a "copyleft"

policy that forces any enhancement made to the software to be bound by the same or similar (restrictive) license (Medappa and Srivastava 2017). Permissive licenses, on the other hand, do not have the copyleft feature and allow contributors to use open source software to build proprietary or "closed" software. Previous studies of the impact of the FLOSS license type have found that the type of license used influences a project's popularity and the number and productivity of contributors (Medappa and Srivastava 2017; Stewart, Ammeter, et al. 2006). Hence, we used the flag *restrictive license flag* to control the effect of the type of license (restrictive vs. permissive) on the success of a project. Lastly, we include the fixed effects of the *main programming language*. Despite mixed results in the extant literature, the type of programming language used has been found to be an important antecedent to FLOSS project success (Subramaniam et al. 2009). Further, the ease of coordinating development activities may differ across programming languages, with languages more conducive to modular architecture displaying greater ease of coordination (Baldwin and von Hippel 2011). We controlled for these effects by including a dummy variable for each programming language used in our sample.

Analysis

We included two regression models in our analysis. The regression model used to test hypotheses 1 and 2 is based on survival analysis (Hosmer et al. 2008). The second model, based on the HLM approach, is used to test hypothesis 3 (Raudenbush and Bryk 2002). Hypotheses 1 and 2 predict that the proportion of core contributors in a project has a negative relationship with the survival of a project which is moderated by the ownership of the project. To test these hypotheses, we included the proportion of core contributors and its interaction term with the organization owner flag in the Cox proportional hazard model. Hypothesis 3 predicts that the type of ownership of a project influences the number of contributions made by its core contributors. Because of the nested nature of the data, where, individual contributors are nested within projects, we adopt HLM with number of contributions from the contributor as the dependent variable of interest. The following section details the results of the regression models and the checks we employed to ensure the validity of the results.

Model and Results

Table 2 provides the means, standard deviations, and correlation coefficients for the variables used in the analyses. From Table 2, we can observe that the proportion of core contributors is negatively correlated with project inactivity flag (r = 0.20, p < .01). This negative correlation provides some indication of the potentially negative relationship between the proportion of core contributors and project survival. We also recognize a strong positive skew in the variables *—stars, forks, average task complexity* and *size of project*. In order to normalize the positively skewed data, statistical texts commonly recommend the use of log transformations (Carte and Russel 2003). After log transformation of these variables, their residuals were found to closely match a normal distribution.

Table 1. Mean, Standard Deviation, and Pairwise Correlation Coefficients of the Variables

	Variable	Mean	Std.	Min.	Max.	1	2	3	4	5	6	7	8	9	10
1	Stars	361.09	1216.60	1	24645										
2	Forks	89.13	421.96	0	17198	0.51**									
3	Average task complexity	2.51	14.95	1	1135	-0.01	0.00								
4	No of programming languages	3.17	3.00	1	87	0.04**	0.05**	0.02							
5	Size of project in megabytes	2.16	22.27	0.0001	1175	0.00	0.01	0.01	0.27**						
6	Owner experience	67.24	134.32	1	1721	0.05**	0.02	-0.01	-0.05**	-0.02					
7	Restrictive license regime	0.11	0.32	0	1	-0.03*	0.01	0.01	0.14**	0.07**	-0.09**				
8	Organization owner flag	0.53	0.50	0	1	-0.04**	0.00	0.01	0.14**	0.05**	0.04**	0.01			
9	Number contributors	12.90	26.30	2	1632	0.34**	0.66**	-0.01	0.07**	0.02	0.04**	0.00	0.06**		
10	Proportion core contributors	0.39	0.29	0.001	1	-0.23**	-0.15**	0.04**	0.12**	0.03*	0.00	0.05**	0.42**	-0.2**	
11	Project inactivity flag (2 Year)	0.26	0.44	0	1	-0.13**	-0.09**	-0.01	-0.07**	-0.01	0.01	-0.02	-0.05**	-0.12**	0.20**

* p<0.05; ** p<0.01

Table 2. Mean, Standard Deviation, and Pairwise Correlation Coefficients of the Variables

Hypotheses Linking the Proportion of Core Contributors and Project Survival

Hypothesis 1 predicts a negative relationship between the proportion of core contributors and the survival of a FLOSS project. Hypothesis 2 predicts that this negative relationship is mitigated when organizations own FLOSS projects. We use the semi-parametric Cox proportional hazard regression model for the survival analysis, since it relaxes the specification requirement for the baseline hazard function. The Cox model does not depend on distributional assumptions of survival time, provides flexibility for time dependent explanatory variables, and allows the identification of hazard ratio as the relative risk of project inactivity given that the project is at a given period in its life (Hosmer et al. 2008).

In this model the project duration is assumed to have a continuous distribution function $f(t)$. The probability that the duration of the project will be less than t:

$$F(s) = Prob(T \leq t) = \int_0^t f(s)ds$$

The survival function $S(t)$ is defined as the probability that the duration is greater than t:

$$S(t) = Prob(T > t) = 1 - F(t)$$

The likelihood of project inactivity at time t given that it has lasted till time t, is given by the hazard rate:

$$h(t) = \frac{f(t)}{S(t)}$$

In Cox proportional hazard models, we model the hazard rate as an exponential function of the predictors:

$$h(t|x,\beta) = h_0(t)e^{\beta_i x_i}$$

Where, $x_i$denotes the i^{th} predictor and $\beta_i$denotes the i^{th} coefficient. $h_0(t)$ is the baseline hazard function, which corresponds to the situation where all predictor variables are zero (Samoladas et al. 2010).

Table 3 provides the coefficients estimated using the Cox proportional hazard models. Hypothesis 1 predicts that the proportion of core contributors has a negative influence on the survival of the project (positive influence on the likelihood of project inactivity). From model 1a (Table 3), we find that the proportion of core contributors does not significantly influence the likelihood of project inactivity (β_1 = 0.090, p > .10). Hypothesis 2 on the other hand predicts that in the case of organization owned FLOSS projects the negative effect of the proportion of core contributors on the survival of the project is mitigated. From model 1b (Table 3), we find that the interaction term is negative and significant (β_2 = -0.863, p < .01). This indicates that the influence of proportion of core contributors on the likelihood of project inactivity is significantly less for organization owned projects as compared to individual owned projects.

Deeper analysis of model 1b reveals a crossover interaction of ownership type, which tends to shift the relationship between proportion of core contributors and likelihood of project inactivity from positive for individual owned projects to negative for organization owned projects; i.e. $|\beta_1| < |\beta_2|$. The crossover interaction effect is the likely cause for the insignificance seen in the direct effect (hypothesis 1). This suggests that the nature of relationship between the core contributors and likelihood of project inactivity is conditional on the type of ownership of the project. We explore the implications of this interesting finding in the discussion section.

When modeling a Cox proportional hazard model a key assumption is proportional hazards (Hosmer et al. 2008). In order to check if this assumption is satisfied, we test if the hazard functions of survival curves at two different strata are different (Samoladas et al. 2010). The result of our test rejects the null hypothesis that the hazard functions are not proportional (Prob>chi2: 0.299), providing support for the proportionality assumption. Further, to correct for any potential heteroscedasticity in the error terms, we used heteroscedasticity consistent standard errors in all of our models (Hayes and Cai 2007).

Survival Analysis: Cox-Proportional Hazard Model	DV : Likelihood of Project Inactivity	
	Model 1a	Model 1b
Log(number of stars)	-0.193 (0.00)**	-0.18 (0.00)**
Log(number of forks)	-0.474 (0.00)**	-0.489 (0.00)**
Log(avg. size of tasks)	0.056 -0.16	0.049 -0.23
Log(size of project)	-0.057 (0.00)**	-0.062 (0.00)**
Number of languages	-0.015 -0.27	-0.013 -0.34
Owner experience	0 -0.26	0 -0.26
Restrictive license flag	-0.047 -0.6	-0.047 -0.6
Organization owner flag	-0.284 (0.00)**	0.123 -0.29
Proportion of core contributors (ß$_1$)	0.092 -0.49	0.582 (0.00)**
Proportion of core contributors X Organization owner flag (ß$_2$)		-0.863 (0.00)**
Fixed effect of number of contributors	YES	YES
Fixed effect of main programming language	YES	YES
N	6431	6431

* p<0.05; ** p<0.01

Table 3. Results of Survival Analysis

HLM Analysis	DV : Contributions of the contributor
	Model 2a
Log(number of stars)	0.14 0.74
Log(number of forks)	0.317 0.637
Log(avg. size of tasks)	0.664 0.35
Log(size of project)	0.894 (0.00)**
Number of languages	0.319 0.051
Owner experience	0 0.766
Restrictive license flag	0.514 0.708
Organization owner flag	-0.96 0.262
Core contributor flag (ß$_1$)	119.74 (0.00)**
Core contributor flag X Organization owner flag (ß$_2$)	-46.123 (0.00)**
Fixed effect of number of contributors	YES
Fixed effect of main programming language	YES
Number of groups	6431
Number of observations	83307

* p<0.05; ** p<0.01

Table 4. Results of HLM Analysis

Hypotheses Linking Organization Ownership and Number of Code Contributions of the Core Contributor

Hypothesis 3 predicts that the average number of code contributions per core contributor decreases when organizations own FLOSS projects. We used HLM to test the relationship because of the existence of two hierarchical levels of analysis i.e. contributors nested within projects (Raudenbush and Bryk 2002). Because of the hierarchical nature of the contributor level data, we expect that the influence of ownership type on the number of contributions per contributor may be different for different projects leading to aggregation bias, misestimated precision, and the “unit of analysis” problem (Setia et al. 2012). To account for this heterogeneity and to offer a more robust model for the hierarchical data, we adopt HLM to test hypothesis 3.

Table 4, provides the results of the HLM model we employed. Hypothesis 3 predicts that organization ownership has a negative influence on the number of code contributions made by the core contributors. From model 2a (Table 4), we find that the organization ownership tends to decrease the total number of code contributions made by the core contributor ($ß_2$ = -46.123, p<.01) providing support for hypothesis 3.

Discussion and Conclusion

Despite the FLOSS development model’s increasing prominence in practice and research, the antecedents to its success have not been completely understood. In specific, while extant research has well established the significance of network governance mechanisms in managing distributed work (Sagers 2004; Wasko et al. 2004), the socio technical influence of source code access restrictions calls for a stronger theoretical enquiry. As a step forward in this direction, our research sought to unearth the mechanisms through which

FLOSS team composition (measured as the proportion of contributors who are given write access to the source code) influences the survival of individual- and organization-owned FLOSS Projects.

Based on the survival analysis of a large sample of FLOSS projects owned by individuals and a wide range of organizations, we find that the proportion of contributors who are given write access to the source code exhibit opposing effects on project survival, which is conditional on the ownership of the project. Surprisingly, we find that the expected negative relationship between proportion of core contributors and survival of the projects does not hold for organization owned FLOSS projects. A deeper analysis of the hazard ratio for the sub-group of individual owned projects shows that as the proportion of core contributors increases from 0 to 1, the hazard rates increases by 0.96 ($p < .01$). Keeping everything else constant, these figures translates to a 21% increased chance that a project will become inactive for one standard deviation increase in the proportion of core contributors. In contrast, for the sub-group of organization owned projects, as the proportion of core contributors increases from 0 to 1, the hazard ratio decreases by 0.34 ($p < .01$). This translates to a 10% decreased chance that a project becomes inactive for one standard deviation increase in proportion of core contributors in the case of organization owned FLOSS projects. The decreased likelihood of project inactivity for the sub-group of organization owned projects is theorized to be an outcome of the interplay between network governance mechanisms and project management and control practices of the organization owner. Furthermore, we find that the introduction of project management and control practices is found to shift the nature of work for core contributors away from pure code contributions to more governance related activities resulting in decreased number of code contributions made by core contributors.

Implications

Our research contributes to IS and organization theory in three ways. First, our study advances the existing literature on contributor roles in FLOSS projects (Rullani and Haefliger 2013; Sagers 2004; Setia et al. 2012), as it unearths the complex role of access restrictions in mitigating coordination challenges and influencing project survival. From the standpoint of coordination theories (Malone and Crowston 1994) and the theory of network governance (Jones et al. 1997; Wasko et al. 2004), we highlight the importance of considering the interplay between network governance mechanisms and project management practices brought in by the owners. These considerations are necessary to fully understand the influence of network governance mechanisms on the success of projects.

Second, our research advances the literature surrounding organizational participation in FLOSS projects (Stewart et al. 2006; Wagstrom 2009; Capra et al. 2011) by delineating the moderating mechanisms brought forth by organizational ownership in FLOSS projects. The debate regarding openness vs. control in FLOSS environments has received considerable attention because finding the right balance between the two can influence the success of FLOSS projects (Teigland et al. 2014). Our inquiry adds an interesting dimension to this debate by theorizing that control (through project management practices) and openness (by providing access to the source code) can complement each other under certain conditions. That is, based on our findings, we can say that project management practices introduced by organizations can help enhance the openness of the projects by lowering the need for informal coordination mechanisms like access restrictions and collective sanctions (Wasko et al. 2004). Organizations have been tempted to adopt the FLOSS model of development and tap into the vast reserves of programming skills spread across the globe. Despite these intentions, organizations are still unsure of how the management and coordination practices that they have developed in-house can complement the FLOSS model of development. This research tries to shed some light on this aspect of FLOSS projects and informs project owners about the usefulness of establishing formal management and coordination mechanisms before considering the adoption of a less restrictive and open collaborative development environment.

Lastly, survival and sustenance of projects is an important, yet underdeveloped dimension of FLOSS project success. FLOSS environments are synonymous with limited amount of time pressures and project milestones that allow the emergence of unique structures of work (Howison and Crowston 2014). Given the limited time pressures, successful FLOSS projects are expected to sustain participation and remain active for prolonged periods of time with frequent software releases. Our research sheds light on this dimension of success by unearthing the coordination risks that emerge from changes to the team composition and its potential impact on the survival of the FLOSS projects.

Limitations

Although we have tried our best to ensure theoretical and methodological rigor in this research, there are certain limitations that needs to be considered. First, our operationalization of core and periphery was done so that we could study the influence of access restrictions on the survival of the project. To identify the contributors who were given source code access, we studied the project logs for two years and identified the contributors who had made direct changes to the source code during this time (using push events). However, this operationalization misses other role classifications (like project leader, maintainer) and more complex team hierarchies. Further, this operationalization does not consider the number of code contributions or the density of communication ties to differentiate core and peripheral contributors. Understanding the aforementioned two aspects of contributor roles in relation to access restrictions may be necessary to fully understand the phenomenon. Second, our choice of GitHub platform and the duration of study raises the question —can the results be replicated across other platforms and time intervals? Our choice of GitHub was based on its popularity among programmers, its integrated social features, and the availability of detailed metadata. However, future research is necessary to confirm if the mechanisms that are seen to operate in GitHub would exist in platforms that adopt different work flows. Regarding the timing of the study, we considered FLOSS projects that were started in the first five months of 2014 and studied the events added to it for a duration of two years (i.e. till the end of 2015). Projects that did not have any source code additions made after the year 2015 were deemed as inactive. However, we may have missed out on the temporal effects which may be crucial to understand how the influence of the team composition changes over time. A deeper temporal study can enhance our understanding about the impact that the life-cycle stage of a FLOSS project has on its team composition and how the impact may differ in large, mature projects like Eclipse, Firefox, and GNOME. Notwithstanding these limitations, our study contributes to the academic understanding of the influence of restricting access to the code in FLOSS development and the mechanisms through which team composition influences FLOSS project survival in individual as well as organizational contexts.

References

Baldwin, C., and von Hippel, E. 2011. "Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation," *Organization Science* (22:6), pp. 1399–1417.

Capra, E., Francalanci, C., Merlo, F., and Rossi-lamastra, C. 2011. "Firms ' Involvement in Open Source Projects : A Trade-off between Software Structural Quality and Popularity," *The Journal of Systems and Software* (84), pp. 144–161.

Carte, T. A., and Russel, C. J. 2003. "In Pursuit of Moderation: Nine Common Errors and Their Solutions," *MIS Quarterly* (27:3), pp. 479–501.

Chengalur-smith, I., Sidorova, A., and Daniel, S. 2010. "Sustainability of Free / Libre Open Source Projects : A Longitudinal Study Sustainability of Free / Libre Open Source Projects : A Longitudinal Study," *Journal of the Association for Information Systems* (11), pp. 657–683.

Coverity Inc. 2013. "Coverity Scan: 2013 Open Source Report," pp. 1–23. Accessed January 4, 2017, http://softwareintegrity.coverity.com/register-for-scan-report-2013.html?cs=pr.

Crowston, K. 1997. "A Coordination Theory Approach Organizational Process Design," *Organization Science* (8:2), pp. 157–175.

Crowston, K., Howison, J., and Annabi, H. 2006. "Information Systems Success in Free and Open Source Software Development: Theory and Measures," *Software Process Improvement and Practice* (11:2), pp. 123–148.

Crowston, K., Wei, K., Howison, J., and Wiggins, A. 2012. "Free/Libre Open-Source Software Development," *ACM Computing Surveys* (44:2), pp. 1–35.

Crowston, K., Wei, K., Li, Q., and Howison, J. 2006. "Core and Periphery in Free / Libre and Open Source Software Team Communications," *Institute for SOftware Research*, pp. 1–7.

Dahlander, L., and Magnusson, M. G. 2005. "Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms," *Research Policy* (34:4), pp. 481–493.

Fershtman, C., and Gandal, N. 2011. "Direct and Indirect Knowledge Spillovers : The ' Social Network ' of Open-Source Projects," *Rand Journal of Economics* (42:1), pp. 70–91.

Fitzgerald, B. 2006. "The Transformation of Open Source Software," *MIS Quarterly* (30:3), p. 587.

Gamalielsson, J., and Lundell, B. 2014. "Sustainability of Open Source Software Communities beyond a

Fork: How and Why Has the LibreOffice Project Evolved?," *Journal of Systems and Software* (89:1), Elsevier Inc., pp. 128–145.

Gordon, G. G. 1991. "Industry Determinants of Organizational Culture," *Academy of Management Review*.

Gousios, G., Pinzger, M., and Deursen, A. Van. 2014. "An Exploratory Study of the Pull-Based Software Development Model," *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, pp. 345–355.

Grewal, R., Lilien, G. L., and Girish, M. 2006. "Location , Location , Location : How Network Embeddedness Affects Project Success in Open Source Systems," *Management Science* (52:7), pp. 1043–1056.

Hayes, A. F., and Cai, L. 2007. "Using Heteroskedasticity-Consistent Standard Error Estimators in OLS Regression : An Introduction and Software Implementation," *Behavior Research Methods* (39:4), pp. 709–722.

Hosmer, D. W., Lemeshow, S., and May, S. 2008. "Applied Survival Analysis. Regression Modeling of Time-to-Event Data," *John Wiley & Sons* (Vol. 41), New York, NY: John Wiley & Sons.

Howison, J., and Crowston, K. 2014. "Collaboration through Open Superposition: A Theory of the Open Source Way," *MIS Quarterly* (38:1), pp. 29–50.

Jehn, K. A., and Mannix, E. A. 2001. "The Dynamic Nature of Conflict: A Longitudinal Study of Intragroup Conflict and Group Performance," *Academy of Management Journal* (44:2), pp. 238–251.

Jones, C., Hesterly, W. S., and Borgatti, S. P. 1997. "A General Theory of Network Governance: Exchange Conditions and Social Mechanisms," *Academy of Management Review* (22:4), pp. 911–945.

Kalliamvakou, E., Gousios, G., Singer, L., Blincoe, K., German, D. M., and Damian, D. 2014. "The Promises and Perils of Mining GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, pp. 92–101.

Krishnamurthy, S. 2005. "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects (Originally Published in Volume 7, Number 6, June 2002," *First Monday; Special Issue #2: Open Source — 3 October 2005* (October), pp. 1–12.

von Krogh, G., Haefliger, S., Spaeth, S., and Wallin, M. W. 2012. "Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development," *MIS Quarterly* (36:2), pp. 649–676.

Lee, S.-Y. T., Kim, H.-W., and Gupta, S. 2009. "Measuring Open Source Software Success," *The International Journal of Management Science* (37), pp. 426–438.

Licorish, S. A., and MacDonell, S. G. 2013. "The True Role of Active Communicators," *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering - EASE '13*, p. 228.

Licorish, S. A., and MacDonell, S. G. 2014. "Understanding the Attitudes, Knowledge Sharing Behaviors and Task Performance of Core Developers: A Longitudinal Study," *Information and Software Technology* (56:12), Elsevier B.V., pp. 1578–1596.

Lindberg, A., Berente, N., Gaskin, J., and Lyytinen, K. 2016. "Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project," *Information Systems Research* (December), pp. 1–22.

Lohr, S. 2001. "Some I.B.M. Tools to Be Put in Public Domain.," *New York Times (November 5),* http://www.nytimes.com/2001/11/05/technology/05OPEN.html?pagewanted=all.

Malone, T. W., and Crowston, K. 1994. "The Interdisciplinary Study of Coordination," *ACM Computing* (26:1), pp. 87–119.

Matter, D., Kuhn, A., and Nierstrasz, O. 2009. "Assigning Bug Reports Using a Vocabulary-Based Expertise Model of Developers," *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, MSR 2009* (Msr), pp. 131–140.

Medappa, P. K., and Srivastava, S. C. 2016. "Does the Task Structure of Open Source Projects Matter ? Superposition and Value Creation," *ICIS Proceedings*, pp. 1–10.

Medappa, P. K., and Srivastava, S. C. 2017. "License Choice and the Changing Structures of Work in Organization Owned Open Source Projects," in *SIGMIS CPR, Bangalore*, pp. 117–123.

Midha, V., and Palvia, P. 2012. "Factors Affecting the Success of Open Source Software," *The Journal of Systems & Software* (85:4), Elsevier Inc., pp. 895–905.

Neath, K. 2010. "Introducing Organizations." Accessed March 10, 2018, https://blog.github.com/2010-06-29-introducing-organizations/

O'Mahony, S. 2005. "Nonprofit Foundations and Their Role in Community-Firm Software Collaboration," in *Perspectives on Free and Open Source Software*, B. Fitzgerald, J. Feller, S. A. Hissam, and K. R. Lakhani (eds.), MIT Press, Cambridge, MA, pp. 393–413.

Raudenbush, S. W., and Bryk, A. S. 2002. "Hierarchical Linear Models: Applications and Data Analysis Methods," *Advanced Quantitative Techniques in the Social Sciences 1*.

Raymond, E. S. 1999. "The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary," *O'Reilly*.

Rebeca, M.-D., and García, C. E. 2009. *Returns from Social Capital in Open Source Software Networks*, pp. 277–295.

Rigby, P. C., and Hassan, A. E. 2007. "What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List," *Proceedings - ICSE 2007 Workshops: Fourth International Workshop on Mining Software Repositories, MSR 2007*.

Rullani, F., and Haefliger, S. 2013. "The Periphery on Stage: The Intra-Organizational Dynamics in Online Communities of Creation," *Research Policy* (42:4), Elsevier B.V., pp. 941–953.

Rynes, S. L., Bartunek, J. M., and Daft, R. L. 2001. "Across the Great Divide : Knowledge Creation and Transfer between Practitione ...," *Academy of Management Journal* (44:2), pp. 340–355.

Sagers, G. W. 2004. "The Influence of Network Governance Factors on Success in Open Source Software Development Projects," in *Proceedings of the International Conference in Information Systems*, pp. 427–438.

Samoladas, I., Angelis, L., and Stamelos, I. 2010. "Survival Analysis on the Duration of Open Source Projects," *Information and Software Technology*, pp. 902–922.

Schaarschmidt, M., Gianfranco, W., and Von Kortzfleisch, H. 2015. "How Do Firms Influence Open Source Software Communities? A Framework and Empirical Analysis of Different Governance Modes," *Information and Organization* (25:2), pp. 99–114.

Setia, P., Rajagopalan, B., Sambamurthy, V., and Calantone, R. 2012. "How Peripheral Developers Contribute to Open-Source Software Development," *Information Systems Research* (23:1), pp. 144–163.

Stewart, K. J., Ammeter, A. P., and Maruping, L. M. 2006. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," *Information Systems Research* (17:2), pp. 126–144.

Stewart, K. J., Darcy, D. P., and Daniel, S. L. 2006. "Opportunities and Challenges Applying Functional Data Analysis to the Study of Open Source Software Evolution," *Statistical Science* (21:2), pp. 167–178.

Subramaniam, C., Sen, R., and Nelson, M. L. 2009. "Determinants of Open Source Software Project Success : A Longitudinal Study ☆," *Decision Support Systems* (46:2), Elsevier B.V., pp. 576–585.

Teigland, R., Di Gangi, P. M., Flåten, B. T., Giovacchini, E., and Pastorino, N. 2014. "Balancing on a Tightrope: Managing the Boundaries of a Firm-Sponsored OSS Community and Its Impact on Innovation and Absorptive Capacity," *Information and Organization* (24:1), pp. 25–47.

Wagstrom, P. A. 2009. "Vertical Interaction in Open Software Engineering Communities," Carnegie Mellon University.

Wasko, M. M., Faraj, S., and Teigland, R. 2004. "Collective Action and Knowledge Contribution in Electronic Networks of Practice," *Journal of the Association for Information Systems* (5:11–12), pp. 493–513.

van Wendel de Joode, R. 2004. "Managing Conflicts in Open Source Communities," *Electronic Markets* (14:2), pp. 104–113.

Wenger, E. 2005. "Communities of Practice," *Communities of Practice. A Brief Introduction*.

Zanetti, M. S., Scholtes, I., Tessone, C. J., and Schweitzer, F. 2013. "Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities," *In Proceedings of the International Conference on Software Engineering*, pp. 1032–1041.