THESE DE DOCTORAT

DE

L'UNIVERSITE PARIS-SACLAY

PREPAREE A "HEC PARIS"

ECOLE DOCTORALE N° 578

Sciences de l'homme et de la société (SHS)

Spécialité de doctorat : Sciences de gestion

Par

M. Poonacha MEDAPPA

Essays on Value Creation in the Open Source Phenomenon: Understanding the Influence of Work Structures, Team Composition, and Community Ideologies

Thèse présentée et soutenue à Jouy-en-Josas, le 23 August 2019 :

Composition du Jury :

Madame Ou, Carol	Professeure de Management, TiSEM, Tilburg University	Rapporteur
Monsieur Bala, Hillol	Professeur associé de Systèmes d'information, Kelly School of Business	Rapporteur
Monsieur Li, Xitong	Professeur associé de ISOM, HEC Paris	Examinateur
Monsieur Srivastava, Shirish C.	Professeur de ISOM, HEC Paris	Directeur de thèse

L’université Paris-Saclay n’entend donner aucune approbation ou improbation aux opinions émises dans cette thèse. Ces opinions doivent être considérées comme propres à leur auteur.

Acknowledgement

I am extremely fortunate to have met and been in the company of some amazing people during my PhD. Although they deserve so much more, I would like to offer my sincere gratitude to a few among the many people who have helped me along the way.

First and foremost, I sincerely thank my supervisor Shirish, for seeing the potential in me and for giving me the opportunity to pursue PhD under him at HEC Paris. I cannot underscore enough how important he has been during this long and arduous journey. When I was facing any roadblocks, personal or professional, I always had the comfort of knowing that I had his support and guidance. His patience and teaching have helped me grow as a researcher and as an academician. His enthusiasm for research, dedication to his students, and sacrifice to the academic community never ceases to amaze me. He has and will always be a wonderful mentor and an immense source of inspiration for me.

I would also like to thank my thesis committee members Xitong, Carol, and Hillol for committing their time to read and evaluate my dissertation. I admire them and their work deeply. I am extremely privileged to have their inputs and participation. Several other professors helped me in many ways. I am very grateful to Marie-Hélène who gave me several opportunities to teach and from whom I learned and improved my teaching skills. To Sri for his guidance on several matters and for always keeping his door open for me. To Giada, Fred, and Sylvie who were always welcoming and kind. So much so, that I always felt at home with them and eagerly looked forward to the times I would find their company and learn from them.

Pursuing the PhD came with many challenges. I was privileged to receive the help of many from the PhD office. I specially would like to thank Mélanie, Françoise, and Anne for all their hard work and experience which helped me overcome a wide variety of problems. And to Kristine and Britta who are such wonderful and empathetic leaders of the program.

Leaving my home to come to France would have been challenging if not for all the wonderful friends I made during my PhD. I am glad to have met so many friends with whom I could share the ups and downs of my PhD life. I will cherish the many close moments we shared together –the difficult times, the moments of excitement, and the celebrations. To my wonderful batchmates - Saverio, Kerem, Ana, Emanuel, Romain, and Pekka with whom so many joyous movements were shared. Specially to Saverio – well, what can I say? I am just so glad that he was around when I needed. To Yang and Elena, who were there during the difficult last years of PhD to lend their support and company. And to Martina, who gave me the initial push and strength to start my PhD. You all are such wonderful, wonderful people.

Finally, I thank the two most important people in my life, my loving parents, who gave up so much for me and are continuing to do so. I have the best role models to follow. My goal is to become as good as them.

It is somewhat sad to write the last part of this acknowledgement as it holds some connotation of an “end”. However, I rejoice by the fact that I am still connected to everyone mentioned here and will find new opportunities to remain close to them. Thank you.

Contents

List of Tables

List of Figures

1 Introduction

In the current digitally enabled collaborative environment, Free (Libre) and Open Source Software (FLOSS) projects have become ubiquitous. Founded on the principles of openness and co-operation, FLOSS has over time come to refer to software whose source code is available to the general public for use and modification from its original design. Since the 1980s, which saw the formalization of the concept of FLOSS through the creation of licenses and institutions that protected its principles, open source has been transforming the information technology (IT) organizational landscape. The decade that followed its formalization, saw the first releases of Linux, Apache webserver, and Python; projects which would go on to become the poster children for the FLOSS movement. In the 2000s, motivated by the success of community FLOSS projects, IT organizations started cautiously embracing the phenomenon. In 2001, IBM opened the source code of several of its software tools (estimated at $40 million) to the public domain, creating the Eclipse open source project[1]. The current decade, the 2010s, has seen the phenomenon sustain its exponential growth both in terms of contributors coming together on collaborative development platforms like GitHub, as well as in terms of organizations supporting and taking ownership of FLOSS projects. The importance of the FLOSS model of development for the future of IT organizations became evident in June 2018, when Microsoft announced that it was purchasing GitHub for an estimated $7.5 billion[2].

The transformation of FLOSS from an ideology in the 1980's to a phenomenon that has become central to the strategic decision of IT organizations leads to the question - what does open source have in store for the future? A clue for answering this question comes from Microsoft's

1 http://www.nytimes.com/2001/11/05/technology/05OPEN.html?pagewanted=all

2 https://www.theverge.com/2018/6/4/17422788/microsoft-github-acquisition-official-deal

GitHub acquisition announcement, where they (Microsoft) state the bright future for FLOSS and collaborative development.

> "Computing is becoming embedded in the world, with every part of our daily life and work and every aspect of our society and economy being transformed by digital technology. Developers are the builders of this new era, writing the world's code. And GitHub is their home…
>
> … In short, developers will be at the center of solving the world's most pressing challenges. However, the real power comes when every developer can create together, collaborate, share code and build on each other's work."(Microsoft News Center 2018)

With digital transformations (e.g. 3D printing, blockchain technology, digitally enabled development platforms) making industries increasingly information oriented, new opportunities have emerged in non-IT organizations to adopt practices that have been successful in the IT industry. Predictably, the FLOSS model of development has attracted considerable attention from other disciplines. The allure of being able to tap into the vast reserves of skills spread across the globe, enabling the creation of products and services of high quality and functionality at a low cost appeals to many organizations and industries (Coverity Inc. 2013). For example, with life sciences increasingly becoming an information orientated science, it has been suggested that what worked for FLOSS development might be an answer to the spiraling cost of drug R&D (Munos 2006). Although some initiatives (e.g. MMV[3], DNDi[4], CAMBIA[5]) have looked towards adopting a FLOSS approach to drug discovery, they have been successful in adopting it only during the early phases, where ideas and solutions are crowdsourced from the community. While many organizations are considering the FLOSS approach for developing their products, migrating to a model of development that does not conform to traditional contractual and governance mechanisms, and organizational

[3] https://www.mmv.org/

[4] https://www.dndi.org/

[5] https://cambia.org/

boundaries can seem daunting. Practically, organizations are seeking to benefit from the opportunities offered by open collaboration but are unsure about how to integrate them with their own strengths and practices that they have painstakingly built over time. As information systems (IS) researchers, we are in a unique position to enrich the theories surrounding the FLOSS artifact and inform organizations on how they can facilitate value by adopting a FLOSS approach to development. Herein lies my long-term research objective - to study the value creation mechanisms associated with the FLOSS model of development and understand how these mechanisms can be replicated more widely across IT and non-IT organizations. In essence, my long-term research objective is comprised of two parts: (a) in the first part there is a need to understand the value creation mechanisms of the FLOSS artifact and the contextual conditions that limit their applicability, and (b) in the second part there is a need to understand how these mechanisms can be institutionalized and replicated across other industries. My dissertation is a sincere attempt to address the first part of this long-term research objective and my planned future work will look towards expanding my dissertation findings to address the second part.

Structured as three essays, this dissertation considers the value creation mechanisms associated with three important aspects of FLOSS projects - a) work structures, b) team composition and governance, and c) community ideologies. To begin, any theoretical enquiry of the value creation mechanisms associated with the FLOSS model of development starts at the information technology artifact, the actual software developed, which creates value through its use in the community. And fundamental to building the complex FLOSS artifact is the unique work orchestration mechanisms that emerge to motivate and coordinate task work of geographically distributed individuals and organizations (Howison and Crowston 2014; Lindberg et al. 2016). Therefore, replicating the FLOSS model of development requires careful understanding of the *work structures* that provide the framework for effectively organizing task work and the mechanisms through which they facilitate value. Second, gathered around

the FLOSS artifact, is the *team of contributors* comprised of both individuals and organizations who collaborate to build the software. In the absence of formal governance practices like – requirements planning and release management, communities of practice sustain themselves by nurturing informal network governance mechanisms like – access restrictions, collective sanctions, macroculture, and reputation (Jones et al. 1997). Fostering informal network governance mechanisms in FLOSS *teams* to protect social exchanges and overcome coordination challenges is an important aspect of sustaining collaboration in FLOSS projects and hence closely associated with FLOSS project value. Third, encompassing both the work structures and teams of contributors are the ideological undercurrents that shape the objectives and motivations of the FLOSS community (Daniel et al. 2018). Given its effect on the motivation of the contributors, ideological changes are expected to moderate the value creation mechanisms associated with different FLOSS attributes, like the project's work structures. Therefore, harnessing the full potential of the FLOSS phenomenon would require a clear understanding of the *ideologies* that shape the community and ensuring a seamless fit between the ideological needs of the community and the different project attributes.

Conceptually, the three essays are grounded in theories from information systems and organization studies. The essays enrich these theories by not only addressing some of the existing theoretical gaps, but also by clarifying and challenging some of the assumptions on which the theories are founded (Alvesson and Sandberg 2011). The following subsections briefly introduces the theoretical background and the motivation for the three essays.

Work structures: Drawing from commercial software development practices, early research on the task characteristics of FLOSS projects indicates the importance of codebase architecture (e.g., modularity; Baldwin and Clark 2006) and of coordination mechanisms (e.g., Chua and Adrian 2010, Crowston et al. 2005, Mockus et al. 2002) for ensuring successful collaboration. While this large body of research provides an excellent understanding of FLOSS task work from

an architectural perspective, rather less is known about the sociotechnical nature of work organization during the production process and its implication on FLOSS project outcomes. Recognizing this gap, recent works of Howison and Crowston (2014) and Lindberg et al. (2016), have called attention to the characteristics of the software artifact, such as the emergent structures of work —which may be instrumental in overcoming the challenges related to FLOSS task work orchestration. These studies have observed the emergence of unique routines in the work structures of FLOSS projects that are surprisingly effective at establishing the delicate balance between —managing developers' contributions, and sustaining the contributors' needs for openness and autonomy. In this context, Howison and Crowston (2014), observed and conceptualized superposition of tasks as the dominant work orchestration mechanism in FLOSS projects, wherein motivationally independent tasks are incrementally layered to create the software. This work orchestration mechanism is different from that observed in the case of traditional software development, where the focus is towards co-work and concurrent task development through a modular task design. Although the dominance of the superposed orchestration mechanism in FLOSS projects has been attributed to its ability to satisfy the psychological needs of the intrinsically motivated contributors, it is still unclear if this motivational influence, theorized using self-determination theory (SDT; Ryan and Deci 2000), could be scaled up to the project level. This calls for a deeper theoretical enquiry to better understand how these unique ways of organizing task work influence project success. Following this line of enquiry, essay 1 attempts to enrich the theory of collaboration through open superposition (Howison and Crowston 2014) by examining the boundaries describing the influence of task superposition on FLOSS project success. This leads to the broad research question that essay 1 addresses:

How do work structures influence the success of FLOSS projects?

Team composition and governance: FLOSS project contributors are often classified as belonging to either the core or the peripheral group of developers (Crowston, Wei, et al. 2006). In Distributed Version Control Systems (DVCS) based development platforms like GitHub, we can delineate the core and peripheral contributors based on their rights of access to the main project code (Rullani and Haefliger 2013). In this classification, the core contributors are those who can directly make changes to the main project code, while the peripheral contributors are those who are not given write access to the project but still contribute to the project. Studies that have looked at the contributor groups from the nature of their participation and type of contribution have established the importance of both the core and peripheral groups as antecedents to the success of the project (Sagers 2004; Setia et al. 2012). However, the interplay between the two groups of contributors and their governance is less understood. Given this gap in our understanding, the second essay examines the mechanisms through which effective organization of teams can overcome the challenges of distributed work. Specifically, the second essay addresses the following broad research question:

How do informal governance mechanisms that emerge in FLOSS teams influence the survival of FLOSS projects?

Community ideologies: Though, volunteer driven FLOSS development was founded on the ideological beliefs of 'openness', 'co-operation', and 'absence of any commercial appropriation', in recent years the FLOSS movement has witnessed two ideological shifts. First, the emergence of 'permissive FLOSS licenses' that allow commercial appropriation of the collaboratively developed code (Daniel et al. 2018), and second, 'organizational ownership' of FLOSS projects (Fitzgerald 2006). The fusion of the two vastly different ideologies (open vs. commercial software) has created a new corporate-communal landscape which has altered the value creation mechanisms that were embedded in the founding ideologies of FLOSS (Germonprez et al. 2016). Given this dynamism of community ideologies, understanding ideological shifts that reshapes the phenomenon and their impact on FLOSS project outcomes is important for

ensuring the long-term viability of the FLOSS development approach. Theoretically, this dynamism implies that the mechanisms that are shaped by the ideological beliefs are associated with boundary conditions of time, which need to be clearly established (Suddaby 2010). Of the mechanisms shaped by community ideologies, its influence on the motivations of contributors is particularly noteworthy (Daniel et al. 2018). Because ideological beliefs shape the motivational needs of the volunteer contributors (Daniel et al. 2018), and motivational needs of contributors shape the emergent work structures in FLOSS projects (Howison and Crowston 2014), it is expected that ideological shifts could influence the mechanisms through which dominant work structures in FLOSS projects are related to their outcomes. Motivated by the need to understand these ideological shifts considering their influence on the unique work orchestration mechanisms examined in essay 1, essay 3 attempts to answer the following research question:

How do ideological shifts transform the value creation mechanisms associated with FLOSS work structures?

An important aspect of this dissertation is to expand our knowledge regarding organizational participation in FLOSS projects. Organizational participation in FLOSS projects can result in a trade-off between openness (increasing autonomy and stimulating innovation, creativity, and organizational growth) and control (over platform activities, efficient development practices, and intellectual property right appropriation; Engeström 2007; Jarvenpaa and Lang 2011). The trade-off manifests as a boundary management problem that, if effectively managed, can directly influence the innovative and absorptive capacity of the FLOSS community (Teigland et al. 2014). Given this trade-off when organizations own FLOSS projects, each essay of this dissertation explores the implications of organizational ownership on the specific value creation mechanism being studied.

Structure of The Dissertation

This dissertation is structured as three similarly themed but separate essays exploring the value creation mechanisms associated with FLOSS projects. The three essays have separated theoretical foundations and their implications contribute to different aspects of FLOSS research and practice. To collect data for our analysis, we selected FLOSS projects started in early 2014 by both organizations and individuals on GitHub. We collected detailed task level data and followed the projects for a period of two years. During this period, some projects in our sample grew and become popular, for example, the project POP[6] started by Facebook and the project Slick[7], which was started by an individual (Ken Wheeler). During the same period, some other projects in our sample did not gain in popularity and became inactive.

The first essay, examines how the unique nature of FLOSS work which is dominated by the sequential layering of individual tasks, referred to as superposition, acts as an antecedent to the project's success. Building on the theory of collaboration through open superposition (Howison and Crowston 2014), the essay theorizes the motivational mechanisms that operate within superposed work structures and studies the contextual conditions that limit the influence of superposition on FLOSS project value. Furthermore, given the increasing usage of FLOSS by organizations, the study investigates the specificities brought to these motivational mechanisms when FLOSS projects are owned by organizations. Developing an innovative operationalization of the work structures of FLOSS projects, this essay explores the relationship between the degree of superposition and the success of the project for individual- and organization-owned projects. The findings of the first essay advance our understanding of work structures, motivation, and organizational participation in FLOSS environments.

---

6 https://github.com/facebook/pop

7 https://github.com/kenwheeler/slick

While the first essay establishes the importance of task-work organization in FLOSS projects, the second essay expands the inquiry into the role of team composition in the project's success. Building on the theories of coordination (Malone and Crowston 1994) and network governance (Jones et al. 1997), this essay studies the influence of source code access restrictions imposed on team members in mitigating coordination challenges. The study also investigates the changes brought to the coordination mechanisms when open source projects are owned by organizations. Using a Cox proportional hazard model (Hosmer et al. 2008a), the essay explores the relationship between the proportion of contributors who are given write access to the source code in the team and the survival of the project. Overall, the second essay advances our understanding about contributor roles, access restrictions, and organizational participation in open source environments. The findings provide open source researchers and practitioners with fresh insights for better understanding and modeling project teams to facilitate their success.

The third essay pursues an overarching view of the FLOSS community by examining the ideological foundations of the FLOSS community and studies its influence on project success. This essay scrutinizes two ideological shifts seen in the FLOSS community that have altered the beliefs of 'openness' and 'prevention of commercial appropriation', on which the open source phenomenon was founded. Rooted in self-determination theory (Ryan and Deci 2000), this essay theorizes the mechanisms through which ideological changes influence the pathways through which work structures in FLOSS projects are related to their success. Using an instrument variable approach, this essay explores the moderating influence of the ideological shift pertaining to license type on the relationship between the work structures and project success for both individual and organization owned projects. Overall, the third essay advances our understanding of the important role that ideologies play in shaping the relationship between work structures and success of the FLOSS projects.

Each essay is self-contained in terms of literature review, hypotheses development, and implications for research and practice. The essays together contribute to different aspects of literature on FLOSS. The theoretical foundation and research hypotheses for all the three essays are summarized in Table 1-1. Further, Table 1-2 presents the research questions, methods & measures, and important findings from all the three essays at a glance.

Table 1-1: Three Essays at a Glance: Theoretical Foundation and Research Hypotheses

Essay	Theoretical Foundation and Hypotheses
1	**Theoretical foundation:** • Theory of collaboration through open superposition (Howison and Crowston 2014) • Self-determination theory (Ryan and Deci 2000) • Affective events theory (Weiss and Cropanzo 1996) **Hypotheses:** • Hypothesis 1. In the context of FLOSS projects, the degree of superposition has a nonlinear relationship with project popularity such that project popularity increases with an increase in the degree of superposition up to a particular value (the turning point). Beyond this optimal degree of superposition, any further increase reduces the popularity of the project • Hypothesis 2a: In the context of FLOSS projects, the project ownership type moderates the relationship between the degree of superposition and project popularity such that the degree of superposition has a significantly lower influence on the popularity of the project for organization-owned projects than for individual-owned projects • Hypothesis 2b. In the case of organization-owned projects, the degree of superposition at which project popularity is at a maximum (the turning point) is significantly lower than for individual-owned projects
2	**Theoretical foundation:** • Coordination theory (Malone and Crowston 1994) • Theory of network governance (Jones et al. 1997)

	Hypotheses: • Hypothesis 1: A greater proportion of core contributors in a project will lead to a lower chance of survival of the project • Hypothesis 2: Organizational ownership mitigates the negative influence that the proportion of core contributors has on project survival • Hypothesis 3: The average code contributions per core contributor decreases in the case of organization owned project as compared to individual owned projects
3	**Theoretical foundation:** • Self-determination theory (Ryan and Deci 2000) • Ideological fit in open source communities (Daniel et al. 2018) **Hypotheses: First Ideological shift - license choice** • Hypothesis 1a: For individual owned projects, the type of license moderates the relationship between the degree of superposition and the popularity of FLOSS projects, such that, for projects with restrictive licenses an increase in the degree of superposition tends to have a higher positive influence on the popularity of the project than for projects with permissive licenses • Hypothesis 1b: For individual owned projects, the type of license moderates the relationship between the degree of superposition and the survival of FLOSS projects, such that, for projects with restrictive licenses, an increase in degree of superposition tends to have a higher positive influence on the survival of the project than for projects with permissive licenses **Hypotheses: Second Ideological shift - organizational ownership** • Hypothesis 2a: For organization owned projects, the moderating influence of license type on the relationship between the degree of superposition and the popularity of FLOSS projects is less in comparison to individual owned projects • Hypothesis 2b: For organization owned projects, the moderating influence of license choice on the relationship between the degree of superposition and the survival of FLOSS projects is less in comparison to individual owned projects

Table 1-2: Three Essays at a Glance: Research Questions, Methods, and Main Findings

Essay (Research Questions and Measures)	Essay (Models and Main Findings)
Essay 1: Work structures of FLOSS projects **Research questions:** • How does the extent of task superposition influence FLOSS project success? • How do organization-owned FLOSS projects differ from individual-owned FLOSS projects in terms of task superposition, and does this difference influence project success? **Unit of analysis:** • Project level of analysis **Dependent variable:** • Project popularity measured as the number of stars that a project has received **Independent variable:** • Degree of superposition, operationalized as the ratio of number of versions of the project to number of individual task contributions in the project • Ownership type of the project	**Empirical model:** • Ordinary least squares • Negative binomial regression **Main findings:** The results from the analysis of a large sample of FLOSS projects hosted on GitHub support a nonlinear relationship between the degree of superposition and the success of the FLOSS project. Moreover, we find that the type of ownership moderates this nonlinear relationship such that (1) organizational ownership mitigates the influence of the degree of superposition on the success of the project, and (2) under organizational ownership, the optimal degree of superposition (the point at which the success of the project is at a maximum) is lower than for individual-owned projects.

Essay (Research Questions and Measures)	Essay (Models and Main Findings)
Essay 2: Team Composition and Governance of Open Source Projects **Research questions:** • What role does contributor access restrictions have in influencing the survival of FLOSS projects? • How does organizational ownership moderate this relationship? **Unit of analysis:** • Hypotheses 1 and 2 – Project level of analysis • Hypothesis 3 – Contributor level of analysis **Dependent variables:** • Hazard rate: Likelihood a project becomes inactive at time *t* given that it has survived till time *t* • Average code contributions per contributor **Independent variables:** • Proportion of contributors with write access (core contributors) • Contributor type (core or peripheral contributor) • Ownership type of the project	**Empirical model:** • Cox-proportional hazard model • Hierarchical linear model **Main findings:** Using a Cox proportional hazard model, this essay demonstrates that the relationship between the proportion of core contributors (who are given write access to the source code) and the survival of the project, is moderated by the nature of project ownership — individual versus organization owned projects. Interestingly, the observed moderation is a crossover interaction effect that changes from negative for individual owned projects to positive for organization owned projects.

Essay (Research Questions and Measures)	Essay (Models and Main Findings)
Essay 3: Community ideologies **Research question:** • How have the ideological shifts invoked by (a) the emergence of permissive licenses, and (b) the shift towards organizational ownership, transformed the influence of FLOSS work structures on project outcomes? **Unit of analysis:** • Project level of analysis **Dependent variables:** • Project popularity measured as the number of stars that a project has received • Likelihood of survival of the project **Independent variables:** • Choice of license (Permissive or restrictive) • Degree of superposition, operationalized as the ratio of number of versions of the project to number of tasks in the project **Instrument variables for choice of license:** • Country's cultural dimension – Individualism vs. collectivism • Country's social protection contribution as a percentage of total expenditure in the year 2014-2015	**Empirical model:** • 2 stage least square model with endogenous regressors • Probit model with endogenous regressors **Main findings:** Using an instrument variable approach, our analysis of a large sample of FLOSS projects hosted on GitHub confirm the significance of both the ideological shifts with some interesting contextual differences across the two project outcomes. Specifically, we find that the ideological shifts pertaining to license type and organizational ownership has a significant moderating influence on the relationship between the work structures and project popularity. However, for project survival, the findings are inconclusive when we adopt the instrument variable approach.

2 Essay 1: Work Structures of FLOSS Projects

Abstract

Collaboration through open superposition describes the dominant work orchestration mechanism observed in Free (Libre) and Open Source Software (FLOSS), wherein the software development occurs by the sequential layering of individual tasks. This work orchestration mechanism is different from the traditional idea of software development, where the focus is towards co-work and concurrent development facilitated by a modular software design architecture. This essay theorizes and examines the motivational mechanisms that operate within superposed work structures to influence the success of FLOSS projects. We also unearth the contextual conditions that may limit the influence of the superposed nature of work on FLOSS project success. Furthermore, given the increasing usage of FLOSS by organizations, we investigate the specificities brought to these motivational mechanisms when FLOSS projects are owned by organizations. The results from our analysis of over 6500 FLOSS projects hosted on GitHub support a nonlinear relationship between the degree of superposition and the success of the FLOSS project. Moreover, we find that the type of ownership moderates this nonlinear relationship such that (1) organizational ownership mitigates the influence of the degree of superposition on the success of the project, and (2) under organizational ownership, the optimal degree of superposition (the point at which the success of the project is at a maximum) is lower than for individual-owned projects. This research advances our understanding of work structures, motivation, and organizational participation in FLOSS environments by describing the influence of task structures on the success of projects. The study also provides FLOSS practitioners with valuable insights for modeling project task structures to facilitate their success.

Keywords: open source software; collaboration; virtual teams; motivation; task structure; IS development; superposition; project success; econometrics

Introduction

Over the years, the FLOSS model of development has allowed project owners to effectively tap into the vast reserves of programming skills spread across the globe to create software that surpasses proprietary software in both quality and functionality (Coverity Inc. 2013). In addition, FLOSS also offers advantages in terms of evolved coordination and improved motivational mechanisms, which are increasingly attracting both —individual developers and commercial organizations to experiment with FLOSS as a viable mode of software development (see Ke and Zhang 2010, von Krogh et al. 2012). While this model of development offers several benefits to the project owners, the orchestration of task work across geographically dispersed contributors, who are often sporadically engaged in different projects, can be challenging (Howison and Crowston 2014, Lindberg et al. 2016). This challenge is exacerbated by the fact that governance mechanisms commonly used to organize task work in traditional software development projects have been found to conflict with the contributors' motivational needs (Ke and Zhang 2010) and established social practices of FLOSS communities that epitomize the principles of openness and autonomy (von Krogh et al. 2012).

Recent works of Howison and Crowston (2014) and Lindberg et al. (2016), have called attention to the characteristics of the software artifact, such as the emergent structures of work —which may be instrumental in overcoming the challenges related to FLOSS task work orchestration. These studies have observed the emergence of unique routines in the work structures of FLOSS projects that are surprisingly effective at establishing the delicate balance between —managing developers' contributions, and sustaining the contributors' needs for openness and autonomy. In this context, Howison and Crowston (2014), observed and conceptualized superposition of tasks as the dominant work orchestration mechanism in FLOSS projects, wherein motivationally independent tasks are incrementally layered to create the software. This work orchestration mechanism is different from that observed in the case of traditional software development, where the focus is towards co-work and concurrent task development through a modular task design. The authors found evidence of this form of work

orchestration in the Fire and Gaim FLOSS projects, where approximately 80 percent of tasks involved no co-work (Howison and Crowston 2014).

Although the dominance of the superposed orchestration mechanism in FLOSS projects has been attributed to its ability to satisfy the psychological needs of the intrinsically motivated contributors, it is still unclear if this motivational influence, theorized using self-determination theory (SDT; Ryan and Deci 2000), could be scaled up to the project level. This calls for a deeper theoretical enquiry to better understand if and how these unique ways of organizing task work influence project success. Following this line of enquiry, this essay attempts to enrich the theory of collaboration through open superposition (Howison and Crowston 2014) by unearthing the boundaries describing the influence of task superposition on FLOSS project success. This leads us to our first research question:

RQ1: How does the extent of task superposition influence FLOSS project success?

Given that organizations are increasingly shifting their strategic focus towards FLOSS development (Microsoft News Center 2018), it is imperative to understand if the assumptions and mechanisms that form the basis of the theory of superposition are also applicable to the context of organizational ownership. Apparently, organizations and FLOSS communities are grounded in very different premises (profit maximization versus sharing ideology) but the coexistence of these seemingly orthogonal social practices and institutional norms is imperative for the success of organization owned FLOSS projects (von Krogh et al. 2012, Howison and Crowston 2014). We posit that organizational participation in FLOSS projects will have strong implications for developers' motivations and the underlying mechanisms that influence project success —which leads to the second research question:

RQ2: *How do organization-owned FLOSS projects differ from individual-owned FLOSS projects in terms of task superposition, and does this difference influence project success?*

By answering these research questions, we aim to make the following contributions to theory and practice. First, we enrich the theory of superposition (Howison and Crowston 2014), by examining whether satisfying the psychological needs at the level of contributors could be

effectively scaled-up to the level of the project and made to manifest as project success. By doing so, we offer a more nuanced conceptualization of the mechanisms that explain the role of superposition of task work in facilitating FLOSS project success. The usefulness of this conceptualization is exhibited by the boundary conditions that we uncover for the applicability of the phenomenon (Suddaby 2010). Second, this essay contributes to the theoretical understanding of the different ways in which superposition influences project success in individual and organizational-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 (e.g., Capra et al. 2011, Fitzgerald 2006). Building on this prior research stream, we show that the influence of superposition on the success of the project is sensitive to the actors involved in the project and their models of engagement. Cognizance of these influences is crucial for understanding how organizations can establish the delicate balance between openness (stimulating innovation, creativity, and organizational growth) and control (over platform activities, efficient development practices, and intellectual property right appropriation; Engeström 2007, Jarvenpaa and Lang 2011), which is an important antecedent to the success of FLOSS projects (Teigland et al. 2014). Lastly, the results of this study may help advance theories regarding FLOSS success and offer practical insights to project owners. Management scholars have expressed considerable concern about the failure of academic research to penetrate the practitioner community (Rynes et al. 2001). I believe that a better understanding of the influence of the nature of task work on the success of the project can lead to better management practices for planning potentially successful FLOSS projects.

Conceptual Development

Before discussing the relationship between degree of superposition and FLOSS project success, we develop the notion of FLOSS project success in the context of this study. We then provide a brief literature background regarding the nature of task work in FLOSS projects, following which we develop the theoretical arguments surrounding the main relationship between

superposition and project success (Hypothesis 1). Subsequently, we present the different models of organizational engagement in FLOSS projects and describe how these models of engagement can influence the relationship between superposition and project success (Hypotheses 2a and 2b).

FLOSS 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 2004). Furthermore, FLOSS projects are freely available in the public domain, where there is no need to request permission to use them. Nonetheless, researchers have continually attempted to describe the meaning of success in the context of FLOSS projects. Crowston 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) adapted Delone and McLean's (1992) IS success model to develop 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. Most empirical studies that examine the mechanisms associated with FLOSS project success have operationalized success along one or more of the aforementioned dimensions (Crowston et al. 2012). The choice of the success measures in these studies is not only a result of the nature of the relationship being examined but also conditional on the availability of data that would allow their operationalization.

Among the different available measures, 'popularity of the project' is a particularly useful measure of success because popular projects tend to engage a larger community of users which leads to improved ideas (Howison and Crowston 2014). Moreover, measures of popularity are also correlated to the number of bugs, the number of participants in the bug trackers (Crowston, Howison, et al. 2006), and the total number of developers overall (Krishnamurthy

2002, Subramaniam et al. 2009). These correlations suggest that project popularity can capture the essence of both quality and functionality of the FLOSS software. Given this salience, different measures of popularity have been frequently used to determine FLOSS project success, for example —number of downloads (Fershtman and Gandal 2011; Rebeca and García 2009), commercial success (Grewal et al. 2006; Midha and Palvia 2012), user interest (Subramaniam and Nelson 2009, Stewart et al. 2006), OSS product awareness (Setia et al. 2012) and project viewership and downloads (Crowston, Howison, et al. 2006). Following the example of these studies and in recognition of the importance of attracting greater attention from the community, we adopt popularity as the measure of FLOSS project success and use it to hypothesize and test our relationships.

Nature of Task Work in FLOSS projects

Drawing from traditional commercial software development practices, early research on the task characteristics of FLOSS projects indicates the importance of modular software architectures for ensuring successful collaboration. For example, Baldwin and Clark (2006) show that a high level of modularity coupled with greater design options not only attracts more contributors but also helps to sustain their cooperation in FLOSS projects. However, an increased number of FLOSS contributors can make task coordination difficult. Recognizing the problem of task coordination in large FLOSS projects, researchers soon began to examine the unique mechanisms through which FLOSS projects could overcome coordination issues. For example, Crowston et al. (2005) used coordination theory (Malone and Crowston 1994) to contrast coordination mechanisms in commercial software development projects with those of FLOSS projects. Another study by Crowston and Scozzi (2004) examined the coordination mechanisms involved in bug-fixing processes and found that the tasks involved are sequential and comprise only a few steps. Chua and Adrian (2010) examined how the characteristics of material artifacts impact cross-project coordination in large open source projects. Mockus et al. (2002) examined the impact of project size on the coordination mechanisms adopted.

While this large body of research provides an excellent understanding of FLOSS task work from the perspective of codebase architecture (e.g., modularity; Baldwin and Clark 2006) and of coordination mechanisms (e.g., Chua and Adrian 2010, Crowston et al. 2005, Mockus et al. 2002), rather less is known about the sociotechnical nature of work organization during the production process. Recognizing this gap, two recent works have attempted to provide a greater understanding from this perspective - a) Howison and Crowston, (2014) who found that superposition of tasks is the dominant work- orchestrating mechanism in FLOSS environments, on the basis of which, they conceptualized the theory of collaboration through open superposition b) Lindberg et al. (2016) who theorized developer and developmental interdependencies outside the purview of superposition, which are resolved by unique task and knowledge routines. The current research attempts to advance the theory of collaboration through open superposition by scrutinizing the assumptions that define the boundaries for the influence of superposition. While the current study is restricted to understanding the influence of task work orchestration as proposed by the theory of collaboration through open superposition, we acknowledge that future studies delineating the works of Howison and Crowston, (2014) and Lindberg et al. (2016) would supplement the effort to understand the influence of work structures on the success of projects.

Theory of Collaboration through Open Superposition. Superposition is the process through which software development occurs in a sequential manner, with changes to the software added incrementally, on top of one another. Each change represents a task that is independently built by a contributor and has its own functional payoff through the improvements it brings to the application (Howison and Crowston 2014). Superposed task work is characterized by (a) individual task work, and (b) incremental layering of motivationally independent tasks. This unique work-breakdown structure has been found to accomplish complex work through a process of 'productive deferral'; Productive deferral is the process by which, tasks that are envisioned to be too large to be implemented through individual, motivationally independent layers are deferred until code written by someone else

(and sometimes for entirely different reasons) makes the envisioned task easy enough to be undertaken through relatively simple, quick individual work (Howison and Crowston 2014). Thus, FLOSS contributors often wait for the 'missing piece' of code to be developed by someone else in the community before getting back to complete the large task. We leverage on the latent mechanism of productive deferral, prevalent in FLOSS development, to describe the hypothesized relationships.

The theory of collaboration through open superposition states that in the case of FLOSS projects, an emergent superposition of tasks provides the most effective work-breakdown structure for enhancing motivation to contribute, and at the same time allows for the creation of complex software. To build their argument, Howison and Crowston (2014) invoke theories of motivation and coordination. Specifically, they expand the work of Ke and Zhang (2010), who apply self-determination theory (SDT; Ryan and Deci 2000) and affective events theory (AET; Weiss and Cropanzo 1996) to show that superposition creates an effective work-breakdown structure that satisfies the three psychological needs of autonomy, competence, and relatedness posited by SDT, leading contributors to expend greater task effort in FLOSS projects. First, superposition minimizes the interdependencies among contributors, thereby satisfying their need for *autonomy*. Second, superposition promotes contributors' sense of *competence*, because each task independently results in an improvement to the shared output of the project. Third, superposition addresses the need for *relatedness*, as the layering of tasks on the work of others and the potential support from other contributors provides connectedness in a manner that does not undermine contributors' autonomy. In the subsequent subsection, we look at how satisfying the motivational mechanisms invoked by superposition may influence the popularity of the project.

Relationship between Degree of Superposition and FLOSS Project Popularity. By providing the motivational mechanisms that satisfy the innate psychological needs of competence, autonomy, and relatedness, superposition generates a positive affective state for contributors (Richard M. Ryan and Deci 2000). The increased positive affective state enhances

the task effort that an individual will expend (Weiss and Cropanzo 1996), encouraging greater contributions to the FLOSS project and leading to an increase in the quality, functionality, and usability of the software for the end user (Ke and Zhang 2010). Consequently, an increase in the degree of superposition is expected to positively influence the popularity of a FLOSS project among users. But will continual increases in the degree of superposition be associated with progressive growth in the FLOSS project's popularity? To answer this question, we need to delve deeper into the mechanisms that contribute to the increased popularity of FLOSS projects.

While superposition provides a positive motivational mechanism for individuals to contribute to FLOSS projects, we contend that there is a potential cost involved in adopting a superposed work-breakdown structure. This cost, which stems from the inefficiencies involved in adopting a sequential pattern of development (with very little co-work), becomes a concern at higher degrees of superposition, where contributors tend to predominantly defer complex/big tasks rather than engage in co-work. This increased adoption of the productive deferral mechanism and avoidance of co-work associated with a high degree of superposition will encourage potential contributors to wait until the missing piece has been provided by someone else in the FLOSS community. This may lead to a perceived loss of the contributor's control over the project (Guenter et al. 2014), leading to some frustration (Selvidge et al. 2002). Overall, this may result in an arousal of negative affect in contributors (Zhang 2013), especially if they have to face long delays while waiting for the work of others to make the envisioned task simpler (Selvidge et al. 2002). Prior studies have shown that negative affective states caused by loss of control and frustration result in reduced task effort (Greenberger and Strasser 1986; Weiss and Cropanzo 1996; Zhang 2013), which may result in a decrease in the quality, functionality, and usability of the software. This may in turn be instrumental in reducing the popularity of the project.

Hence, we can see that superposition has both positive and negative latent influences. The positive motivational environment enabled by enhanced autonomy tends to be countered by

the negative influence of loss of control and frustration due to avoidance of co-work at increasing levels of superposition. To understand the combined effect of these latent influences we can envision their approximate functional forms and combine them additively (Haans et al. 2015). In our case, the two latent functions are - a) the influence of superposition in increasing the positive affective state of the contributor due to increased autonomy, and b) the influence of superposition in increasing the negative affective state of the contributor due to increased loss of control and frustration. We propose that the relationship between degree of superposition and the positive affective state of the contributor due to increased autonomy follows the law of diminishing returns. That is, with each additional increase in superposition while keeping other aspects constant, there is a decline in the marginal benefit that a contributor perceives from an increase in autonomy. On the other hand, we propose that loss of control and frustration caused due to avoidance of co-work and productive deferral mechanisms tends to manifest itself at high values of superposition. That is, the influence of superposition on increasing the negative affective state of contributors follows an increasing trend. Thus, we see that the latent effect of superposition on the positive affective state (through an increase in autonomy) increases at a decreasing rate and eventually levels of; while its effect on the negative affective state (through an increase in frustration and loss of control) is low at lesser levels of superposition but increases at an increasing rate (see Figure 2-1). When latent relationships of these forms are combined, the result is an overall inverted U-shaped relationship between degree of superposition and the net affective state [8](Figure 2-1; Haans et al. 2015). Following AET (Weiss and Cropanzo 1996), changes in the affective states of contributors can result in corresponding changes to the task effort they expend. In turn, the changes in task effort manifest as variations in its popularity.

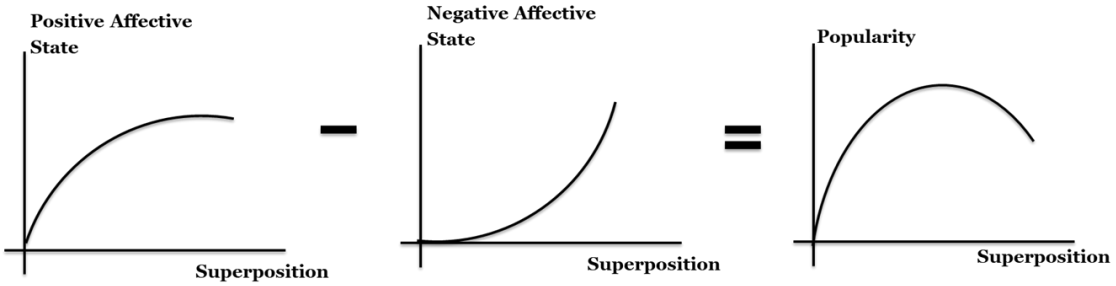
Thus, we posit that in the FLOSS context, the relationship between superposition and the popularity of a project is positive until the popularity of the project reaches a maximum, after

[8] The latent relationships proposed are illustrative and may differ slightly. See Haans et al. (2015), for a more detailed explanation of the different conditions under which the U-shaped relationship can emerge.

which the relationship becomes negative. That is, as the degree of superposition initially increases, the popularity of the FLOSS project increases because in the beginning superposition provides the right motivational mechanisms to attract and retain contributors. But at a certain degree of superposition (the turning point), the marginal benefit from an increase in autonomy is counterbalanced by a reduction in project popularity, which is attributable to the negative affect arising from productive deferral mechanisms and avoidance of co-work. Hence, we hypothesize:

Hypothesis 1. In the context of FLOSS projects, the degree of superposition has a nonlinear relationship with project popularity such that project popularity increases with an increase in the degree of superposition up to a particular value (the turning point). Beyond this optimal degree of superposition, any further increase reduces the popularity of the project.

Figure 2-1: Effect of Superposition on the Popularity of FLOSS Projects through its Latent Influence on the Affective State of the Contributors



Effect of superposition on the positive affective state of the contributors through the provision of autonomy

Effect of superposition on the negative affective state of the contributors due to increased frustration and loss of control

Organizational Ownership of FLOSS Projects

By early 2000, organizations began to realize the commercial potential of FLOSS (Wagstrom 2009). This led to a transformation of FLOSS communities, which had mainly been volunteer driven, to communities that attracted organizations and commercial interests (Fitzgerald 2006). Organizational ownership in FLOSS projects has not only been found to provide extrinsic motivation through rewards and signaling (Lerner and Tirole 2003), but can also create an environment that enhances the intrinsic motivation of contributors (Spaeth et al. 2015). The added influence of organizational ownership on the project contributors'

motivations and the owner's potential need for control may influence the relationship between superposition and the popularity of the project. This influence can be studied through three non-mutually-exclusive models of organizational engagement: the coding, support, and management models (Capra et al. 2011). In the *coding model* of engagement, the organization owner supports the development activities of the project by involving its employees and/or opening some of its proprietary code to the FLOSS project. In the *support model*, the organization owner provides support to nondevelopmental activities through direct or indirect financial contributions. In the *management model*, the organization owner engages in project management and coordination activities. In the subsequent subsections, we study how the models of organizational engagement tend to influence the relationship between superposition and popularity of the project.

Coding model. In the coding model, organizations engage in development activities by contributing to the code, often in cooperation with volunteer contributors**.** Here, paid employees of the organization commit to writing the code that helps fix bugs, customizing the existing code, and building distribution packages (Capra et al. 2011). For example, Facebook contributed more than nine million lines of code to the open source community through 200 different projects in the first six months of 2014 alone (Pearce 2014). This includes some of their major projects, such as React and Pop that also receive many contributions from the external community.

In this model of engagement, an organization contributes the bandwidth of its employees to the FLOSS project. Unlike volunteer contributors, who often contribute during their free time, the employees of an organization receive tangible rewards (in terms of salary) for their contributions. The effect of the coding model on the relationship between degree of superposition and the popularity of a project can be related to the effect of introducing extrinsically motivated contributors to a FLOSS project. Extrinsically motivated contributors who receive tangible rewards have less need for autonomy than intrinsically motivated volunteer contributors (Deci et al. 1999). Further, the introduction of organizational employees

affords the project owner more control over the execution of big or complex tasks. That is, organizations can direct their employees to devote their effort to addressing the big or complex tasks through co-work rather than relying on productive deferral to complete the task; making such deferral less of a factor. Therefore, we posit that organizations' engagement in the development activities of their FLOSS projects will tend to dampen the net effect of superposition on the positive affective state of the contributors, due to the inclusion of more extrinsically motivated employees with less need for autonomy. Thus, by participating in the development activities of the projects they own, organizations tend to mitigate the influence of superposition on the popularity of these projects.

In addition to including extrinsically motivated employees in a FLOSS project, organization ownership of FLOSS projects can also provide certain extrinsic motivations for the volunteer contributors. Specifically, signaling for career advancement has been identified as an important extrinsic motivation for contributing to FLOSS projects (Lerner and Tirole 2003). Thus, in organization-owned FLOSS projects, volunteer contributors may also be extrinsically motivated by the opportunity to signal their abilities to the organization owner. Provision of this additional extrinsic motivation to participate can further dampen the influence of superposition on the popularity of the project.

Support (non-development) model**.** In the support model, organizations provide financial, infrastructure, marketing, and/or customer support (Capra et al. 2011). Each of the non-developmental support activities is an organizational investment in the project. Organizational investment in FLOSS projects creates opportunity costs, increasing the time cost of money and pressure to see payoffs sooner rather than later (Howison and Crowston 2014). Because of the increased time cost of money, delays due to productive deferral and avoidance of co-work have a larger negative influence on project popularity. Thus, in the case of organization-owned projects, increased productive deferral and avoidance of co-work associated with a high degree of superposition may not only lead to a negative affective state for the contributors, due to frustration and loss of control, but may also increase the time cost of money for their

investments. The increase in the time cost of money can de-incentivize the organization owner from making future investments in the project leading to a reduced focus towards adding new functionality and improving quality of the software. Because of the added negative influence of the time cost of money, we posit that projects with higher superposition tend to lose out on investments that would otherwise have helped enhance their popularity compared to projects with lower superposition.

Management model. In the management model, an organization contributes to a project by performing activities related to project coordination and management (Capra et al. 2011). This model of participation has been described by Fitzgerald (2006) in terms of strategic planning initiatives and project management practices that organizations bring to FLOSS projects for the efficient coordination and management of development activities. This model of participation has also 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). Using Raymond's (1998) bazaar metaphor, it can be said that organizational ownership of FLOSS projects leads to the introduction of management practices that tend to transform the adopted model of development from a bazaar into a cathedral-like form. On the other hand, individual-owned projects tend to retain the bazaar model of FLOSS development, with fewer formal management practices than found in organization-owned FLOSS projects.

The introduction of well-defined goals and mechanisms for coordinating and controlling FLOSS development activities can also influence how the degree of superposition relates to the popularity of a project. Given that there are extrinsically motivated contributors who have less need for autonomy and the increased time cost of money, the formal practices and coordination structure allow organizations to focus on more efficient development practices and eventually lower the product delivery times. Thus, by engaging in the management model, the organization owner can set up practices that facilitate co-work instead of relying on

superposition to accomplish the tasks. For example, in the GNOME project, the GNOME Foundation acts as a centralized release coordination authority, creating release schedules and coordinating modules for release (O'Mahony 2005). These practices allow the organization owner to best leverage the presence of extrinsic motivation in the contributors and minimize the negative influence of the time cost of money. We posit that this increased emphasis on organized and concurrent work when organizations own FLOSS projects facilitates the moderating influence of the coding and support model on the relationship between superposition and the popularity of the project.

Moderating Effect of Owner Type on the Relationship between Degree of Superposition and FLOSS Project Popularity. Based on the three models of organizational involvement, the moderation influence of organizational ownership manifests as two related effects that tend to change the shape of the inverted U-shaped relationship between degree of superposition and popularity of the project. The changes to the shape of the curve is observed (a) as a flattening of the inverted U-shaped curve, and (b) as a left shift of the turning point. Table 2.1 summarizes the moderating influence of organizational ownership through the different models of engagement and its expected influence on the shape of the curve.

Table 2-1: Moderating Influence of Organizational Ownership through Different Models of Engagement

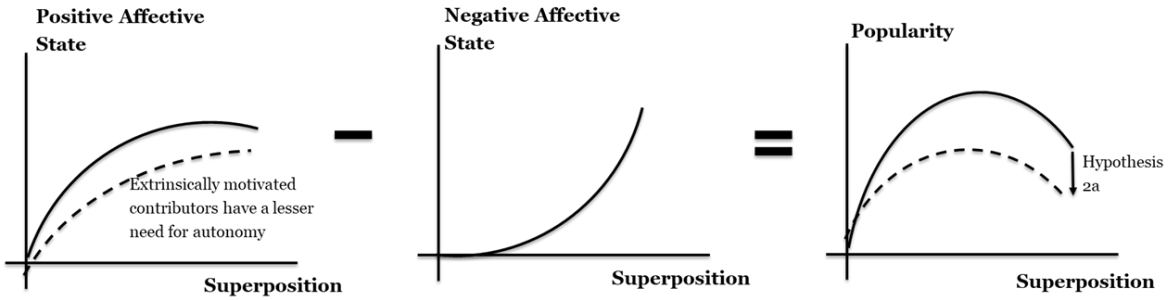
Model of Organizational Engagement	**Expected Changes to FLOSS Project**	**Expected Latent Influence**	**Expected Influence on the Relationship between Superposition and Popularity**
Coding model	Inclusion of extrinsically motivated contributors and increase in the net extrinsic motivation of the contributors to the project	Diminishes the influence of superposition on the net affective state of the contributors	Primarily weakens the curvilinearity and results in a flattening of the curve (Hypothesis 2a)

Support (nondevelopment) model	Provision of direct and indirect financial contributions for nondevelopment support for the project	Introduces the latent negative influence of time cost of money	Primarily results in the inclusion of a negative linear effect that results in a left shift of the turning point (Hypothesis 2b)
Management model	Introduction of project management and coordination practices for more efficient development	Facilitates and promotes co-work in the presence of extrinsically motivated contributors and an increased time cost of money	Facilitates hypotheses 2a and 2b

Flattening of the curve occurs when the moderator influences the latent mechanisms in such a way that the overall shape of the observed relationship becomes flatter, *although the turning point of the relationship need not change* (Haans et al. 2015). When the influence of organizational ownership is seen as an increase in the net extrinsic motivation of the contributors (through the coding model of engagement and the effect of signaling), the moderation effect is to weaken the latent positive influence of superposition on the net affective state of the contributors. Following AET (Weiss and Cropanzo 1996), changes in the affective states of contributors can result in corresponding changes to the task effort they expend. In turn, these changes in task effort manifest as variations in the popularity of the project. As shown in Figure 2-2, weakening the curvilinearity of the latent influence of degree of superposition on the positive affective state of the contributors largely results in a flattening of the curve (Haans et al. 2015). Based on this understanding, we believe that for organization-owned FLOSS projects, superposition will play a smaller role in determining the popularity of the project. Hence, we hypothesize:

Hypothesis 2a: *In the context of FLOSS projects, the project ownership type moderates the relationship between the degree of superposition and project popularity such that the degree of superposition has a significantly lower influence on the popularity of the project for organization-owned projects than for individual-owned projects.*

Figure 2-2: Effect of Increase in Net Extrinsic Motivation of the Contributors when Organizations Own FLOSS Projects



Effect of superposition on the positive affective state of the contributors through the provision of autonomy

Effect of superposition on the negative affective state of the contributors due to increased frustration and loss of control

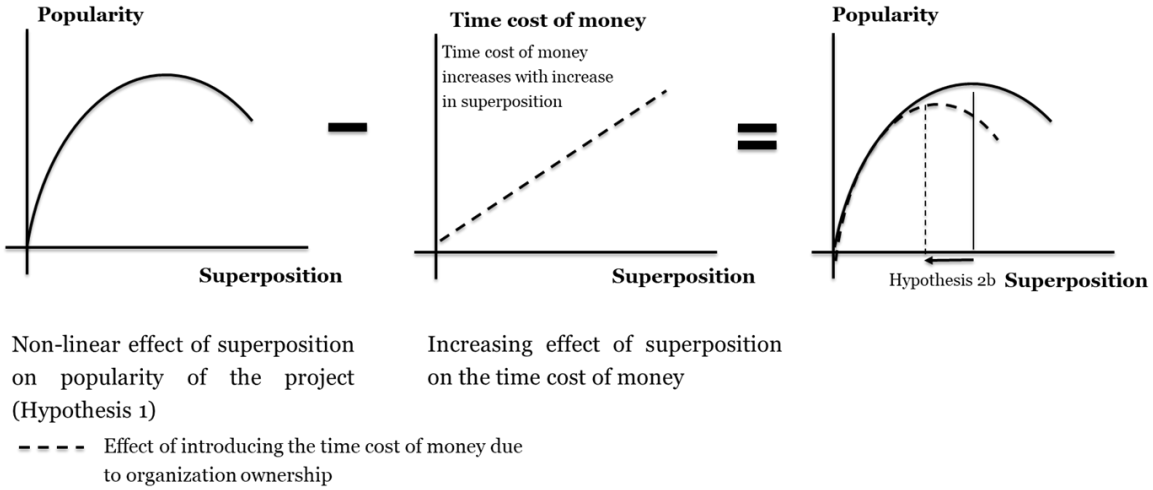
Effect of increasing the net extrinsic motivation of contributors due to organization ownership

A shift in the turning point occurs when the moderator influences the latent mechanisms in such a way that the turning point of the observed relationship changes location, *although the shape of the curve may not change* (Haans et al. 2015). When organization ownership is seen in terms of increasing the time cost of money, we can think of the moderating effect as strengthening the negative influence of superposition on the popularity of the project. This added negative influence can be approximated as a latent linear effect.[9] The inclusion of the negative linear mechanism largely results in the left shift of the turning point of the curve (Figure 2-3; Haans et al. 2015). Therefore, we contend that the degree of superposition at which the popularity of the project is at a maximum (the turning point) is significantly lower for organization-owned than for individual-owned FLOSS projects. Hence, we hypothesize:

Hypothesis 2b. *In the case of organization-owned projects, the degree of superposition at which project popularity is at a maximum (the turning point) is significantly lower than for individual-owned projects.*

[9] The time value of money is often calculated using the formula $PV = FV * (1 + r)^{-t}$, where PV is the present value, FV is the future value of the investment, r is the interest/discount rate, and t is the time for realization of the investment. Given $|r| < 1$ and $|tr| \ll 1$, we can use binomial approximation to reduce the equation to a linear form: $PV = FV - FV * t * r$. In our case, we consider the time cost of money as the value of $FV * t * r$. If we assume the time of completion of a major release t to be linearly dependent on the degree of superposition, then the time cost of money linearly increases with increases in superposition.

Figure 2-3: Effect of Time-Cost of Money when Organizations Own FLOSS Project



Construct Development

The phenomenon of superposition was conceptualized by Howison and Crowston (2014) based on their exhaustive observations of FLOSS projects. The detailed approach adopted by these researchers offered the necessary depth for a rich theorization of the concept. But to examine the influence of superposition on the popularity of a FLOSS project, we first need to translate the concept into a well-defined theoretical construct —*degree of superposition* (Suddaby 2010). While developing this construct, we try to remain true to the concept of superposition as introduced by Howison and Crowston (2014). In the subsequent subsections we elaborate on the important characteristics specific to the FLOSS model of development, namely (1) task, and (2) version and development release, before we define our focal construct —degree of superposition.

Task

A task is a unit of contribution to a FLOSS project. Howison and Crowston (2014) describe a task as a sequence of actions that leads to a change in the shared output of the project, which could be a new feature, a bug fix, updated documentation, and so forth. The actions within a task may include adding/deleting code, changing files, incorporating comments, or even reviewing code. Thus, a task can begin with messages on a FLOSS community mailing list

signaling a project need, continue with development, involve peer reviews, and subsequently result in a functional change to the application itself (Howison and Crowston 2014).

In Git based FLOSS development platforms such as GitHub, there are two ways that contributors can add (merge) their tasks to the main project code. Contributors who have push (write) access to the project can directly add their tasks to the project. Push events are typically used by the core set of developers to include certain immediate changes in the project. In contrast, contributors who do not have write access to projects will need to issue a request to "pull" their tasks into the project. Once a pull request is submitted, a contributor with push access can choose to merge the task into the main project, suggest modifications, or reject it. This pull-based software development model allows a project to be forked (cloned) into a different local branch so that changes can be worked independently of the main project code. When a set of changes is ready to be added to the main project code, the contributors create a pull request. A member of the project's core team (the contributors with "push" access) is then responsible for inspecting the changes and merging them in to the project's main project code (Gousios et al. 2014). It is important to note that once a project is forked, the local code branch might include contributions from several contributors who collaborate to implement changes before the pull request is made. Appendix 1 details the approach we used to calculate the number of tasks by identifying their associated push and pull request events using the GitHub project log data.

Version and Development Release

Each version is a snapshot of the project at a point in time. Version control is a system that maintains these snapshots of the project and records changes to a file or set of files over time and between versions. Such a system allows us to revert files back to a previous state, compare changes over time, determine the authors of the changes, and more.

In Git based version control systems like GitHub, we can define versions as the snapshot of the project available at the end of a specific day. Based on this definition, we can say that a new version of a project is created on any day that sees tasks being added to the project. The new

version differs from the previous version of the project by the tasks that have been included to it through push and pull request events created on that day. A version in this context can be compared to a development release (Michlmayr and Fitzgerald 2012). Development releases are aimed at contributors interested in working on the project or experienced users who need the most updated version of the code. Development releases need not be thoroughly tested and documented (unlike major releases) and may not ensure stability. These releases come with the latest code changes that will be merged to the project after some unit testing and/or some form of peer review. Appendix 2 details the approach we used to operationalize versions using the GitHub project log data.

Degree of Superposition

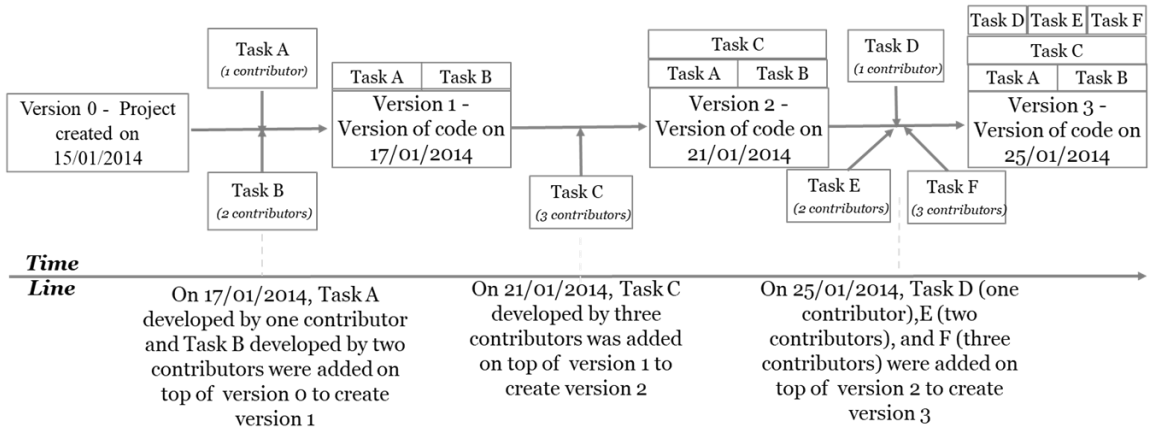
While theorizing collaboration through open superposition, Howison and Crowston (2014) contended that in the case of FLOSS projects, tasks are implemented by *individual* contributors who realize some functional payoff from the creation of the task. Over time, these individual tasks are *layered* on top of one another to incrementally build the FLOSS software. The superposed organization of individual tasks accomplishes complex tasks by the process of productive deferral.

In essence, there are three characteristics of the concept of superposition that need to be considered: (a) task work is accomplished by individual contributors, (b) individual tasks are sequentially layered on top of one another, and (c) productive deferral occurs in order to accomplish complex tasks through (a) and (b). While individual task work and sequential layering of tasks define how superposed the organization of the tasks is, productive deferral provides the mechanism that allows complex work to be accomplished by a superposed organization of tasks. In our operationalization of superposition, we capture the two aspects of superposed organization of tasks, sequential layering of tasks and individual task work. Because productive deferral can be considered as a latent mechanism that allows the orchestration of work through superposition rather than being a part of the measure of superposed task work structure, we do not include it in our operationalization of the construct.

Consequently, we define the construct *degree of superposition* as the ratio of the *total number of versions* of the FLOSS project to the total number of *individual task contributions* to the project. Based on this definition, the degree of superposition for projects increases as they adopt a more sequential and individual development approach, reaching a value of 1 when each individual task contribution represents a new version, and decreasing as projects adopt a more concurrent and collaborative development approach in which more tasks are concurrently added to a version and more contributors are collaboratively working on tasks. We can also conclude that projects which predominantly adopt productive deferral to accomplish complex tasks, will exhibit higher degrees of superposition than projects that do not adopt productive deferral mechanisms.

Consider the example illustrated in Figure 2-4. Task A, developed by one contributor, and task B, developed by two contributors, are added to version 0 of the project, which was available on January 15th. The addition of tasks A and B results in version 1 of the project on the 17th. Task C, developed by three contributors, builds on the version that was available on the 17th (version 1), creating version 2 of the project. Task D, developed by one contributor, task E, developed by two contributors, and task F, developed by three contributors, build on the version available on the 21st (version 2), creating version 3 of the project. This example has six tasks (tasks A–F), has a total of 12 individual task contributions that result in the three versions; hence the degree of superposition is 3/12, or 0.25.

Figure 2-4: An Example for the Calculation of Degree of Superposition



In this figure
- Number of versions of project = 3
- Number of individual task contributions = (1+2+3+1+2+3) = 12
- Degree of superposition = Number of versions of project / Sum total of the number of individual contributions in all tasks of the project = 3/12 = 0.25

Methodology

To understand the mechanisms through which superposition 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 for a sample of FLOSS projects 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 regressions at the project level to test the hypothesized relationships. 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 GitHub Archive database (Grigorik 2012). Since this database is large (about 432 GB, with 134

million rows for the year 2014 alone; Google 2014), we needed the bandwidth provided by a tool like Google's bigquery to run queries and export the results. We restricted our analysis to projects that were started during the first five months of 2014 to reduce the number and size of queries. Further, the event-level data collection for each project was restricted to the development work that was undertaken in the year 2014. In all, we ran more than 30,000 queries over a period of 20 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 or personal stores should be avoided. The complete list of perils identified by Kalliamvakou et al. (2014) and a summary of the methods we adopted to address these appear in Appendix 3. After filtering the dataset to address the perils, we were left with a sample of 6571 FLOSS projects, of which 3086 are individual-owned and 3485 are organization-owned that we considered for our analysis.

Measurement

GitHub offers a good environment for measuring the degree of superposition and studying its relationship to the project's popularity. More specifically, two features of GitHub make it ideal for this research. First, the granularity of the data and the availability of time stamps for all events enabled us to clearly identify the versions of each project and the task order, which allowed us to operationalize the degree of superposition. Second, the availability of detailed contributor- and project-specific data allowed us to measure the dependent variable and create a rich set of controls. The different variables that we used in this research and their measures are detailed in the following subsections.

Dependent variable. *Popularity of project* is the dependent variable of interest for this research. As mentioned in the section "FLOSS Project Success," we consider the popularity of the project to be an important measure of FLOSS project success. In GitHub, users can "star" projects in order to keep track of those that they find interesting and also to show their appreciation for these projects (Borges & Valente 2018; GitHub 2017a). The number of stars a

project has received indicates approximately the number of people who are interested in and show support for that project. Count of stars is therefore a commonly used measure for identifying popular projects in the GitHub environment: GitHub itself uses stars to identify trending projects and in its project rankings (GitHub 2017a), Jarczyk et al. (2014) use the log transformation of the number of stars as a measure of the quality and popularity of GitHub projects, and Tsay et al. (2014) use the number of stars as a measure of popularity and project establishment. In recognition of the usefulness of count of stars as a measure of popularity, we adopt it as the dependent variable to test our relationships.

Popularity measures have often been operationalized using count of downloads (e.g., Crowston et al. 2006, Grewal et al. 2006, Midha and Palvia 2012, Setia et al. 2012). In GitHub, a potential problem of using the count of downloads as a measure of FLOSS popularity is that this measure is available only for major releases of projects. Code reuse in GitHub can occur by forking a project, developing on the forked branch, and then merging that branch into a different project (without the actual download of a major release). This aspect of code reuse is not captured by the count of downloads. Further, this measure could underestimate the popularity of younger projects or projects that take longer to be packaged into a major release and distributed to end users. Because our study considers the first year in the life of a project, a relatively smaller percentage of projects in our sample see major releases for which GitHub provides a count of downloads. Hence, we adopt count of stars rather than downloads as the measure of popularity in our study.

Independent variables. To study the influences of the degree of superposition and ownership on the popularity of a FLOSS project, we employed two independent variables – *degree of superposition* and *project ownership*. The *degree of superposition* was measured as the ratio of the total number of versions of the project to the total number of individual task contributions made to the project (refer to the section "Construct Development"). The method we adopted for identifying tasks within a project is based on the understanding that a task is a sequence of actions that leads to a change in the shared output of the project (Howison and

Crowston 2014). The project logs maintained by GitHub provide detailed information regarding the timing and ownership of push and pull request events that allowed us not only to identify tasks for each project but also determine the version that they belong to. Appendices 1 and 2 detail the approaches that we adopted to identify tasks and versions leveraging the workflow of Git based FLOSS development platforms. Based on this operationalization, the degree of superposition for a project takes a value between 0 and 1. If *degree of superposition* = 1, all the project's tasks were implemented individually and added sequentially, with each individual task contribution representing a new version of the project. The degree of superposition decreases as a project adopts a concurrent development approach and approaches 0 as greater number of individual task contributions get piled onto individual versions of the project.

To study the moderating influence of project ownership on the relationship between the degree of superposition and the popularity of a FLOSS project, we also used a flag, *project ownership*, 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 identify if a project is owned by an organization or an individual.

Control variables. GitHub maintains detailed project- and user-level data, which allows the introduction of a rich set of control variables. Consequently, we identified two kinds of control variables for our analysis: contributor/owner characteristics and project characteristics.

Contributor/owner characteristics. We identified three measures to control the influence of contributor and project owner characteristics on the relationship between the degree of superposition and the popularity of a project. 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 popularity of the project (Stewart, Ammeter, et al. 2006; Subramaniam et al. 2009). At the same time, as the number of contributors increases, there is a tendency to adopt a more concurrent form of development to coordinate multiple contributors. In addition to the number of contributors, the *average number of contributions per contributor* is also expected to positively influence the popularity of a project (Subramaniam et al. 2009). Hence, we also control for the average number of commits per contributor, where a commit is any change or set of changes that is locally saved to file (GitHub 2017b). Lastly, we control for the *experience of the project owner* in terms of the number of FLOSS projects the owner participates in. 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 FLOSS project.

Project characteristics. We identified eight measures to control for different project characteristics. First, 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). Second, 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. Third, we controlled for the *average task size*, measured as the average number of commits made within the project's tasks. 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 large tasks will

have a higher need for co-work than projects with smaller tasks. Fourth, popular FLOSS projects see continuous enhancement and maintenance of the code, which results in multiple stable versions delivered one after another (Raymond 1998). The *project completion flag* identifies whether the project ended within the data collection period (i.e., in 2014) and became inactive. Because popular projects show continuous activity, we expected project completion or a short project life to be negatively correlated to the popularity of the project. Fifth, 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 compatible license scheme (Singh and Phelps 2013). 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 (e.g., Colazo et al. 2005, Stewart et al. 2006). Hence, we used the flag *restrictive license regime* to control the effect of the type of license (restrictive vs. permissive) on the popularity of a project. Sixth, we controlled for the *average idle time* between tasks in a project. The average idle time between tasks in a project is a measure of the activity level of the project. Popular FLOSS projects have been found to show high activity, with releases occurring early and often (Raymond 1998). This relationship has also been empirically observed, with project activity found to positively impact popularity of the project (Subramaniam et al. 2009). We also include two fixed effects: the fixed effects of the *main programming language* and the fixed effects of the *month of creation*. 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 programing languages, with languages more conducive to modular architecture displaying greater ease of coordination (Baldwin and Clark 2006). We controlled for these effects by including a dummy variable for each programming language used in our sample. Lastly, time is expected to influence the

nature of routines associated with FLOSS projects (Lindberg 2015). Since our analysis includes projects that were started during the first five months of 2014, it was necessary to include the fixed effects of the month of creation to control for the influence of time on the hypothesized relationships.

Analysis

We include two regression models in our analysis. The main regression model is based on ordinary least squares (OLS) with the log transformation of the number of stars as the dependent variable. The second model, based on the negative binomial approach, addresses the count nature of the dependent variable. Hypothesis 1 predicts that the degree of superposition has an inverted U-shaped relationship with the popularity of a project. To test this hypothesis, we included the square of the degree of superposition and created a polynomial regression model. Hypothesis 2 predicts that the project ownership moderates the relationship between the degree of superposition and the popularity of the project. To test this hypothesis, we introduced the interactions of project ownership with the degree of superposition terms in the regression models. The following section details the results of the regression models and the checks we employed to ensure the validity of the results.

Models and Results

From Table 2-2, which provides the means, standard deviations, and correlation coefficients for the variables used in the analyses, we can observe a strong positive skew in some of the variables. In particular, the residuals of the variables *stars*, *size of project*, and *average task size* show significant positive skew in their distribution.

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

	Variable	Mean	Std. Dev.	1	2	3	4	5	6	7	8	9	10	11	12
1	Stars	194.69	643.09												
2	Degree of superposition	0.705	0.172	-0.084**											
3	Degree of superposition square	0.526	0.226	-0.084**	0.986**										
4	No of programming languages	3.171	3.003	0.043**	-0.049**	-0.028									
5	Total contributors	8.859	15.932	0.198**	-0.185**	-0.17**	0.267**								
6	Size of project	2.164	22.257	-0.003	-0.01	-0.005	0.269**	0.199**							
7	Average commits per contribut	51.416	103.29	-0.026*	0.022	0.038**	0.305**	0.061**	0.109**						
8	Average task complexity	2.513	14.946	-0.011	-0.048**	-0.047**	0.021	0.077**	0.008	0.032**					
9	Owner flag	0.53	0.499	-0.045**	-0.198**	-0.192**	0.141**	0.130**	0.047**	0.064**	0.013				
10	Project inactivity flag	0.877	0.328	0.077**	0.083**	0.074**	0.087**	0.081**	0.025*	0.077**	0.007	0.094**			
11	Restrictive license regime	0.114	0.318	-0.036**	0.028	0.033**	0.142**	0.054**	0.067**	0.132**	0.014	0.008	0.013		
12	Owner experience	67.204	134.25	0.06**	-0.120**	-0.124**	-0.046**	0.055**	-0.02	-0.056**	-0.010	0.035**	0.035**	-0.095**	
13	Average idle time	15.678	21.167	-0.082**	0.098**	0.080**	-0.199**	-0.147**	-0.041**	-0.208**	-0.020	-0.164**	-0.02	-0.087**	0.030*

* $p < 0.05$; ** $p < 0.01$.

In order to normalize the positively skewed data, statistical texts commonly recommend the use of log transformations (Carte and Russel 2003; Singh et. al. 2013). After log transformation of the variables *stars*, *size of project*, and *average task size*, their residuals were found to closely match a normal distribution. Moreover, previous research that uses the number of GitHub stars as an outcome measure recommends using its log transformation. For example, Jarczyk et al. (2014) used the log transformation of the number of stars to measure project quality because of the power-law distribution that GitHub stars follow.

Table 2-3 provides the results of the regression models we employed. Models 1a to 1c provide the results of the stepwise log-linear OLS regression analysis. Models 2a to 2c provide the results of the negative binomial regression analysis with the number of stars as the dependent variable. Because the inclusion of higher-order terms (the square of the degree of superposition and the interaction terms) in the regression equation may lead to computational errors due to multicollinearity (Aiken and West 1991), we also tested the results for the mean centered model. In the mean centered model, multicollinearity as indicated by the variance inflation factor (VIF) is less than 5 for all constructs. This indicates that the results are not confounded by multicollinearity (Hair et al. 2010). 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).

Table 2-3: Results of Moderated Regression Analysis

	Model - OLS DV : Log(Stars)			Model - Negative Binomial DV : Stars		
	Model 1a - Control Variables	Model 1b - Main Effects	Model 1c - Interaction Effects	Model 2a - Control Variables	Model 2b - Main Effects	Model 2c - Interaction Effects
No of programming languages	-0.013 -0.17	-0.013 -0.18	-0.008 -0.37	-0.007 -0.55	-0.009 -0.45	-0.007 -0.5
Total contributors	0.014 (0.00)**	0.013 (0.00)**	0.014 (0.00)**	0.037 (0.00)**	0.035 (0.00)**	0.038 (0.00)**
Log(Size of project)	0.011 -0.38	0.018 -0.16	0.043 (0.00)**	0.035 -0.09	0.044 (0.03)*	0.082 (0.00)**
Average commits per contributor	-0.0004 (0.01)*	-0.0004 (0.03)*	-0.0004 (0.01)**	-0.001 (0.00)**	-0.001 (0.01)**	-0.001 (0.00)**
Project inactivity flag	0.832 (0.00)**	0.827 (0.00)**	0.898 (0.00)**	0.918 (0.00)**	0.916 (0.00)**	0.876 (0.00)**
Restrictive license regime	-0.332 (0.00)**	-0.333 (0.00)**	-0.351 (0.00)**	-0.371 (0.00)**	-0.391 (0.00)**	-0.427 (0.00)**
Log(Average task complexity)	-0.07 (0.01)*	-0.098 (0.00)**	-0.093 (0.00)**	-0.117 (0.01)**	-0.134 (0.00)**	-0.133 (0.00)**
Owner experience	0 -0.36	0 -0.51	0 -0.32	0.0004 (0.04)*	0.0005 (0.05)*	0.001 (0.02)*
Average idle time	-0.005 (0.00)**	-0.005 (0.00)**	-0.007 (0.00)**	-0.01 (0.00)**	-0.01 (0.00)**	-0.01 (0.00)**
Degree of superposition (β_1)		2.876 (0.00)**	5.853 (0.00)**		3.188 (0.03)*	7.588 (0.00)**
Degree of superposition squared (β_2)		-2.453 (0.00)**	-4.682 (0.00)**		-2.622 (0.01)*	-5.85 (0.00)**
Project ownership flag (β_3)			0.889 (0.04)*			1.665 (0.03)*
Degree of superposition X Project ownership flag (β_4)			-4.398 (0.00)**			-6.726 (0.00)**
Degree of superposition squared X Project ownership flag (β_5)			2.928 (0.00)**			4.508 (0.01)**
Fixed effects of month of creation	Yes	Yes	Yes	Yes	Yes	Yes
Fixed effects of main programming language	Yes	Yes	Yes	Yes	Yes	Yes
R-Square	0.133	0.137	0.1797	-	-	-
ΔR-Square		.004**	0.043**	-	-	-
AIC	3.634	3.63	3.58	11.559	11.555	11.491
N	6,571	6,571	6,571	6,571	6,571	6,571

* $p < 0.05$; ** $p < 0.01$.

Hypothesis Linking the Degree of Superposition and Project Popularity

Hypothesis 1 predicts a nonlinear relationship between the degree of superposition and the popularity of a FLOSS project. This nonlinear relationship resembles an inverted U-shaped curve, with the turning point representing the maximum value of project popularity. To test

hypothesis 1, we regressed the dependent variable (project popularity) on the independent variable (degree of superposition) and its squared term. It is essential to retain the linear term of the degree of superposition (Aiken and West 1991), as leaving it out would assume that the turning point occurs at 0 degrees of superposition (Haans et al. 2015). We followed the three-step procedure proposed by Lind and Mehlum (2010) to confirm the presence of an inverted U-shaped relationship in Model 1b. In the first step, we checked the direction and significance of the coefficient of the squared term of the degree of superposition (β_2). From Table 2-3, Model 1b, we can see that β_2 $(-2.453,\ p < 0.01)$ is negative and significant, a necessary condition for the existence of an inverted U-shaped relationship (Haans et al. 2015; Lind and Mehlum 2010). In the next step, we checked whether the slopes at both ends of the data range (i.e., when the degree of superposition = 0 or 1) are significant and in the directions characterizing an inverted U. For an inverted U-shaped relationship to exist, the slopes at the low end of the range of the independent variable (i.e., when the degree of superposition is 0) should be positive and significant, while those at the high end of the range (i.e., when the degree of superposition is 1) should be negative and significant (Haans et al. 2015; Lind and Mehlum 2010). We tested Model 1b to confirm that the slope when the degree of superposition is 0 is positive and significant ($\frac{dy}{dx}_{x=0} = +2.866, p < 0.01$) and that when the degree of superposition is 1, the slope is negative and significant ($\frac{dy}{dx}_{\mathrm{x=1}} = -2.023, p < 0.01$). In the last step, we checked whether the turning point lies within the data range for the degree of superposition (i.e., the turning point should lie between the values 0 and 1 for the degree of superposition). The turning point for the quadratic equation is given by – $(\beta_1 / 2\beta_2)$[10]. Since our sample was large, the delta method (Rao 1972) could be used to calculate the confidence interval. Using this method, the test for the 99% confidence interval for the turning point confirmed that it lies between 0 and 1 $(0.530 < (-\beta_1 / 2\beta_2) = 0.586 < 0.642)$. The results of

[10] For a quadratic equation of the form, $y = \mathrm{a} * x^2 + \mathrm{b} * x + \mathrm{c}$, the turning point is given by $-b\ /\ (2 * \mathrm{a})$

these three tests provide the necessary and sufficient conditions (Lind and Mehlum 2010) to support Hypothesis 1. That is, the popularity of a FLOSS project increases with an increase in the degree of superposition until it reaches a maximum (the turning point), after which any further increase in the degree of superposition will result in a decrease in the popularity of the project.

Hypothesis Linking Ownership and Project Popularity

Hypothesis 2 predicts a moderating role of owner type on the relationship between the degree of superposition and the popularity of a project. To test this hypothesis, we included the interaction of owner type with the degree of superposition and its squared term in Model 1c (Table 2-3). We found that inclusion of the interaction terms helps explain significant variance in the dependent variable over and above the main effects ($\Delta R^2 = 0.043, p < 0.01$).

Hypothesis 2a predicts that the influence of the degree of superposition on the popularity of the project is lower for organization-owned FLOSS projects than for individual-owned FLOSS projects. This interaction effect is expected to flatten or reduce the curvature of the inverted U-shaped relationship. To test this interaction effect, it is sufficient to check the significance and direction of the coefficient of the interaction with the squared term of the degree of superposition (β_5; Haans et al. 2015). From Model 1c (Table 2-3), we can see that β_5 is positive and significant (2.928, $p < 0.01$), confirming that organizational ownership tends to flatten the inverted U-shaped relationship. Thus, Hypothesis 2a is supported, and we can conclude that organizational ownership of FLOSS projects tends to reduce the influence of the degree of superposition on the popularity of the project. Figure 2-5 plots the relationship between the degree of superposition and the popularity of the project for individual- and organization-owned FLOSS projects based on the results of Models 1c (Table 2-3). The flattening effect can be observed in the plots, with organization-owned FLOSS projects exhibiting a relatively flat curve compared to individual-owned FLOSS projects.

Hypothesis 2b predicts that the turning point for organization-owned projects occurs at a lower degree of superposition than for individual-owned projects. To test this hypothesis, we derived

the turning points for individual- and organization-owned FLOSS projects and tested for their equality (Haans et al. 2015). Since the relationship is quadratic, the turning points are given by $-\frac{\beta_1}{2*\beta_2}$ for individual-owned projects (z = 0) and $-\frac{\beta_1+\beta_4}{2*(\beta_2+\boldsymbol{\beta_5})}$ for organization-owned projects (z = 1)[11]. The generalized Wald type test for this nonlinear inequality rejects the null hypothesis that the turning points are equal ($chi^2 = 11.68,\ p < 0.01$; Model 1c; Phillips and Park 1988). This provides support for hypothesis 2b. The shift of the turning point can be observed in the plots depicted in Figure 2-5. From the plots, we can observe that the degree of superposition at which popularity is at a maximum for organization-owned projects (0.415) is considerably lower than that for individual-owned projects (0.625).

[11] The full regression model with interaction (Table 2-3, Model 1c) can be represented by:

$$\boldsymbol{y = \beta_1 * x + \beta_2 * x^2 + \beta_3 * z + \beta_4 * x * z + \beta_5 * x^2 * z + \beta_i * Controls_i + Const},$$

where

$y = popularity\ of\ the\ project,$

$x = degree\ of\ superposition,$

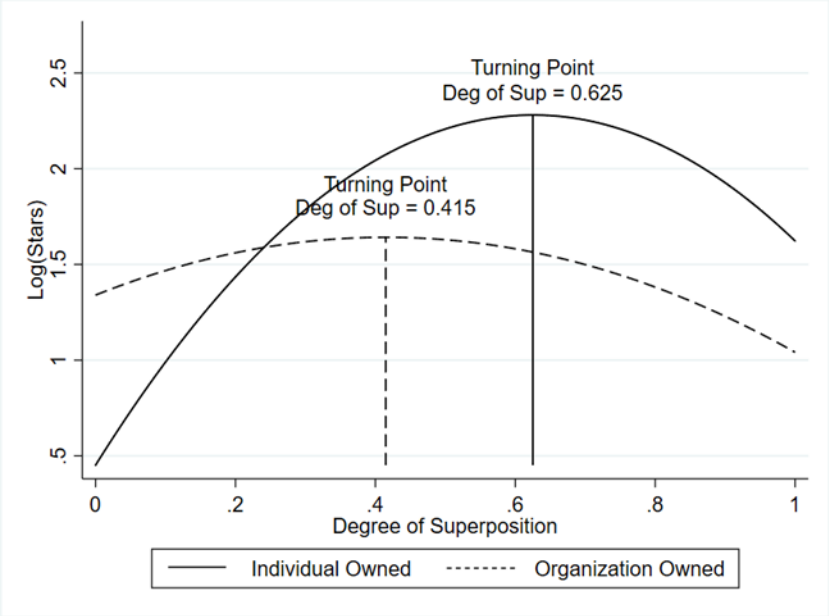
$z = project\ ownership\ flag$

$(1\ if\ organization\ owned\ and\ 0\ if\ individual\ owned),$

$\beta s = regression\ coefficients, and$

In this quadratic equation, the turning point is given by $-\frac{\beta_1+\beta_4*z}{2*(\beta_2+\boldsymbol{\beta_5}*\boldsymbol{z})}$, with z= 0 representing the turning point for individual-owned projects and z= 1 representing the turning point for organization-owned projects.

Figure 2-5: Plots of the Relationship Between Degree of Superposition and Popularity of the Project for Individual- and Organization-Owned FLOSS Projects



Negative Binomial Regression Model

Taking into account the overdispersed nature of the number of stars, we estimated the negative binomial model (Model 2, Table 2-3; see Cameron and Trivedi 2013, Chapter 4). In this model, we used the original scale for the number of stars (not log-transformed) as the dependent variable and regressed it against the quadratic terms of the degree of superposition and their interactions with the ownership flag. We followed the three-step procedure proposed by Lind and Mehlum (2010) to test Hypothesis 1, 2a and 2b. We find support for the three hypotheses in the negative binomial model. Since the model predicts the log of the expected number of stars as a function of the predictor variables, the interpretations of the coefficient are like the log-linear OLS model (Model 1c). That is, for a one-unit change in the predictor variable, the number of stars is expected to change by the respective regression coefficient, given that the other predictor variables in the model are held constant.

In addition to the negative binomial model, the following section presents three robustness tests: (a) tests for model specification, (b) test for the dependent variable and, (c) test for the nature and treatment of data. In tests for model specification, we test if the relationship is

indeed quadratic and not due to a few outliers. As a test for the dependent variable, we include survival analysis to provide an independent view of project success that is not captured by counts of stars. Lastly, we included a falsification test to show that the observed relationships are more likely due to the theorized mechanisms rather than an outcome of the nature of the sample and the approach we adopted to operationalize the constructs.

Supplementary Analyses for Robustness

Tests for Model Specification

Test for cubic specification. As a robustness test for the model specification, we confirmed that the relationship is indeed U-shaped rather than S-shaped (Haans et al. 2015) by including the cubic term (x^3) of the degree of superposition to Model 1b (Table 2-3). We found that inclusion of the cubic term does not increase the model fit ($\Delta R^2 = 0.000,\ p > .10$) and that the coefficient of the cubic term is not significant ($p > .10$).

Test for outliers. In order to guard against the possibility that a few extreme observations are driving the results, it is common for researchers to exclude outliers from the sample and then re-estimate the model (Haans et al. 2015). We adopted Cook's distance (Cook 1977) to eliminate outliers from Models 1b and 1c (Table 2-3). On re-estimating the models after the exclusion of outliers, we found that our conclusions remained unchanged.

Test for the Dependent Variable – Survival Analysis

In their empirical analysis of FLOSS projects' success, Crowston et al. (2006) found that the common measures used for success, such as popularity, were correlated to each other but that project lifespan was unique as it did not correlate to the other measures of success. Given this finding, the inclusion of survival analysis provides an independent view of project success that is not captured by counts of stars. In the survival analysis, we tested our hypotheses using the likelihood that a FLOSS project would become inactive as the dependent variable (Hosmer et al. 2008b). In this model (Table 2-4), the dependent variable is a combination of a binary

variable that indicates whether a project became inactive within three years of its inception and a continuous variable that measures the number of active days for the project.

Table 2-4: Results of The Survival Analysis

	Model - Survival Analysis		
	DV :Likelihood of Project Inactivity		
	Model 3a - Control Variables	Model 3b - Main Effects	Model 3c - Interaction Effects
No of programming languages	0.004 -0.68	0 -0.9	-0.001 -0.92
Total contributors	-0.031 (0.00)**	-0.036 (0.00)**	-0.034 (0.00)**
Log(Size of project in kilobytes)	-0.061 (0.00)**	-0.06 (0.00)**	-0.051 (0.00)**
Average commits per contributor	-0.003 (0.00)**	-0.003 (0.00)**	-0.003 (0.00)**
Restrictive license regime	0.03 -0.65	0.038 -0.56	0.034 -0.61
Log(Average task complexity)	0.059 -0.06	0.027 -0.42	0.04 -0.23
Owner experience	0 -0.32	0 -0.54	0 -0.46
Average idle time	-0.002 -0.08	-0.002 -0.11	-0.002 -0.1
Degree of superposition (β_1)		-4.13 (0.00)**	-7.196 (0.00)**
Degree of superposition squared (β_2)		2.673 (0.00)**	4.613 (0.00)**
Project ownership flag (β_3)			-1.785 (0.00)**
Degree of superposition X Project ownership flag (β_4)			4.402 (0.00)**
Degree of superposition squared X Project ownership flag (β_5)			-2.793 (0.01)*
Fixed effects of month of creation	Yes	Yes	Yes
Fixed effects of main programming language	Yes	Yes	Yes
R-Square	-	-	-
ΔR-Square	-	-	-
AIC	51851.89	51854.33	51772.97
N	6,571	6,571	6,571

* $p < 0.05$; ** $p < 0.01$.

We use the semi-parametric Cox regression model for the survival analysis, since it relaxes the specification requirement for the baseline hazard function (Hosmer et al. 2008b). Since the dependent variable in this model is an inverse measure of project success, Hypothesis 1 predicts a noninverted U-shaped relationship between likelihood of project inactivity and

degree of superposition. Model 3c (Table 2-4) provides the results of the survival analysis with the likelihood of inactivity of the project as the dependent variable. In this model, we find support for Hypotheses 1 and 2a. However, a left shift of the turning point when organizations own FLOSS projects (Hypothesis 2b) was not observed. We discuss the implications of these results in the discussion section of this paper.

Test for Nature and Treatment of Data – Falsification Test

The mechanisms through which superposition influences the popularity of a project are restricted to FLOSS-like environments. In these environments, superposition minimizes the interdependencies amongst contributors, thereby satisfying their need for autonomy (Howison and Crowston 2014), which in turn increases their task efforts (Ke and Zhang 2010). Thus, in FLOSS-like environments, superposition characterized by sequential and individual development is more effective in motivating contributions than is engagement in concurrent development. In non-software-development environments, where the primary activity is information gathering and storage, we do not expect this mechanism to be prominent. This is because in non-software-development projects such as creating lists, collating best practices and examples, and storing information, co-work does not undermine the autonomy afforded to the contributor since there are limited interdependencies between individual tasks. Thus, in these environments we do not expect the hypothesized inverted U-shaped relationship to exist between the degree of superposition and the popularity of a project. In the falsification test, we use non-software-development projects such as personal stores, mirrors of other projects, lists, and other examples obtained from GitHub. In this sample, the presence of an inverted U-shaped relationship between degree of superposition and project popularity would indicate that either (a) the nature of the data and the operationalization of the constructs could cause an inverted U-shaped relationship or (b) there exist interdependencies between tasks in the sample that allow the mechanisms of superposition to operate in these non-software-development projects. By reading the project descriptions, we excluded projects that could potentially have task-level interdependencies (e.g., editing wikis, tutorials) from the test, as

they could potentially adopt a superposed work-breakdown structure. After further elimination of projects that were either too small or had very few contributors, we were left with a sample of 260 non-software-development projects.

Table 2-5: Results of the Falsification Test

	DV : Log(Stars)	
	Model 4a - Linear Model	Model 4b - Quadratic Model
Total contributors	0.007 (0.04)*	0.009 (0.04)*
Log(Size of project)	0.086 0.205	0.089 0.187
Average commits per contributor	-0.002 (0.02)*	-0.002 (0.04)*
Project completed flag	0.617 (0.03)*	0.572 (0.05)*
Restrictive license regime	0.266 0.436	0.354 0.305
Log(Average task complexity)	0.035 0.821	0.021 0.894
Owner experience	0 0.655	0 0.505
Average idle time	0.005 0.394	0.007 0.271
Project ownership flag	-1.049 (0.00)**	-1.059 (0.00)**
Degree of superposition (β_1)	-2.233 (0.00)**	1.37 0.638
Degree of superposition squared (β_2)		-2.74 0.194
Fixed effects of month of creation	Yes	Yes
R-Square	0.2342	0.2395
ΔR-Square		0.005
N	260	260

* $p < 0.05$; ** $p < 0.01$.

Table 2-5 lists the results of the regressions carried out with this sample. Model 4a is a linear regression model that tests the linear relationship between the degree of superposition and the popularity of the project. The results of this model show that the coefficient of the degree of superposition term is negative and significant within the 95% confidence interval (-2.233, $p <$ 0.05). Model 4b includes the quadratic term of the degree of superposition to test for a curvilinear relationship between the degree of superposition and the popularity of the project. We found that inclusion of the quadratic term does not increase the model fit ($\Delta R^2 = 0.005$, $p > .10$) and that the coefficient of the quadratic term is not significant ($p > .10$). These results

support the conclusion that the relationship between the degree of superposition and the popularity of the project for this sample of non-software-development projects is linearly decreasing, as indicated by Model 4a. This means that concurrent task organization was preferred over the superposed development approach for this sample of non-software-development projects. Because the mechanisms of superposition that result in our curvilinear hypothesis are restricted to FLOSS development projects, the results of this falsification test provide additional support for the observed relationships in the main models (Table 2-3, Models 1b and 1c) being due to the theorized mechanisms rather than an outcome of the nature and treatment of the data.

Discussion and Contributions

Despite the FLOSS development model's increasing prominence in practice and research, the antecedents to its success have not been completely understood. In particular, the sociotechnical aspect of its work structures and the role that this plays in a project's ability to attract users and developers has received only limited attention from previous researchers (Howison and Crowston 2014). Considering the steady shift towards FLOSS in recent years and its expected acceleration in the future, clarifying the influence of its work structures is important for better understanding the phenomenon and informing project owners about how they can benefit from adopting this unique model of development. As a step forward in this direction, our research sought to unearth the mechanisms through which superposition influenced the success of individual- and organization-owned FLOSS Projects.

Using a large sample of FLOSS projects owned by individuals and a wide range of organizations, we find that the emergence of sequentially layered and individual task work, referred to as the superposed organization of work, can enhance the popularity of a project. Superposed organization of work encourages task effort by satisfying intrinsically motivated contributors' need for greater work autonomy (Howison and Crowston 2014). The contributors' increased task efforts may be invested in adding new functionalities and improving the quality of the project, resulting in an overall increase in the popularity of the

project. For individual-owned projects, as the degree of superposition increases from 0 to its turning point, the popularity, measured as the average number of stars for the project is found to increase by more than five times, from 11.53 to 71.79, holding everything else constant, while for organization-owned projects it is found to increase from 28.26 to 38.08 (see Model 1c, Table 2-3). As mentioned in the initial part of the paper, an increase in popularity signifies project success — because greater popularity implies a larger community of users participating in bug identification and improvement of ideas (Crowston, Howison, et al. 2006). And some of these users, over time, may progress to become active developers for the project (Krishnamurthy 2002, Subramaniam et al. 2009).

Although our study confirms the proposition laid out by Howison and Crowston (2014) by showing that superposed organization of tasks has important benefits in terms of enhancing the success of a project, it also indicates that the degree of superposition exhibits decreasing returns to scale and eventually dampens project success. We theorize that as the degree of superposition increases, delays due to the avoidance of concurrent work and co-work may induce a sense of frustration and loss of control for the contributors, which tends to influence the success of the project negatively. In our study, the decreasing returns are evident for both individual- and organization-owned projects,[12] both of which show an overall negative influence of the degree of superposition beyond the turning point. For individual-owned projects, the average number of stars is found to decrease from 71.79 to 37.12 as the degree of superposition increases from its turning point to its highest value of 1, keeping everything else constant, while this number decreases from 38.08 to 20.78 for organization-owned projects. These findings have strong implications for both theory and practice.

[12] For individual-owned projects: $\frac{dy}{dx}_{x=0} = +5.85$, $\frac{dy}{dx}_{x=0.25} = +3.51$, $\frac{dy}{dx}_{x=Turning\ Point} = 0$, $\frac{dy}{dx}_{x=0.75} = -1.17$, $\frac{dy}{dx}_{x=1} = -3.52$, and for organization-owned projects: $\frac{dy}{dx}_{x=0} = +1.44$, $\frac{dy}{dx}_{x=0.25} = +0.57$, $\frac{dy}{dx}_{x=Turning\ Point} = 0$, $\frac{dy}{dx}_{x=0.75} = -1.18$, $\frac{dy}{dx}_{x=1} = -2.06$, where y = log(stars) and x = degree of superposition

Implications

Our research contributes to IS and organization theory in three ways. First, our study advances the existing literature on motivation (e.g. Ke and Zhang 2010, von Krogh et al. 2012, Ryan and Deci 2000) and work structures (e.g. Howison and Crowston 2014, Lindberg et al. 2016) in FLOSS projects, as it takes a significant step in establishing the role of work organization as a key driver for contributors' motivations and also as an antecedent to project success. Sociotechnical mechanisms associated with work structures have been largely neglected in studies of communities of practice. For example, the theory of network governance (Jones et al. 1997) identifies four social mechanisms that emerge in communities of practice in-order to overcome coordination challenges - imposing access restrictions, collective sanctions, reputational identity and creating a macroculture. By progressing the research agenda initiated by Howison and Crowston, (2014) and Lindberg et al. (2016), we demonstrate that work structures can perhaps be a fifth mechanism of network governance that not only helps overcome coordination issues but also motivates contributors in FLOSS-like environments. Further, although SDT has predominantly been used to study the motivation of the contributors in FLOSS environments (e.g., Ke and Zhang 2010; Lakhani and Wolf 2005; Roberts et al. 2006), relatively few have tried to link individual motivation to the success of the project (von Krogh et al. 2012). By providing theoretical arguments and finding empirical support for an inverted U-shaped relationship, our study shows that the complete picture of individual motivation in the context of task work often goes beyond the three psychological needs postulated by SDT. From the standpoint of motivational theories (e.g. SDT), our study cautions against assuming that the mechanisms operating at the individual level directly scale-up to the team or project level. Our findings suggest that the difficulties in scaling these mechanisms up to the project level manifest as boundary conditions describing the application of theories from one level of analysis to another. Understanding these boundary conditions and their implications for project success can be beneficial as it enriches the response to the question raised by von Krogh et al. (2012, p. 650): "How and why do OSS developers produce high-quality software when they do?".

Second, our research advances the literature surrounding organizational participation in FLOSS projects (Capra et al. 2011; Fitzgerald 2006; Spaeth et al. 2015; Stewart, Ammeter, et al. 2006; Wagstrom 2009) by enhancing our understanding of how organizational participation influences FLOSS projects in general and their work structures in particular. Practically, organizations seek to benefit from the strengths of open collaboration but are unsure about how to integrate it with their own strengths. By trying to introduce controls and/or speeding up development, organizational involvement can "kill the goose that lays the golden egg" by undermining the community process. This leads to a trade-off between openness and control (Engeström 2007; Jarvenpaa and Lang 2011; Teigland et al. 2014), that when effectively managed can lead to the success of FLOSS projects (Teigland et al. 2014). Our inquiry is one of the first to understand this trade-off in the realm of work structures and the nature of the ownership of FLOSS projects. With organizational ownership, the increased net extrinsic motivation of the contributors and the higher time cost of money undermine the need for openness and autonomy of work, allowing the project owner to exert a greater influence the relationship between superposed work structures and project success. The extent to which the organization owner can influence the relationship between superposed work structures and project success depends on the models of organizational involvement. The greater the organization owner is willing to invest in the coding model of engagement by contributing code and employees' time, the lesser will be the overall influence of superposed work structures on the success of the project. The greater the organization owner invests in the support activities of the FLOSS project, greater is the time-cost of money —creating a higher need for efficient development practices in-lieu of superposition.

Third, our work develops a clear construct for the concept of superposition that can be used for future theory development anchored around this construct. A well-defined theoretical construct with delineated scope conditions is essential to the process of building strong theory (Suddaby 2010; Sutton and Staw 1995). As a step towards this goal, we have defined and operationalized the construct -*degree of superposition* and examined its behavior in relation

to project success under different contextual conditions, i.e., at low and high degrees of superposition and under different ownership types and models of organizational engagement. In doing so, we have tried to establish the scope conditions associated with space and value for the construct (Suddaby 2010). Further, this construct can be easily operationalized using task-specific data that are available in the change logs for projects. We believe that this construct will not only help researchers to advance the theory of superposition and unearth new relationships but can also be used as a tool to measure and monitor project work structures.

Limitations and Directions for Future Research

Although we have carefully tried to ensure theoretical and methodological rigor in this research, there are certain limitations that need to be considered. First, our choice of GitHub as a source of data was based on its popularity among programmers, its integrated social features, and the availability of detailed metadata. While GitHub offers a good dataset for the purposes of this study, certain perils are associated with its use (Kalliamvakou et al. 2014). To minimize the effect of these perils, we introduced multiple controls, filters, and manual checks. The details of the perils in using this dataset and the checks we adopted to overcome them are listed in Appendix 3. Second, our operationalization of tasks and versions is tuned to Git based platforms like GitHub and limited to the nature of data made available by that platform. When it comes to operationalizing tasks, we have taken special care to ensure that each identified task is unique (i.e., we have avoided double counting of tasks) and made a significant change to the shared output of the project (see Appendix 1 for details). However, it is possible that some events involved a greater amount of effort and implemented larger changes to a project than others. Ideally, each event should be manually analyzed to determine whether it made a significant change to the shared output of the project before qualifying it as a task. But this is practically very difficult, because many active projects have several hundred events in a year. However, to minimize the impact of varying task sizes across projects, we included the control variable *average task size of the project* in our regression models. When it comes to versions, we operationalize it as the state of the project at the end of a specific day of activity (see

Appendix 2 for details). While this operationalization allows us to generalize the construct and leverage the workflow of Git based FLOSS development platforms, the actual meaning of versions may differ across platforms and sometimes even across projects. This problem is exacerbated by the fact that the literature is overloaded with multiple terms that conflate the meaning of versions. For example, major release, minor release, development release, user version, and patches. Despite these issues, we have retained the term "version" and presented one approach to operationalize it. Future studies could tune their approach for operationalizing versions to one that is more suitable for the nature of workflow adopted by the platform under study. As a robustness test to confirm that the observed results were not due to the way we have operationalized the constructs, we included a falsification test (results provided in the subsection titled "Supplementary Analyses for Robustness") to show that the hypothesized relationships are not significant in projects that are unlikely to see the mechanism of superposition in action. Third, we used counts of stars as the measure of success because it captures how popular a project is. In their empirical analysis of FLOSS project success, Crowston et al. (2006) found that the common measures used for success, such as popularity, are correlated to each other but that project lifespan was unique in not correlating to the other measures of success. Given this multidimensional nature of the success construct, we included survival analysis to provide an independent view of project success not captured by counts of stars. The result of the survival analysis provides support for Hypotheses 1 and 2a but is not significant for the shift in the turning point proposed in Hypothesis 2b (see Appendix 1 for details). The nonsignificance may be because the meaning of the lifespan of a project might be slightly different for organization-owned FLOSS projects—that is, while successful individual-owned projects are expected to remain active for a prolonged period, organization-owned projects may come with specific end goals and timelines. The mixed results we have seen for Hypothesis 2b call for a deeper inquiry into the temporal effect of the work structure, which might exhibit changes across years and along the life cycle of a project (Lindberg 2015). The temporal effect of superposition provides a potential area for future research, which can try to unearth the scope conditions for the construct associated with time (Suddaby 2010).

Understanding the temporal effect can deepen our understanding about the impact that the life-cycle stage of a FLOSS project has on its work structures and how the impact may differ in large, mature projects like Eclipse, Firefox, and GNOME. Fourth, one of our goals in taking up this research was to call attention to the importance of sociotechnical mechanisms invoked by the work structures in FLOSS projects. Along this line, we have tried to enrich the theory of superposition by establishing a better context and identifying boundaries to its influence. However, Lindberg et al. (2016) identified unresolved developer and developmental interdependencies that tend to be resolved through two unique sociotechnical mechanisms i.e. direct implementation and knowledge integration. While these mechanisms seem to address dependencies outside the purview of superposition, they are possibly related to the productive deferral mechanism of superposition because they serve the same purpose of addressing complexities in coordinating development work. Since productive deferral has been treated as a latent mechanism in this study, a more thorough development of this concept may be necessary to integrate the works of Howison and Crowston (2014) and Lindberg et al. (2016) so as to provide a full picture about the role of work structures in FLOSS projects. Although these linkages remain unresolved and require deeper investigation, our study contributes to the literature by taking a modest step forward in enriching our understanding of work structures in FLOSS development and the mechanisms through which superposition influences FLOSS project success in individual as well as organizational contexts.

3 Essay 2: Team Composition and Governance

Abstract

This essay examines the influence of source code access restrictions on the likelihood of survival of open source software projects. Building on the theories of coordination and network governance, we study the influence of access restrictions in mitigating coordination challenges and overcoming the variance in contributors' expectations. 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. Using a Cox proportional hazard model, we demonstrate that the relationship between the proportion of contributors (who are given write access to the source code) and the survival of the project, is moderated by the nature of project ownership — individual versus organization owned projects. Interestingly, the observed moderation is a crossover interaction effect that changes from negative for individual owned projects to positive for organization owned projects. This research advances our understanding about contributor roles, access restrictions, and organizational participation in open source environments. The findings provide open source researchers and practitioners with fresh insights for better understanding and modeling project teams to facilitate their success.

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

Introduction

In the current digitally enabled collaborative environment, 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 companies 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). In Distributed Version Control Systems (DVCS) based development platforms like GitHub, we can delineate the core and peripheral contributors based on their rights of access to the main project code (Rullani and Haefliger 2013). In this classification, the core contributors are those who can directly make changes to the main project code, while the peripheral contributors are those who are not given write access to the project but still contribute to the project. 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). 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, very few studies have tried to examine FLOSS 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), this essay addresses the following two questions: a) *What role does contributor access restrictions 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 the theoretical understanding of 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).

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 write 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. 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. 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 lead 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 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, which 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, which emerge when a task requires a resource for completion, and (c)

resource – resource dependencies, which 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 manifest 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, 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 positive influence on the awareness and adoption of the FLOSS product but can also help during the development; 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)[13], 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 2018). 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.

[13] 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.

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. 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 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[14]. 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.

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

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. 2008b) 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 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 (Appendix 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. 2008b). 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 2016. 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[15] at any point during the years 2014 and 2015, we identified that contributor as a core contributor. Table 3-1 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 3-1, we note that the mean contributions of the core contributor is much higher than that of the peripheral contributor (Crowston, Wei, et al. 2006).

Table 3-1: Mean and Standard Deviations of Contributions Made by Contributors

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

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

[15] 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.

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.

Contributor/owner characteristics: 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.

Project characteristics: 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 2018). 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. 2008b). 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 3-2 provides the means, standard deviations, and correlation coefficients for the variables used in the analyses. From Table 3-2, 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 3-2: Mean, Standard Deviation, and Pairwise Correlation Coefficients of the Variables

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

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. 2008b).

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 β_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 3-3 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 3-3), 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 3-3), 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. 2008b). 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).

Table 3-3: Results of Survival Analysis

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 3-4: Results of HLM 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

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 3-4 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 3-4), we find that the organization ownership tends to decrease the total number of code contributions made by the core contributor (β_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 workflows. 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 study of the temporal aspects of the project can enhance our understanding about the impact that the life-cycle stage of a FLOSS project has on its team composition. 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.

Future work

As an extension to this work, we are examining the influence of access restrictions on the average creativity of the contributor's contributions (commits[16]). Following Yong et al. (2014), we consider creativity to be consisting of two components – novelty and usefulness. The novelty component of creativity depends on the exchange, combination, and transformation of

[16] In GitHub, a commit, or "revision", is an individual change to a file (or set of files). It is similar to saving a file, except that it creates a unique ID (a.k.a. the "SHA" or "hash") that allows us to keep record of what changes were made when and by who. Commits usually contain a commit message which is a brief description of what changes were made (https://help.github.com/articles/github-glossary/#commit).

unique ideas into new, never before seen ideas (Yong et al. 2014). The novelty of a commit is considered high if the commit implements important or challenging changes by using new ideas, references, and techniques. The usefulness component of creativity reflects the extent to which the implemented changes take into consideration practical issues and constraints (Yong et al. 2014). The usefulness of a commit is considered high if the changes implemented are valuable to the end user and at the same time has an appealing simplicity (i.e. the commit is not overly complex). Adopting a consensual assessment approach (Amabile 1982), we measure the creativity of individual commits as their *novelty* and *usefulness,* assessed through expert ratings. Two expert programmers, who have considerable experience in the open source community (one of whom is a stackoverflow C/C++ community moderator), will be engaged to label a random sample of commits on the basis of their novelty and usefulness. Using the labeled sample of commits and adopting a supervised machine learning approach, we intend to build a classification model to determine the novelty and usefulness of a commit. Using the developed measure, we plan to study the influence of access restrictions on the average creative output of the contributors. By extending our research in this direction, we contribute to the understanding of the mechanisms that lead to an increase in creative outcomes in open source software projects.

4 Essay 3: Community Ideologies

Abstract

Though, volunteer driven Free (Libre) and Open Source Software (FLOSS) development was founded on the ideological beliefs of 'openness' and 'absence of any commercial appropriation', in recent years FLOSS movement has witnessed two ideological shifts. First, the emergence of 'permissive FLOSS licenses' that allow commercial appropriation of the collaboratively developed code, and second, 'organizational ownership' of FLOSS projects. Because ideological beliefs shape the motivational needs of the volunteer contributors, it is expected that ideological shifts could influence the mechanisms through which dominant work structures in FLOSS projects are related to their outcomes. Motivated by the need to understand the impact of these ideological shifts, we theorize the mechanisms through which the two ideological shifts alter the influence of FLOSS work structures on project outcomes of popularity and survival. Using an instrument variable approach, the results from our analysis of over 4000 FLOSS projects hosted on GitHub confirm the significance of both the ideological shifts with some interesting contextual differences across the two project outcomes. Specifically, we find that the ideological shifts pertaining to license type and organizational ownership has a significant moderating influence on the relationship between the work structures and project popularity.

Keywords: Open source software, FLOSS, ideologies, structures of work, license, motivation, copyleft, superposition

Introduction

Digitally enabled transformations have led to profound changes in the way individuals and organizations undertake software development work. Amongst them, the transformation of software development from being traditionally in-house to one that is open source is particularly noteworthy (Ågerfalk et al. 2015). Central to this transformation are the ideological shifts that permitted the FLOSS movement, founded on the ideological beliefs of openness and absence of commercial appropriation (Ljungberg 2000), become more accepting of

commercial participation. In specific, two ideological shifts have shaped the way we see FLOSS today – (a) the emergence of permissive licenses that allowed commercial appropriation of the collaboratively developed code (Daniel et al. 2018), and (b) the movement towards organizational ownership that allowed the emergence of a corporate-communal landscape in FLOSS projects (Fitzgerald 2006; Germonprez et al. 2016). Because ideological beliefs shape the motivational needs of the volunteer contributors (Daniel et al. 2018), and motivational needs of contributors shape the emergent work structures in FLOSS projects (Howison and Crowston 2014), it is expected that ideological shifts could influence the mechanisms through which dominant work structures in FLOSS projects are related to their outcomes. Motivated by the need to understand these ideological shifts considering their influence on the unique work orchestration mechanisms in FLOSS projects, our study attempts to answer the following research question:

RQ: How have the ideological shifts invoked by (a) the emergence of permissive licenses, and (b) the shift towards organizational ownership, transformed the influence of FLOSS work structures on project outcomes?

The creation of the FLOSS movement, founded on the ideological beliefs of openness and no commercial appropriation, changed the nature of motivation of the developers from being largely extrinsic (as in employees of an organization) to more intrinsic (as in volunteer contributors) (Ke and Zhang 2010). The intrinsically motivated volunteer contributors differed in their psychological needs leading to the emergence of new work orchestration mechanisms that soon came to dominate FLOSS development work (Howison and Crowston 2014). In this context, Howison and Crowston (2014), observed and conceptualized superposition of tasks as the dominant work orchestration mechanism in FLOSS projects, wherein motivationally independent tasks are incrementally layered to create the software in a sequential manner. This work orchestration mechanism is different from that observed in the case of traditional software development, where the focus is towards co-work and concurrent task development through a modular task design (Baldwin and Clark 2006). Grounding our study in the theory of collaboration through open superposition (Howison and Crowston 2014) and the ideological

influences on contributor motivations (Daniel et al. 2018; Stewart and Gosain 2006), we examine the changes brought about to the relationship between FLOSS work structures and project outcomes in the context of the ideological shifts that are reshaping the FLOSS phenomenon.

Through this research, we aim to make the following contributions to theory and practice. First, we advance the understanding on FLOSS ideologies by teasing out the influence of FLOSS ideological shifts on software development project outcomes. We do this by theorizing the changes to the motivational mechanisms invoked by the ideological shifts and their subsequent influence on the relationship between the work structures of FLOSS projects and their project outcomes. This theorization provides a novel understanding of the impact of ideological shifts, which links the literature on work structures in FLOSS projects (Howison and Crowston 2014) and ideologies in FLOSS communities (Daniel et al. 2018; Stewart and Gosain 2006). Second, we contribute to the theoretical understanding on the different ways in which the choice of license type influences project outcomes in individual and organizational-owned FLOSS projects. Prior research has examined many important aspects related to choice of license types (e.g. Singh and Phelps 2013, Stewart et al. 2006) and organizational participation (e.g. Fitzgerald, 2006; Capra *et al.*, 2011; Germonprez *et al.*, 2016) in FLOSS projects and has laid the groundwork for a deeper theoretical inquiry. Building on these prior research streams, we show that the ideological influence of the choice of license types on FLOSS project outcomes is sensitive to the organizational participation in the project. Lastly, the results from this study help advance theories related to the value creation mechanisms that facilitate FLOSS project outcomes and offer practical insights to project owners. 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 of the ideological influence of the choice of license type on the outcomes of the project can lead to better management practices for planning potentially successful FLOSS projects.

Theoretical Background

To understand the influence that ideological shifts have on the FLOSS project work structures, two important concepts need to be developed: (a) ideological shifts in the FLOSS movement, and (b) the emergent structures of work in FLOSS projects. In this section, we set the context for each of the concepts and provide a brief background on the related literature.

Ideological Shifts in the FLOSS Movement

Traditional (proprietary) software developed by individuals or organizations are protected by copyright laws, which make the software a private good (Lerner and Tirole 2005). The Free Software Foundation (FSF, www.fsf.org) created by Richard Stallman in 1985 tried to create a legal basis which would allow collaboratively developed code to not only remain free for use, but to also ensure the free availability of the source code (Stallman 1999). This movement gave rise to free software movement which provided the early definition of FLOSS built on four essential freedoms that the software needs to provide to its users (Ljungberg 2000):

- The freedom to run the program as the user wishes, for any purpose (freedom 0)
- The freedom to study how the program works, and change it so that it does the computing as the user wishes (freedom 1)
- The freedom to redistribute copies so that the user can help others (freedom 2)
- The freedom to distribute copies of the modified versions to others. By doing this the user can give the whole community a chance to benefit from the implemented changes (freedom 3)

Freedom 0 and 1 together supported the ideological belief of *openness* that ensured FLOSS would remain open for the user to run, modify and use any way the user chooses. Freedom 2 and 3, ensured the user's right to redistribute the current or modified versions of the software to others, centred on the ideological belief of *no commercial appropriation*. To support the development of high quality software based on these ideological beliefs, the movement relied on volunteer contributors who were willing to work and support the ideological beliefs with little or no commercial rewards (Crowston et al. 2012). Hence, we see that the embedded

ideology that saw the success of FLOSS movement had three elements – (a) the ideological belief of *openness*, (b) the ideological belief of *no commercial appropriation*, and (c) the need for *volunteer contributions* to facilitate the development of useful software based on the aforementioned two ideological beliefs.

To protect the ideological beliefs of 'openness' and 'no commercial appropriation', FSF created the *restrictive* licensing policy. Software that adopted restrictive licenses gave any user the right to copy, modify, and redistribute the software. In addition to the above, any enhancement to code and even any proprietary software that made use of the source code would be bound by the same license and would be forced to remain open (Lerner and Tirole 2003; Singh and Phelps 2013). This restrictive feature of the license made sure that any offshoots of the software would also remain free or open, thereby preventing the commercialization and closure of the collaboratively developed code (Singh and Phelps 2013).

First Ideological Shift – Emergence of Permissive Licenses. To relax the non-commercial ideology of the restrictive licenses, permissive licenses emerged (e.g. BSD, ASF and MIT) which provided most of the open source features of the restrictive licenses but also gave the developer the freedom to change the license policy of the built software. The Open Source Initiative (OSI; www.opensource.org) founded in 1998, spearheaded the creation of permissive licenses to provide developers the freedom to use their code whichever way they chose. In principle, the movements founded by FSF and OSI are grounded in different ideologies. While FSF prefers restrictive licenses that limit commercial appropriation of the code, OSI holds a more inclusive view that allows for commercial appropriation (Daniel et al. 2018).

Second Ideological Shift – Organizational Ownership of FLOSS Projects. 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 for 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 primarily 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. This transformation gave rise to the second ideological shift seen in the FLOSS movement, which saw organizations initiating and taking ownership of FLOSS projects. The ideological shift in this case centered around the entry of paid employees into the mix of volunteer contributors, and the inclusion of organizational management and control practices to the FLOSS projects (Capra et al. 2011). These changes not only undermined the ideological beliefs of openness and no commercial appropriation, but also changed the team structure to a hybrid mix of organizational employees and volunteer contributors in the project (Germonprez et al. 2016).

Because ideologies can manifest as important mechanisms that shape the motivational needs of the contributors (Stewart and Gosain 2006), it is expected that the two described ideological shifts could have altered the underlying motivational mechanisms of the FLOSS project contributors. In this research, we study these motivational changes and understand how the ideological shifts create new alignment requirements for the underlying FLOSS work structures and their relationships with projects outcomes.

Emergent Work Structures in FLOSS Projects – Superposition

Recent works of Howison and Crowston (2014) and Lindberg et al. (2016), have called attention to the characteristics of the software artefact, such as the emergent structures of work that are surprisingly effective at establishing the delicate balance between —managing developers' contributions, and sustaining the contributors' motivational needs. In this context, Howison and Crowston (2014), observed and conceptualized *superposition of tasks* as the dominant work orchestration mechanism in FLOSS projects. Superposition is the process through which software development occurs in a sequential manner, with changes to the software added incrementally, on top of one another. Each change represents a task that is independently built by a contributor and has its own functional payoff through the improvements that it brings to the application (Howison and Crowston 2014). Superposed task

work is therefore characterized by (a) individual task work, and (b) incremental layering of motivationally independent tasks. This unique work-breakdown structure has been found to accomplish complex work through a process of 'productive deferral' in which tasks that are envisioned to be too large to be implemented through individual, motivationally independent layers are deferred until code written by someone else (and sometimes for entirely different reasons) makes the envisioned task easy enough to be undertaken through relatively simple, quick individual work (Howison and Crowston 2014).

The dominance of this work orchestration mechanism in FLOSS environments is attributed to its ability to satisfy the psychological needs of the intrinsically motivated volunteer contributors. To build their argument, Howison and Crowston (2014) invoked theories of motivation and coordination. Specifically, they expanded the work of Ke and Zhang (2010), who applied self-determination theory (SDT; Ryan and Deci 2000) to show that superposition creates an effective work-breakdown structure that satisfies the three psychological needs of intrinsically motivated contributors —autonomy, competence, and relatedness. In specific, by minimizing the interdependencies among contributors, this work breakdown structure has been found to enhance the sense of *autonomy* in the intrinsically motivated volunteer contributors without compromising their need for competence and relatedness. Because ideological shifts alter the motivational mechanisms of FLOSS projects, we can say that each ideological shift creates new requirements for aligning projects work structures to the motivational needs of the contributors. Following this line of enquiry, in the following sections, we theorize how the aforementioned ideological shifts influence the relationships between the FLOSS project work structures and their outcomes of popularity and survival.

Theory and Hypothesis

First Ideological Shift – Understanding the Relationship between License Type, Degree of Superposition and Project Outcomes (Popularity and Survival)

The ideological difference between the restrictive and permissive types of licenses stems from their contrasting beliefs regarding the commercial appropriation of collaboratively developed

code. While the proponents of FSF and restrictive licenses stand against commercial appropriation of the code, the OSI holds a more inclusive view and supports permissive licenses that allow for commercial appropriation (Daniel et al. 2018). The ideological belief against the commercial appropriation of the FLOSS software was echoed by Richard Stallman, the founder of FSF and a strong proponent of restrictive licenses - "when information is generally useful, redistributing it makes humanity wealthier no matter who is distributing and no matter who is receiving" (Stewart and Gosain 2006; pp 294). The limitations to commercialize code written under restrictive licenses, may deter extrinsically motivated individuals and organizations who expect tangible rewards for their participation in FLOSS projects. On the contrary, the rigid interpretation of the concept of "information freedom" embedded in the restrictive licenses may attract intrinsically motivated contributors who are driven to make a positive impact on the society and realize a sense of accomplishment through their contribution. Thus, projects that adopt restrictive licenses are likely to attract a greater proportion of intrinsically motivated contributors than projects adopting permissive licenses. This ideological influence on the motivation of the contributors has also been observed, e.g. Sen et al. (2008), found that intrinsically motivated individuals who are driven by the challenge of the work prefer moderately restrictive over permissive licenses, while others who are extrinsically motivated tend to prefer permissive licenses.

To understand the mechanisms through which the motivational needs of the FLOSS contributors influence their project outcomes, we invoke self-determination theory (SDT) and the theory of collaboration through open superposition. SDT and its sub theories postulate the existence of three innate psychological needs: *autonomy*, *relatedness*, and *competence*, which lead to enhanced self-motivation when satisfied and result in a positive affective state (Ke and Zhang 2010; Ryan and Deci 2000). SDT also postulates that intrinsically motivated contributors have a higher psychological need for autonomy and self-regulation as compared to extrinsically motivated individuals. Given this influence, projects that adopt restrictive

licenses need to align the project attributes to meet the higher need for autonomy in the intrinsically motivated contributors.

The emergent structures of work in FLOSS projects have been found to provide an important mechanism through which the contributors' need for autonomy can be satisfied (Howison and Crowston 2014). In specific, the theory of collaboration through open superposition states that in the case of FLOSS projects, an emergent superposition of tasks provides the most effective work-breakdown structure for enhancing the intrinsic motivation to contribute, and at the same time allows for the creation of complex software (Howison and Crowston 2014). Superposed structures of work are found to be more effective than concurrent work when intrinsically motivated contributors are involved because concurrent work tends to create dependencies between contributors, thereby reducing their autonomy (Howison and Crowston 2014). By providing the motivational mechanisms that satisfy the innate psychological needs of autonomy, competence and relatedness, superposition generates a positive affective state for the intrinsically motivated contributors (Ryan and Deci 2000). The increased positive affective state enhances the task effort that an individual will expend (Weiss and Cropanzo 1996), encouraging greater contributions to the FLOSS project and this leads to an increase in the quality, functionality, and usability of the software for the end user (Ke and Zhang 2010). Thus, we posit that in the case of projects that adopt restrictive licenses, an increase in the degree of superposition is expected to have a stronger positive influence on the popularity of a FLOSS project than in the case of permissive licenses (Figure 2-1). Hence, we hypothesize;

Hypothesis 1a: In individual owned projects, the type of license moderates the relationship between the degree of superposition and the popularity of FLOSS projects, such that, for projects with restrictive licenses an increase in the degree of superposition tends to have a higher positive influence on the popularity of the project than for projects with permissive licenses.

It is not only the motivational mechanisms, but also the perceptions of legitimacy that can change when we move from restrictive to permissive licenses. The survival of FLOSS projects is determined by the perceptions of legitimacy that the project is able to engender and sustain

(Chengalur-smith et al. 2010). Suchman (1995), provides an inclusive and broad-based definition of legitimacy as —"a generalized perception or assumption that the actions of an entity are desirable, proper, or appropriate within some socially constructed system of norms, values, beliefs, and definitions" (Suchman 1995; pp 574). Within the socially constructed ideological belief of 'openness' and 'no commercial appropriation' in projects adopting restrictive licenses, the provision of autonomy by adopting a superposed organization of tasks can help enhance the perceptions of legitimacy in the minds of intrinsically motivated contributors who support the principles embedded in FSF. Because permissive licenses adopt a more inclusive view of ideological beliefs on FLOSS than restrictive licenses, contributors participating in such projects may be more open towards other structures of work that offer lesser autonomy. Thus, it is expected that the perceptions of legitimacy are more closely tied to autonomy and superposed work structures in the minds of supporters of FSF and restrictive licenses as compared to supporters of OSI and permissive licenses. Because changes to the perceptions of legitimacy are reflected as changes to the FLOSS project's ability to survive (Chengalur-smith et al. 2010), we hypothesize that;

Hypothesis 1b:In individual owned projects, the type of license moderates the relationship between the degree of superposition and the survival of FLOSS projects, such that, for projects with restrictive licenses, an increase in degree of superposition tends to have a higher positive influence on the survival of the project than for projects with permissive licenses.

Second Ideological Shift –Influence of Organizational Ownership on the Relationship between License Type, Degree of Superposition, and Project Outcomes (Popularity and Survival)

As organizations get involved in FLOSS projects, they are faced with the challenge of balancing a commercial profit value-for-money proposition while still adhering to the ideological beliefs of the open source community (Fitzgerald 2006). The added influence of organizational ownership on the project contributors' motivations and the owner's potential need for control may influence the relationship between superposition and the popularity of the project. First, organizations bring in strategic planning initiatives and introduce project management

practices for the efficient coordination and management of software development activities (Capra et al. 2011; Fitzgerald 2006). For example, in a study of 83 eclipse projects, it was found that —FLOSS projects initiated by market driven organizations employed both leadership and resource deployment control practices but such practices were not employed in individual owned FLOSS projects (Schaarschmidt et al. 2015). Second, organizational ownership changes the nature of the FLOSS project teams from being predominantly voluntary contributors, to a mix of organizational employees and volunteers external to the organization (Capra et al. 2011; Germonprez et al. 2016).

The introduction of well-defined goals and mechanisms for managing and controlling FLOSS development activities along with the changes to the team structure can affect the moderating influence of the choice of license type on the relationship between superposition and project popularity. In projects that adopt restrictive licenses, the formal project management practices and control structures introduced by the organizational owner tends to counter the embedded ideological beliefs of 'openness' in FLOSS (von Krogh et al. 2012). It is therefore expected that there is a mitigation in the ideological belief that are embedded in restrictive licenses when organizations own such projects. Due to this influence, organization owned projects attract fewer intrinsically motivated contributors who share the ideological beliefs of restrictive licenses as compared to individual owned projects. The introduction of employees of organizations into the projects further shifts the nature of motivation from intrinsic to extrinsic. Unlike volunteer contributors, who often contribute during their free time, the employees of an organization receive tangible rewards (in terms of salary) for their contributions. Hence organizational ownership of FLOSS projects introduces a larger proportion of extrinsically motivated contributors who receive tangible rewards and have less need for autonomy than intrinsically motivated volunteer contributors (Deci et al. 1999). Therefore, in the case of projects with restrictive licenses, we contend that mitigation of the ideological belief of 'openness' when organizations own FLOSS projects and the introduction

of organizational employee tends to change the nature of motivation of the contributors from being predominantly intrinsic towards being extrinsic.

On the contrary, the liberal ideology of permissive licenses tends to be more aligned to organizational participation. Hence, the change from individual to organizational ownership does not change the nature of motivation for the contributors to as large an extent as projects with restrictive licenses (see Table 4-1). Overall, when organizations own FLOSS projects there is a greater shift towards extrinsic motivation for projects with restrictive licenses in comparison to projects with permissive licenses.

Table 4-1: Moderating Influence of Organizational Ownership on the Relationship between License Type, Degree of Superposition and Project Popularity

		First Ideological Shift (Hypothesis 1a)	
		Restrictive License	**Permissive License**
Second Ideological Shift (Hypothesis 2a)	**Individual Ownership**	High positive marginal influence of degree of superposition on project popularity **(+++)** • Ideological influence of restrictive license attracts a higher proportion of intrinsically motivated contributors • Intrinsically motivated contributors have a high need for autonomy • Superposition helps satisfy the high needs of autonomy enhancing the task effort and consequently increases the popularity of the project	Moderate positive marginal influence of degree of superposition on project popularity **(++)** • Inclusive nature of permissive license, which permits commercial appropriation, attracts a higher proportion of extrinsically motivated contributors who are also interested in developing code for profit • Extrinsically motivated contributors have a comparatively less need for autonomy • Due to a larger proportion of extrinsically motivated contributors, there is a decrease in the marginal influence of superposition on popularity of projects as compared to individual owned projects with restrictive licenses

	Organizational Ownership	Moderate to low positive marginal influence of degree of superposition on project popularity **(++/+)** • There is an ideological misfit when organization own FLOSS projects and brings in management and control practices. Hence the ideological influence of restrictive licenses which tends to attract intrinsically motivated contributors is mitigated • The introduction of employees of the organization into the projects further shifts the nature of motivation from intrinsic to extrinsic • Considering the net increase in extrinsic motivation of contributors, there is a considerably lower marginal influence of superposition on the popularity of the project as compared to individual owned projects with restrictive licenses	Moderate to low positive marginal influence of degree of superposition on project popularity **(++/+)** • Inclusive nature of permissive license, which permits commercial appropriation, accepts organizational ownership to a greater extent than restrictive licenses. The ideological shift in comparison to individual owned projects with permissive license is small • The introduction of employees of the organization into the projects shifts the nature of motivation from intrinsic to extrinsic • The marginal influence of superposition on popularity remains similar or slightly lower to individual owned projects with permissive licenses

In sum, we expect that the ideological shift introduced by organizational ownership serves to mitigate the difference in motivational mechanisms that is introduced by the ideological shift due to the change in license types. Based on this understanding, we posit that in the case of organization owned FLOSS projects, the positive moderation effect of the choice of license type on the relationship between degree of superposition and the popularity of the project will be lower as compared to individual owned projects (Table 4-1, Figure 4-1). Hence, we hypothesize:

Hypothesis 2a: For organization owned projects, the moderating influence of license type on the relationship between the degree of superposition and the popularity of FLOSS projects is less in comparison to individual owned projects

When it comes to the survival of projects, the changes to the perceptions of legitimacy when organizations own FLOSS projects can have a major influence on the sustenance of the project (Chengalur-smith et al. 2010). In the case of restrictive licenses, organizational ownership may

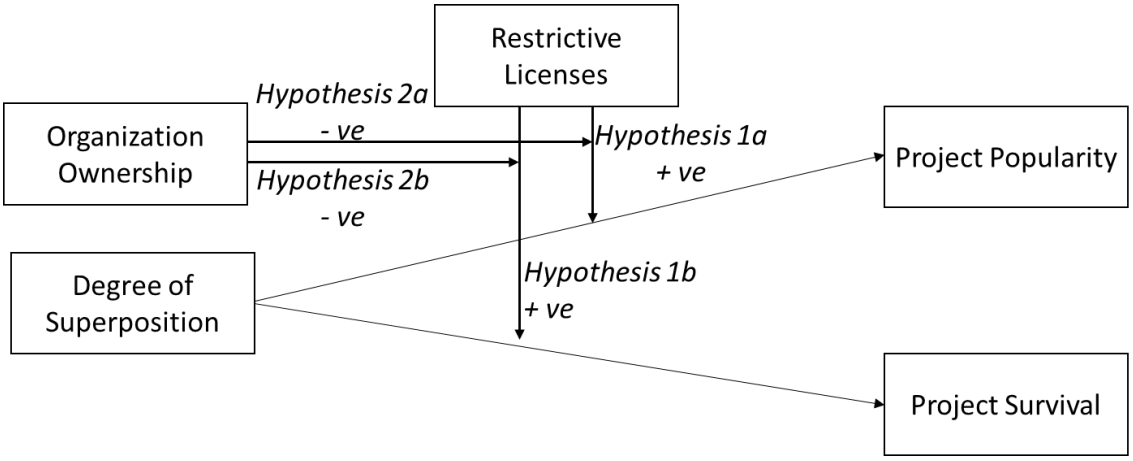
conflict with the ideological beliefs of the volunteer contributors and supporters of restrictive licenses (Stewart, Ammeter, et al. 2006). This is mainly because the introduction of project management and control practices conflicts with the ideas of freedom and autonomy, which are closely associated with the ideological beliefs of FSF. Moreover, most of the project management and control practices may be handled by organizational employees, which may engender a sense of loss of control in the minds of the volunteer contributors. For example, in the Android open source project, employees of Google take on the role of project leaders who lead all technical aspects of the project, including the project roadmap, development, release cycles, versioning, and quality assurance[17].

Because of this sense of loss of control in volunteer contributors when organizations own FLOSS projects, the provision of autonomy through superposition of tasks may not instil the same perceptions of legitimacy in the minds of contributors and supporters of FSF. Because the supporters of permissive licenses have a more inclusive view and are aligned to commercial appropriation, the management and control practices brought in by the organizational ownership may not diminish the influence of the perceptions of legitimacy in these projects (see Figure 4-1). Hence, we hypothesize;

Hypothesis 2b: For organization owned projects, the moderating influence of license choice on the relationship between the degree of superposition and the survival of FLOSS projects is less in comparison to individual owned projects.

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

Figure 4-1: Theoretical Model



Methodology

To understand the influence of the two ideological shifts and test our hypotheses, 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 make 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 for a sample of FLOSS projects from the GHarchive 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 developed our estimation models to test the hypothesized relationships.

Data Collection

We employed Google's bigquery tool to query the archived project log data available in the GitHub Archive database (Grigorik 2012). Since this database is large (about 432 GB, with 134 million rows for the year 2014 alone; Google 2014), we needed the bandwidth provided by a tool like Google's bigquery to run queries and export the results. We restricted our analysis to projects that were started during the first five months of 2014 to reduce the number and size

of queries. Further, the project log data collection for each project was restricted to all development work that was undertaken in the years 2014 and 2015.

Measurement

GitHub offers a good environment for measuring the degree of superposition and studying its relationship to the project's popularity and 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 for all events enabled us to measure the work structure in terms of the degree of superposition. Second, the availability of detailed contributor- and project-specific data allowed us to measure the dependent variables and operationalize the necessary controls. The different variables that we used in this research and their measures are detailed in the following subsections.

Dependent variables. We use two dependent variables in this research (a) *popularity of the project,* and (b) *project survival*. As mentioned in the section "FLOSS Project Outcomes – Popularity and Survival," we consider the popularity of the project to be an important measure of FLOSS project outcome. In GitHub, users can "star" projects in order to keep track of those that they find interesting and also to show their appreciation for these projects (GitHub 2017a). The number of stars a project has received indicates approximately the number of people who are interested in and show support for that project. Count of stars is therefore a commonly used measure for identifying popular projects in the GitHub environment: GitHub itself uses stars to identify trending projects and in its project rankings (GitHub 2017a), Jarczyk et al. (2014) used the log transformation of the number of stars as a measure of the quality and popularity of GitHub projects, and Tsay et al. (2014) used the number of stars as a measure of popularity and project establishment. In recognition of the usefulness of count of stars as a measure of popularity, we adopted it as the first dependent variable to test our relationships.

In addition to project popularity, we included the survival of the project as an alternate measure of FLOSS project outcome. In this model, we considered the event to have happened (project failure), if the project became inactive within the first two years of its inception.

Because we considered projects that were started in early 2014, a project was deemed inactive if no code changes were made to 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 estimated the likelihood of project inactivity at time *t* given that it has lasted till time *t* by calculating the hazard function (Hosmer et al. 2008a).

Independent variables. We employed three independent variables in this study– *restrictive license flag, degree of superposition,* and *project ownership*. The *restrictive license* flag is the main independent variable used in our analysis. This flag takes a value of "1" if the project adopts the restrictive "copyleft" clause as defined by FSF. The license descriptions curated by GitHub was used to determine if the license was restrictive or permissive (https://choosealicense.com/). Some of the projects in our sample did not have a license assigned or adopted a lesser known license for which accurate descriptions were not provided. To avoid miss classification of such projects, we removed such projects from our sample.

We adopted the approach outlined in essay 1 of this dissertation to operationalize the *degree of superposition (DoS*). Specifically, we measured it as the ratio of the total number of versions of the project to the total number of individual task contributions made to the project. This operationalizing of degree of superposition is based on the concept of superposition introduced by Howison and Crowston (2014). Based on this operationalization, the degree of superposition for a project takes a value between 0 and 1. If *degree of superposition* = 1, all the project's tasks were implemented individually and added sequentially, with each individual task contribution representing a new version of the project. The degree of superposition decreases as a project adopts a concurrent development approach and approaches 0 as greater number of individual task contributions get piled onto individual versions of the project.

To study the moderating influence of project ownership, we used a flag, *project ownership*, 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 enable companies to create nested teams with hierarchical access to the code, allowing them to replicate their organizational 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.

Instrument variable for the choice of license. The empirical challenge in most studies on FLOSS licenses is that the choice of license type for a project may be endogenous with outcome parameters of interest - such as project popularity. For example, project owners considering the commercial potential of a project may choose permissive licenses and, at the same time, target functionalities that are useful to the organization rather than to the end-users without paying any particular attention to the popularity of the project. Further, they may adopt a more concurrent style of development to speed up production and meet deadlines. This would generate a downward bias to the marginal influence of degree of superposition on the popularity of the project. In such cases, the observed association between the outcome and explanatory variable is likely to be misleading as the estimates partly reflects omitted factors that are related to both the variables (Angrist & Krueger 2001). Instrument variables are often used to address endogeneity arising from omitted variable biases and hence offers a good identification strategy for our research.

The choice of FLOSS license type is often made by the project owner once they have envisioned the project and laid down its basic objectives. We leverage the fact that cultural and societal background of the project owner may play a role (in addition to the commercialization intentions) in his/her decision to choose a restrictive vs. a permissive license. Specifically, we identify that project owners belonging to more collectivist cultures tend to be more open for choosing restrictive licenses. At the same time, the undertones of beliefs of socialism in

restrictive licenses vs. liberal beliefs in permissive licenses suggest project owners coming from socialist cultures may prefer restrictive licenses. Leveraging the cultural and societal distinctions of the project owners, we have created our instrument variable by collecting the location information of the project owner. Because the location entries are self-reported in GitHub, we manually verify the country of origin of the project owner. We use two country level indicators as instruments for the choice of license:

a) Geert Hofstede dimension – Individualism vs. collectivism (Hofstede 2011). Project owners coming from collectivist cultures should tend to use restrictive licenses as compared to project owners coming from individualistic cultures.
b) Country's social protection contribution as a percentage of total expenditure in the year 2014-2015. The social protection contribution of the country is a good indicator of the country's focus towards social welfare. We leverage this distinction to create the second instrument for the license type. Because restrictive license is closer to social policies than liberal policies, we expect project owners coming from countries that have greater expenditure on social protection to typically lean towards restrictive licenses. We choose 2014-2015 for the instrument since the projects in our sample started in early 2014.

   The aforementioned country level indicators fulfil the conditions for being a good instrument.

Control variables. We controlled for two kinds of variables in our analysis: individual characteristics and project characteristics.

Individual characteristics. We identified three measures to control the influence of individual level characteristics on the hypothesized relationships. 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 different values of the number of contributors in the project. In addition to the number of contributors, the *average number of contributions per contributor* is also expected to influence the project outcomes (Subramaniam et al. 2009). Considering this, we controlled for the average number of commits per contributor, where a commit is any change or set of changes that are locally saved to file (GitHub 2017b).

Project characteristics. We identified six measures to control for different project characteristics. First, 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). Second, 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. Third, we controlled for the *average task size*, measured as the average number of commits made within the project's tasks. 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 large tasks will have a higher need for co-work than projects with smaller tasks. Fourth, we controlled for *number of subscriptions* in the project. In GitHub, subscriptions indicate the number of individuals who receive notifications regarding the project development activities. Like the number of contributors in a project, increase in subscriptions can attract attention to the code development activities, allowing better bug identification and opportunities for code improvement (Stewart, Ammeter, et al. 2006). This can in turn lead to better project outcomes

in terms of its popularity and survival. We also included two fixed effects: the fixed effect for the *main programming language* and the fixed effect for the *month of creation*. Despite mixed results in the extant literature, the type of programming language used has been found to be an important antecedent to FLOSS project outcomes (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 Clark 2006). We controlled for these effects by including a dummy variable for each programming language used in our sample. Lastly, time is expected to influence the nature of routines associated with FLOSS projects (Lindberg 2015). Because our analysis includes projects that commenced during the first five months of 2014, it was necessary to include the fixed effect for the month of creation to control for the influence of time on the hypothesized relationships.

Analysis

We adopted two estimation models to test our hypotheses. For estimating project popularity, we use the negative binomial model. On the other hand, we conducted survival analysis using the accelerated failure-time (AFT) model.

Negative binomial model. Table 4-2 provides the means, standard deviations, and correlation coefficients for the variables used in the analyses. From Table 4-2, we observe that the mean of the count of stars is much lower than its variance, indicating the presence of over-dispersion in the measure. To account for this over-dispersion, we adopted a negative binomial approach with number of stars as the dependent variable to test hypotheses 1a and 2a (Cameron and Trivedi 2013, Chapter 4). In this model, we used the original scale for the number of stars as the dependent variable. Since the model predicts the log of the expected number of stars as a function of the predictor variables, the interpretations of the coefficient are like the log-linear OLS model. That is, for a one-unit change in the predictor variable, the number of stars is expected to change by the respective regression coefficient, given that the other predictor variables in the model are held constant.

2SLS with endogenous regressors. To study the effect of license choice on project popularity and to overcome potential endogeneity issues, we develop an instrument variable for the explanatory variable leveraging the country of origin of the project owner. The instrument relies on calculating the country's cultural index and social protection contribution. Using the developed instrument, we adopt a two-stage least squares model to test hypotheses 1a and 2a.

Survival model. To understand what determines the survival of the project and test hypotheses 1b and 2b, we conducted survival analysis using accelerated failure-time (AFT) model with a lognormal parametric representation of the survival function(Hosmer et al. 2008a). The choice of lognormal parametric representation was based on the understanding that the FLOSS projects experience high chance of mortality at the start but tend to have better chance of survival as the time passes (Samoladas et al. 2010). In this model the natural logarithm of the project duration is assumed to follow a normal distribution. Therefore, we expect the hazard rates, which gives the likelihood of project inactivity at a point in time to be initially increasing and then decreasing with time.

Probit model with endogenous regressors. To study the effect of license choice on project survival, we adopt a probit model with endogenous regressors using the project inactivity flag as the dependent variable. In this model, the dependent variable takes a value of 1 if the project became inactive within the first two years of its activity, else it is 0.

The following section details the results of the regression models and the checks we employed to ensure the validity of our results.

Table 4-2: Mean, Standard Deviation, and Pairwise Correlation Coefficients of the Variables

	Variable	Mean	Std. Dev.	1	2	3	4	5	6	7	8	9	10	11
1	Stars	361.09	1,216.60											
2	Subscriptions	26.15	54.06	0.68**										
3	Average task complexity	2.51	14.95	-0.01	0.00									
4	No of programing languages	3.17	3.00	0.04**	0.11**	0.02								
5	Size of project in megabytes	2.16	22.27	0.00	0.02	0.01	0.27**							
6	Owner experience	67.24	134.32	0.05**	0.10**	-0.01	-0.05**	-0.02						
7	Average commits per contributor	51.39	103.19	-0.02	0.00	0.03	0.30**	0.11**	-0.06**					
8	Organization owner flag	0.53	0.50	-0.04**	0.12**	0.01	0.14**	0.05**	0.04**	0.06**				
9	Restrictive license flag	0.11	0.32	-0.03*	-0.01	0.01	0.14**	0.07**	-0.09**	0.13**	0.01			
10	Degree of superposition	0.70	0.17	-0.07**	-0.12**	-0.05**	-0.05**	-0.01	-0.12**	0.02	-0.20**	0.03**		
11	Total contributors	9.34	21.33	0.16**	0.18**	0.06**	0.20**	0.15**	0.04**	0.05**	0.09**	0.04**	-0.17**	
12	Project inactivity/failure event	0.26	0.44	-0.13**	-0.10**	-0.01	-0.07**	-0.01	0.01	-0.08**	-0.05**	-0.02	-0.05**	-0.06**

* p<0.05; ** p<0.01

Results and Discussion

Table 4-3 provides the results of the regression models for individual owned projects. Models 1a and 1c provide the results of the negative binomial and 2SLS models we employed to estimate project popularity. Models 1b and 1d provide the results of the survival analysis and probit models with endogenous regressors we employed to estimate project popularity. Table 4-4 provides the results of the regression models for organization owned projects. Models 2a and 2c provide the results of the negative binomial and the probit model with endogenous regressors models we employed to estimate project popularity for the organization owned project subgroup. While models 2b and 2d provide the results of the survival analysis and the probit model with endogenous regressors for the organization owned project subgroup. Because the residuals of the variables —size of project, and average task size show significant positive skew in their distribution, we used their log transformations in our estimation models (Carte and Russel 2003; Singh et. al. 2013). After log transformation, their residuals were found to closely match a normal distribution.

Table 4-3: Results of the Moderated Regression Analysis for Individual Owned Projects

	Individual Owner Subgroup			
	Model 1a	Model 1b	Instrument Model 1c - IV	Instrument Model 1d - IV
Dependent Variable	Project Popularity (log(Stars))	Likelihood of Inactivity (Coefficient)	Project Popularity (log(Stars))	Project Inactivity Flag
Owner experience	0.074** 0.02	-0.089* 0.042	0.001* 0.0003	0.0001 0.0004
Avg. contributions per contributor	0.001+ 0.0003	-0.004** 0.001	0.001 0.0005	-0.002* 0.0007
Log(size of project)	-0.052** 0.019	-0.03 0.04	-0.003 0.037	-0.012 0.043
Number of languages	-.035* 0.016	0.056 0.036	-0.011 0.024	0.043 0.027
Log(avg. task size)	0.062 0.045	0.129 0.092	0.07 0.067	0.101 0.072
Subscribers	0.026** 0.001	-0.032** 0.004	0.012** 0.0007	-0.01** 0.002
Restrictive license flag	-1.641** 0.349	2.191** 0.622	-6.896* 3.34	-4.74 3.168
Degree of superposition	-0.342+ 0.191	-0.968* 0.38	-0.431 0.73	-2.28** 0.619
Restrictive license flag X Degree of superposition ($ß_1$)	1.539** 0.448	-2.385** 0.824	8.588* 4.19	6.65+ 3.7
Fixed effect of number of contributors	YES	YES	YES	YES
Fixed effect of main programming language	YES	YES	YES	YES
Fixed effect of month of creation	YES	YES	YES	YES
N	1,929	1,926	1,836	1,620

** p<0.001 *p<0.05 +p<0.1

Hypotheses 1a and 1b predict that the first ideological shift represented by the choice of license type moderates the relationship between degree of superposition and project popularity and survival. To test these hypotheses, we introduced the interactions of restrictive license flag with the degree of superposition term in the regression models. From model 1a and1c (Table 4-3), we find that the interaction term of restrictive license flag with the degree of superposition significantly influences the project popularity for both - the negative binomial ($\beta_1 = 1.539$, $p < .01$) and the 2SLS model ($\beta_1 = 8.58, p < .05$). This provides support for hypothesis 1a. From model 1b and1d (Table 4-3), we find that the interaction term of restrictive license flag with the degree of superposition significantly influences the project survival for ($\beta_1 = -2.385$, $p < .01$) . However, the probit model with endogenous regressors is not significant at the 95%

confidence interval ($\beta_1 = 6.65, p > .05$) and does not lend support for hypothesis 1b.

Table 4-4: Results of the Moderated Regression Analysis for Organization Owned Projects

	Organization Owner Subgroup			
	Model 1a	Model 1b	Instrument Model 1c - IV	Instrument Model 1d - IV
Dependent Variable	Project Popularity (log(Stars))	Likelihood of Inactivity (Coefficient)	Project Popularity (log(Stars))	Project Inactivity Flag
Owner experience	-.141** 0.02	-0.024 0.042	-0.001** 0.0002	-.001** 0.0002
Avg. contributions per contributor	0.001* 0.0003	-0.005** 0.001	0.001+ 0.0002	-0.001** 0.0004
Log(size of project)	-0.02 0.019	-0.016 0.037	0.017 0.023	-0.008 0.028
Number of languages	-0.026+ 0.013	0.016 0.028	-0.007 0.016	0.01 0.022
Log(avg. task size)	-0.101** 0.04	0.03 0.076	-0.053 0.049	-0.018 0.015
Subscribers	0.018** 0.0007	-0.012** 0.002	0.011** 0.0006	-0.004** 0.001
Restrictive license flag	-0.186 0.26	1.544** 0.481	-0.599 1.074	2.013 1.288
Degree of superposition	.422* 0.186	-1.041 0.351	0.561 0.39	-0.471 0.462
Restrictive license flag X Degree of superposition ($ß_1$)	-.678* 0.378	-2.01** 0.717	0.429 1.617	-3.187+ 1.93
Fixed effect of number of contributors	YES	YES	YES	YES
Fixed effect of main programming language	YES	YES	YES	YES
Fixed effect of month of creation	YES	YES	YES	YES
N	2,092	2,090	1,999	1,700

** p<0.001 *p<0.05 +p<0.1

Hypotheses 2a and 2b predict that the nature of influence of license type on the relationship between degree of superposition and project popularity and survival is lower when the project is owned by organizations as compared to individuals. To test these hypotheses, we test the means across the subgroup of individual- and organization-owned projects using the seemingly unrelated regression approach (Zellner 1962). From model 2a and 2c (Table 4-4), tests for mean difference across the subgroup of individual-and organization-owned projects support hypothesis 2a. For project survival, the test for mean difference is supported in the AFT model without instruments. However, the probit model with endogenous regressors does

not lend support for hypothesis 2b. We discuss some of the implications of these findings in the subsequent sections.

Limitations

Although our research purports to make valuable contributions to both research and practice, they need to be interpreted against the backdrop of certain limitations which we acknowledge. First, we studied the influence of ideological shifts on the relationship between work structures and project outcomes because prior research has shown that embedded work structures in FLOSS are dependent on the motivational mechanisms driving the contributors. While this provides an interesting angle to study the influence of the ideological shifts, the complete picture of the ideological influence on the project outcomes may involve other project attributes (e.g. team structure, project type). To minimize the confounding effect of these project attributes we have tried to identify and include them as control variables. Second, the data collection and measurement for our study relied on data collected from GitHub project logs. While GitHub offers a good dataset for the purposes of this study, certain perils are associated with its use (Kalliamvakou et al. 2014). To minimize the effect of these perils, we introduced multiple controls, filters, and manual checks. The details of the perils in using this dataset and the checks we adopted to overcome them are listed in Appendix 3. However, this still leaves the question – can the results be replicated across other platforms? Third, our operationalization of the constructs – degree of superposition, project popularity, project survival was done based on existing definitions of the measures (e.g. Samoladas *et al.*, 2010; Howison & Crowston, 2014; Borges & Valente, 2018). This allowed us to generalize our findings over a large set of projects and test the relationships on the project outcomes – popularity and survival. However, we acknowledge that the subjective nature of these constructs makes it difficult to develop a measure that captures all aspects of the construct. Hence, we need to exercise some caution when interpreting the results of the analyses.

Despite these limitations, our study contributes to the literature by taking a modest step in enriching our understanding of the ideological shifts in FLOSS development and the

mechanisms through which these shifts can influence the relationship between work structures and project outcomes.

Implications

The value creation mechanisms that have resulted in the success of the FLOSS phenomenon have inspired new sourcing strategies and engagement models (Ågerfalk et al. 2015). Moreover, organizations are now approaching FLOSS with the fabled idea of replicating the immense success of projects like Linux and Apache, which continue to dominate their markets. Microsoft, once the "evil enemy" (Ljungberg 2000, pp. 210), has been steadily shifting its image, perhaps with the hope of being looked at as a benevolent dictator in the FLOSS community (Microsoft News Center 2018). On the other side, the free software movement, which was founded to prevent the "commercial hijack" of collaboratively developed code (Ciffolilli 2004), has been evolving into a movement that now regularly witnesses commercial participation and organizational ownership (Fitzgerald 2006; Germonprez et al. 2016). The fusion of the two vastly different ideologies (open vs. commercial software) has resulted in the emergence of a new corporate-communal landscape with multiple interwoven, overlapping, conflicting, and divergent practices that enable and constrain FLOSS engagements (Germonprez et al. 2016). Driven by the need to understand the ideological shifts that led to the evolution of the corporate-communal landscape and unearth their influence on the mechanisms that were embedded in the founding ideology of openness and absence of commercial appropriation, this research sought to answer the following research question - *How have the ideological shifts invoked by (a) the emergence of permissive licenses, and (b) the shift towards organizational ownership, transformed the influence of work structures on FLOSS project outcomes?*

The theoretical and empirical development that was done to answer this question has considerable implications for both theory and practice.

Theoretical Implications: By unearthing the influence of ideological shifts that are associated with the choice of license type and organization ownership of FLOSS projects, we

contribute to three literature streams in information systems and organization studies – literature on FLOSS ideologies, license types, and organizational participation. First, our research adds to the existing understanding on FLOSS ideologies (e.g. Daniel et al. 2018; Ljungberg 2000; Stewart and Gosain 2006) as it takes a significant step in delineating the dynamic nature of ideological beliefs in the FLOSS context. Because of this dynamism, we find that the mechanisms (e.g. motivational mechanisms) that are shaped by the ideological beliefs are associated with boundary conditions of time and space (Suddaby 2010). This finding leads us to an important question: how are ideological shifts triggered in the context of the FLOSS phenomenon? As a preliminary answer to this question, our research finds that ideological shifts can occur due to individuals and institutions questioning the tenets of the embedded ideological beliefs (e.g. creation of OSI and permissive licenses) or from an attempt to find a common ground between entities with different ideologies (e.g. creation of the corporate-communal landscape; Germonprez et al. 2016). Future research can build on these findings to provide a more complete typology of ideological triggers in the context of the FLOSS landscape, which will further our understanding about the important pathways through which ideological shifts can shape project outcomes.

Second, the choice of FLOSS license type is one of the important decisions made by project owners (Fershtman and Gandal 2007), which is instrumental in influencing project outcomes (Stewart, Ammeter, et al. 2006). This salience has attracted researchers across many disciplines who have tried to understand the mechanisms involved in choosing a particular license type (e.g. Lerner and Tirole 2005; Sen et al. 2008; Singh and Phelps 2013) and the impact of license type choice on project outcomes (e.g. Colazo et al. 2005; Scotchmer 2010; Sen et al. 2011; Stewart, Ammeter, et al. 2006). But the research on the influence of the choice of license type on project outcomes, has been less conclusive —with conflicting findings across the main classes of licenses (Crowston et al. 2012). Against this backdrop, our research, grounded in ideological beliefs of FLOSS provides a novel way to understand the influence of license type on project outcomes. From our analyses, we find that the license type choice

exhibits a moderating influence on the relationship between superposed work structures and the project popularity, which is conditional on the type of project ownership. Thus, the influence of choice of license type on the project outcomes depends on how well the project attributes, such as its work structures, are aligned to the ideological beliefs associated with the license type. Further, our study reveals that the ideological beliefs embedded in the principles of restrictive licenses can be mitigated when they are mixed with the contradicting ideological beliefs brought in by organizational ownership of the project.

Third, our research advances the literature surrounding organizational participation in FLOSS projects (Capra et al. 2011; Fitzgerald 2006; Germonprez et al. 2016; Spaeth et al. 2015) by examining how embedded ideological beliefs may change when organizations own FLOSS projects. Further, we find that this change in ideological beliefs brought in by organizations can influence the value creation mechanisms in FLOSS environments. By ascertaining the moderating influence of ideologies on the relationship between dominant work orchestration mechanisms and the project outcomes, we contribute to the emerging literature on corporate-community landscape (Germonprez et al. 2016) and show that task-work design can be an effective response to complex ideologies. Project owners can thus design artefacts that facilitate better project outcomes under different ideological situations.

Practical Implications: Practically, organizations and individual owners of FLOSS projects have constantly been trying to replicate the value creation mechanisms of successful FLOSS projects. Because the motivational mechanisms of the contributors can be embedded in their ideological beliefs (Daniel et al. 2018; Stewart and Gosain 2006), it is important for project owners to understand the influence of ideological changes that may occur when entities of different ideologies come together in FLOSS projects. Of particular importance in this relationship are the dominant work structures that emerge as a means to balance the motivational needs of the contributors and simultaneously overcome the challenges related to FLOSS task work orchestration (Howison and Crowston 2014; Lindberg et al. 2016). While superposition is the dominant work orchestration mechanism that is well suited to the

embedded ideology of openness and limited time pressures (as in non-organizational settings) (Howison and Crowston 2014), our findings suggest that changes to the embedded ideology necessitate corresponding changes to the work structures to facilitate project outcomes. This understanding is especially important for organizations who are constantly trying to balance efficient development practices but are faced with the challenge of meeting the motivational needs of autonomy among the intrinsically driven contributors.

Lastly, the complex nature of FLOSS project outcome measures, suggest that different outcome measures engender different value creation mechanisms associated with the ideological beliefs. For example, popularity and task effort are more closely associated with the motivational mechanisms (Ke and Zhang 2010), while project sustenance may depend more on the perceptions of legitimacy (Chengalur-smith et al. 2010). Because of this difference, when project owners consider the influence of ideologies, the type of project outcome desired by the owner needs to be an important part of their design decisions. Cognizance of these influences can help project owners make sense of the complex influence of ideological beliefs and facilitate better project outcomes.

5 Conclusion

Despite the FLOSS development model's increasing prominence in practice and research, the antecedents to its success have not been completely understood. In particular, the important role played by the sociotechnical aspect of its work structures (Howison and Crowston 2014), the informal governance mechanisms associated with its team composition (Rullani and Haefliger 2013), and motivational mechanisms of community ideologies (Daniel et al. 2018) have only recently received attention from researchers. Considering the steady shift towards FLOSS in recent years and its expected acceleration in the future, clarifying the value creation mechanisms of this phenomenon can inform project owners about how they can benefit from adopting this unique model of development. As a step forward in this direction, this dissertation, comprised of three essays, sought to address the following broad research questions:

1. How do work structures influence the success of FLOSS projects?
2. How do informal governance mechanisms that emerge in FLOSS teams influence the survival of FLOSS projects?
3. How do ideological shifts transform the value creation mechanisms associated with FLOSS work structures?

In the first essay of this dissertation, we study the mechanisms through which the dominant work orchestration mechanisms observed in FLOSS projects, referred to as superposition, influence the success of the project. Further, we show how these mechanisms get altered when organizations own the FLOSS project and participate through different models of organizational engagement (Capra et al. 2011). Our theorization, which is based on self-determination theory (Ryan and Deci 2000), establishes the boundary conditions associated with the influence of work structures on the success of projects. Using a novel construct for the work structures of the project - degree of superposition, we empirically investigate the theorized boundary conditions on a large sample of FLOSS projects hosted on GitHub. The analysis lends support for a nonlinear relationship between the degree of superposition and

the success of the FLOSS project. Moreover, we find that the type of ownership moderates this nonlinear relationship such that (1) organizational ownership mitigates the influence of the degree of superposition on the success of the project, and (2) under organizational ownership, the optimal degree of superposition (the point at which the success of the project is at a maximum) is lower than for individual-owned projects. Theoretically, this essay progresses the research agenda initiated by Howison and Crowston, (2014) and Lindberg et al. (2016), by establishing the role of work structures as a key driver for contributors' motivations and also as an antecedent to project success. From the standpoint of motivational theories (e.g. SDT), we highlight the difficulties in scaling the mechanisms operating at the individual level (e.g. mechanisms associated with human psychology) to the team or project level. The findings from this essay suggest that the difficulties in scaling these individual level mechanisms up to the project level manifest as boundary conditions describing the application of theories from one level of analysis to another. When organizations get involved in FLOSS projects, we show that the resulting increase in net extrinsic motivation of the contributors and a higher time cost of money alters the motivational influence of superposed work structures. Moreover, we find that different models of organizational engagement in the FLOSS project exhibit different moderating influence on the relationship between superposed work structures and the project's success. These findings provide FLOSS practitioners with valuable insights for better modeling work structures to facilitate the success of the project.

The second essay examines FLOSS project teams and the emergence of informal network governance mechanisms in these teams. Building on the theories of coordination (Malone and Crowston 1994) and network governance (Jones et al. 1997), we study the influence of source-code access restrictions in mitigating coordination challenges and protecting social exchanges. Further, we investigate how formal project management and control mechanisms introduced by the organizations can complement the informal governance mechanisms in FLOSS communities. Using a Cox proportional hazard model, we demonstrate that the relationship between the proportion of contributors who are given write access to the source code in the

team and the survival of the project, is moderated by the nature of project ownership — individual versus organization owned projects. Interestingly, the observed moderation is a crossover interaction effect that changes from negative for individual owned projects to positive for organization owned projects. From the standpoint of coordination theory (Malone and Crowston 1994) and the theory of network governance (Jones et al. 1997; Wasko et al. 2004), we expand the current understanding of the mechanisms through which access restrictions in teams contribute to the sustenance of the project. By considering the interplay between network governance mechanisms and project management practices seen in organizations, our theorizing suggests that control (through project management practices) and openness (by providing access to the source code) can complement each other and co-exist under certain conditions. That is, our findings suggest that project management practices introduced by organizations can help protect the openness of FLOSS projects by lowering their need for informal governance mechanisms like access restrictions and collective sanctions (Wasko et al. 2004). Cognizance of these influences can help organizations establish the delicate balance between openness and control (Engeström 2007, Jarvenpaa and Lang 2011), which is an important antecedent to the success of FLOSS projects (Teigland et al. 2014).

Building on extant research on community ideologies (e.g. Daniel et al. 2018; Ljungberg 2000; Stewart and Gosain 2006), the third essay takes a significant step in delineating the dynamic nature of ideological beliefs in FLOSS communities and its influence on the value creation mechanisms associated with the dominant work structures. Using an instrument variable approach, our analysis of FLOSS projects hosted on GitHub confirm that the ideological shifts pertaining to license type and organizational ownership have a moderating influence on the relationship between work structures and project popularity. This essay contributes to the emerging literature on the corporate-communal landscape in FLOSS (Germonprez et al. 2016) by examining the alignment requirements of community ideologies and other project characteristics like the work structures and licenses. While superposition is the dominant work orchestration mechanism that is well suited to the embedded ideology of openness and limited

time pressures (as in non-organizational settings) (Howison and Crowston 2014), our findings suggest that changes to the embedded ideology necessitate corresponding changes to the work structures to facilitate better project outcomes. This understanding is especially important for organizations who are constantly trying to balance efficient development practices but are faced with the challenge of meeting the motivational needs of autonomy among the intrinsically driven contributors.

In summary, the three essays, viewing the FLOSS value through multiple theories and perspectives, contribute to rapidly expanding discipline of open sourcing. The contributions made by each of the three essays to theory and practice have already been discussed in detail in the previous sections of this dissertation. In addition to the already highlighted contributions by each of the three essays, all the essays together clearly bring out some important considerations for future researchers across various domains examining the field of open source.

Future work

In the future, I plan to address the second part of my long-term research objective by contextualizing the value creation mechanisms examined in my dissertation to non-IS disciplines. Doing so may help non-IT organizations understand how they can successfully adopt the FLOSS model of development. My hope is that this may in the future facilitate the emergence of OSS 3.0[18], where we see the transformation of the FLOSS phenomenon into a form that is viable across all industries.

In relation to the work structures, future research can look towards understanding how non-IT related work can be orchestrated in a manner that motivates volunteer contributors and at the same time allows for the creation of complex products. The boundary and contextual conditions that limit/enhance the influence of work structures, studied in the first essay of my

[18] The term OSS 2.0 was coined by Fitzgerald (2006) to describe the transformation of FLOSS phenomenon into a more mainstream and commercially viable form.

dissertation, provide a good starting point to understand the work orchestration mechanisms that can prove successful in non-IT contexts. The key research questions that emerge in this enquiry are: How can we divide complex work into motivationally independent tasks so that it can be accomplished by volunteer contributors? How can we manage interdependencies across complex tasks to ease the coordination challenges? and How can we satisfy the volunteer contributors psychological needs of autonomy, competence, and relatedness, thereby encouraging greater task effort?

Regarding governance structures, future work can attempt to build on the informal network governance structures identified in the second essay of my dissertation to theorize the optimal governance mechanisms that can facilitate the success under different contextual conditions. The dynamism of FLOSS team composition and the do-ocratic style of governance observed in the second essay suggests the emergence of unique governance structures in successful FLOSS projects, which are different from those observed in traditional organizations and communities. However, extant literature has not yet theorized the emergence of a dominant governance mechanism in FLOSS like environments under different contextual conditions. This leads to the question: how can non-IS organizations govern their projects when they adopt a FLOSS approach to development where there is a high level of openness and limited time pressures.

The findings of the third essay indicate that in order to mimic the success of FLOSS, industries need to create the seamless corporate-communal landscape that we see in FLOSS projects. Creating this corporate-communal landscape requires a careful alignment of the community ideologies with other project attributes like work and governance structures to best motivate the contributors. Since licenses can formalize the community ideologies in projects, project owners should pay careful attention to the choice of license for their projects. This leads us to the question: how can we replicate the contractual mechanisms that we see in successful FLOSS projects across non-IT organizations? Or more generally, how can non-IT organizations

facilitate the corporate-communal landscape in their projects? The findings from the third essay of my dissertation provide a good starting point to attempt answering these questions.

While many of these questions are daunting and without direct or simple answers, I am confident that, irrespective of the results, these questions are worthy of a deeper enquiry. The answers to these questions will not only expand our theoretical understanding of the FLOSS phenomenon, but also facilitate a profound positive change in the nature of work across all industries.

6 Bibliography

Ågerfalk, P. J., Fitzgerald, B., and Stol, K.-J. 2015. "Not So Shore Anymore : The New Imperatives When Sourcing in The Age of Open," *Ecis* (2015), pp. 1–17.

Aiken, L. S., and West, S. G. 1991. *Multiple Regression: Testing and Interpreting Interactions*, Newbury Park, CA: SAGE Publications.

Alvesson, M., and Sandberg, J. 2011. "Generating Research Questions through Problematization," *Academy of Management Review* (36:2), pp. 247–271. (https://doi.org/10.5465/amr.2009.0188).

Amabile, T. M. 1982. *Social Psychology of Creativity : A Consensual Assessment Technique*, (43:5), pp. 997–1013.

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. (https://doi.org/10.1287/orsc.1100.0618).

Baldwin, C. Y., and Clark, K. B. 2006. "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?," *Management Science* (52:7), pp. 1116–1127. (https://doi.org/10.1287/mnsc.1060.0546).

Borges, H., and Tulio Valente, M. 2018. "What's in a GitHub Star? Understanding Starring Practices in a Social Coding Platform," *Journal of Systems and Software*. (https://doi.org/10.1016/j.jss.2018.09.016).

Cameron, A. C., and Trivedi, P. K. 2013. "Regression Analysis of Count Data," *Cambridge University Press* (Vol. 53), Cambridge, England: Cambridge University Press. (https://doi.org/10.2307/1271358).

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. (https://doi.org/10.1016/j.jss.2010.09.004).

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.

Chua, C. E. H., and Adrian, Y. K. Y. 2010. “Artifacts , Actors , and Interactions in the Cross-Project Coordination Practices of Open-Source Communities,” *Journal of the Association for Information* (11:12), pp. 838–867.

Ciffolilli, A. 2004. “The Economics of Open Source Hijacking and the Declining Quality of Digital Information Resources: A Case for Copyleft,” *First Monday* (9). (https://doi.org/10.5210/fm.v9i9.1173).

Colazo, J. A., Fang, Y., and Neufeld, D. 2005. “Development Success in Open Source Software Projects: Exploring the Impact of Copylefted Licenses,” in *AMCIS 2005 Proceedings*, pp. 929–936. (http://aisel.aisnet.org/amcis2005/432).

Cook, R. D. 1977. “Detection of Influential Observation in Linear Regression,” *Technometrics* (19:1), pp. 15–18. (https://doi.org/10.2307/1268249).

Coverity Inc. 2013. “Coverity Scan: 2013 Open Source Report,” pp. 1–23. (http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf, accessed January 4, 2017).

Crowston, K. 1997. “A Coordination Theory Approach Organizational Process Design,” *Organization Science* (8:2), pp. 157–175. (http://190.24.150.71:2107/stable/2635308?seq=1&Search=yes&searchText=process&searchText=documentation&list=hide&searchUri=/action/doBasicResults?hp=25&la=&wc=on&fc=off&acc=on&acc=on&bk=off&pm=off&jo=off&ar=off%252).

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. (https://doi.org/10.1002/spip.259).

Crowston, K., and Scozzi, B. 2004. “Coordination Practices Within FLOSS Development Teams: The Bug Fixing Process,” *Computer Supported Activity Coordination* (4:1), pp. 21–30.

Crowston, K., Wei, K., Howison, J., and Wiggins, A. 2012. “Free/Libre Open-Source Software Development,” *ACM Computing Surveys* (44:2), pp. 1–35. (https://doi.org/10.1145/2089125.2089127).

Crowston, K., Wei, K., and Li, Q. 2005. “Coordination of Free / Libre and Open Source Software Development,” in *Institute for Software Research*.

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. (https://doi.org/10.1109/HICSS.2006.101).

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. (https://doi.org/10.1016/j.respol.2005.02.003).

Daniel, S. L., Maruping, L. M., Cataldo, M., and Herbsleb, J. 2018. “The Impact of Ideology Misfit on Open Source Software Communities and Companies,” *MIS Quarterly* (42:4), pp. 1–28. (https://doi.org/10.25300/14242).

Deci, E. L., Koestner, R., and Ryan, R. M. 1999. “A Meta-Analytic Review of Experiments Examining the Effects of Extrinsic Rewards on Intrinsic Motivation.,” *Psychological Bulletin* (125:6), pp. 627–668. (https://doi.org/10.1037/0033-2909.125.6.627).

Engeström, Y. 2007. “From Communities of Practice to Mycorrhizae,” *Communities of Practice: Critical Perspectives.*, pp. 1–20. (https://doi.org/10.4324/9780203101339).

Fershtman, C., and Gandal, N. 2004. “The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination,” *Working Paper*, pp. 1–24. (https://doi.org/10.2139/ssrn.515282).

Fershtman, C., and Gandal, N. 2007. “Open Source Software: Motivation and Restrictive Licensing,” *International Economics and Economic Policy* (4:2), pp. 209–225. (https://doi.org/10.1007/s10368-007-0086-4).

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), pp. 587–598. (http://www.scopus.com/inward/record.url?eid=2-s2.0-33845974935&partnerID=40&md5=8b03a53f72862e732ce35d7ede8b988e).

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. (https://doi.org/10.1016/j.jss.2013.11.1077).

Germonprez, M., Kendall, J. E., Kendall, K. E., Mathiassen, L., and Young, B. 2016. “Engagement with Open Source Communities A Theory of Responsive Design : A Field Study of Corporate Engagement with Open Source Communities,” *Information Systems Research* (28:1), pp. 64–83. (https://doi.org/10.1287/isre.2016.0662).

GitHub. 2017a. “About Stars.” (https://help.github.com/articles/about-stars/, accessed January 1, 2017).

GitHub. 2017b. “Glossary - Commit.” (https://help.github.com/articles/github-

glossary/#commit, accessed January 1, 2017).

GitHub. 2017c. “Users.” (https://developer.github.com/v3/users/, accessed April 5, 2017).

Google. 2014. “Table Details: 2014.” (https://bigquery.cloud.google.com/table/githubarchive:year.2014?pli=1&tab=details, accessed January 1, 2017).

Gordon, G. G. 1991. “Industry Determinants of Organizational Culture,” *Academy of Management Review*. (https://doi.org/10.5465/amr.1991.4278959).

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. (https://doi.org/10.1145/2568225.2568260).

Greenberger, D. B., and Strasser, S. 1986. “Development and Application of a Model of Personal Control in Organizations,” *Academy of Management Review* (11:1), pp. 164–177.

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. (https://doi.org/10.1287/mnsc.1060.0550).

Grigorik, I. 2012. “The GitHub Archive.” (https://www.githubarchive.org/, accessed January 1, 2017).

Guenter, H., van Emmerik, I. J. H., and Schreurs, B. 2014. “The Negative Effects of Delays in Information Exchange: Looking at Workplace Relationships from an Affective Events Perspective,” *Human Resource Management Review* (24:4), pp. 238–298. (https://doi.org/10.1016/j.hrmr.2014.02.001).

Haans, J. F. R., Pieters, C., and He, Z.-L. 2015. “Thinking about U: Theorizing and Testing U- and Inverted U-shaped Relationships in Strategy Research,” *Strategic Management Journal* (37), pp. 1177–1195. (https://doi.org/10.1002/smj).

Hair, J. F., Black, W. C., Babin, B. J., and Anderson, R. E. 2010. “Multivariate Data Analysis 7th Edition,” *Prentice Hall*, Upper Saddle River, NJ: Prentice Hall. (https://doi.org/10.1016/j.ijpharm.2011.02.019).

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.

Hofstede, G. 2011. “Dimensionalizing Cultures: The Hofstede Model in Context,” *Online*

Readings in Psychology and Culture (2:1), pp. 1–26. (https://doi.org/10.9707/2307-0919.1014).

Hosmer, D. W., Lemeshow, S., and May, S. 2008a. "Applied Survival Analysis. Regression Modeling of Time-to-Event Data," *Technometrics*. (https://doi.org/10.2307/1270580).

Hosmer, D. W., Lemeshow, S., and May, S. 2008b. "Applied Survival Analysis. Regression Modeling of Time-to-Event Data," *John Wiley & Sons* (Vol. 41), New York, NY: John Wiley & Sons. (https://doi.org/10.2307/1270580).

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

Jarczyk, O., Gruszka, B., Jaroszewicz, S., and Bukowski, L. 2014. "GitHub Projects. Quality Analysis of Open-Source Software," in *SocInfo 2014: The 6th International Conference on Social Informatics*, pp. 80–94.

Jarvenpaa, S. L., and Lang, K. R. 2011. "Boundary Management in Online Communities: Case Studies of the Nine Inch Nails and Ccmixter Music Remix Sites," *Long Range Planning* (44:5–6), pp. 440–457. (https://doi.org/10.1016/j.lrp.2011.09.002).

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. (https://doi.org/10.2307/3069453).

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.

Ke, W., and Zhang, P. 2010. "The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development," *Journal of the Association for Information Systems* (11:12), pp. 784–808.

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.

Lakhani, K., and Wolf, R. G. 2005. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," in *Perspectives on Free and Open Source Software*, J. Feller, B. Fitzgerald, S. Hissam, and K. R. Lakhani (eds.), Cambridge, MA, pp. 3–22. (https://doi.org/10.2139/ssrn.443040).

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. (https://doi.org/10.1016/j.omega.2007.05.005).

Lerner, J., and Tirole, J. 2003. "Some Simple Economics of Open Source," *The Journal of Industrial Economics* (50:2), pp. 197–234. (https://doi.org/10.1111/1467-6451.00174).

Lerner, J., and Tirole, J. 2005. "The Scope of Open Source Licensing," *Journal of Law, Economics, and Organization* (21:1), pp. 20–56. (https://doi.org/10.1093/jleo/ewi002).

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. (https://doi.org/10.1016/j.infsof.2014.02.004).

Lind, J. T., and Mehlum, H. 2010. "With or without u? The Appropriate Test for a U-Shaped Relationship," *Oxford Bulletin of Economics and Statistics* (72:1), pp. 109–118. (https://doi.org/10.1111/j.1468-0084.2009.00569.x).

Lindberg, A. 2015. "The Origin, Evolution, and Variation of Routine Structures in Open Source Software Development: Three Mixed Computational-Qualitative Studies," Case Western Reserve University.

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. (http://dx.doi.org/10.1287/isre.2016.0673).

Ljungberg, J. 2000. "Open Source Movements as a Model for Organising," *European Journal of Information Systems* (9:4), pp. 208–216. (https://doi.org/10.1057/palgrave.ejis.3000373).

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. (https://doi.org/10.1145/174666.174668).

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. (https://doi.org/10.1109/MSR.2009.5069491).

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.

Medappa, P. K., and Srivastava, S. C. 2018. "Does Superposition Influence the Success of FLOSS Projects? An Examination of Open Source Software Development by Organizations and Individuals," *Information Systems Research (Forthcoming)*.

Michlmayr, M., and Fitzgerald, B. 2012. "Time-Based Release Management in Free and Open Source (FOSS) Projects," *International Journal of Open Source Software and Processes* (4:1), pp. 1–19.

Microsoft News Center. 2018. "Microsoft to Acquire GitHub for $7.5 Billion." (https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/, accessed August 1, 2018).

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. (https://doi.org/10.1016/j.jss.2011.11.010).

Mockus, A., Fielding, R. T., and Herbsleb, J. 2002. "Two Case Studies of Open Source Software Development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology* (11:3), pp. 309–346.

Munos, B. 2006. "Can Open-Source R&D Reinvigorate Drug Research?," *Nature Reviews. Drug Discovery* (5:9), pp. 723–729. (https://doi.org/10.1038/nrd2131).

Neath, K. 2010. "Introducing Organizations." (https://blog.github.com/2010-06-29-introducing-organizations/, accessed August 1, 2018).

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.

Pearce, J. 2014. "9.9 Million Lines of Code and Still Moving Fast - Facebook Open Source in 2014." (https://code.facebook.com/posts/292625127566143/9-9-million-lines-of-code-and-still-moving-fast-facebook-open-source-in-2014/, accessed December 11, 2017).

Phillips, P. C. B., and Park, J. Y. 1988. "On the Formulation of Wald Tests of Nonlinear Restrictions," *Econometrica* (56:5), pp. 1065–1083.

Rao, C. R. 1972. *Linear Statistical Inference and Its Applications*, (2nd edn.), New York, NY:

John Wiley & Sons.

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. 1998. “The Cathedral and the Bazaar,” *First Monday* (3:3). (https://doi.org/10.1007/s12130-999-1026-0).

Raymond, E. S. 1999. “The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,” *O’Reilly*. (https://doi.org/10.1017/CBO9781107415324.004).

Rebeca, M.-D., and García, C. E. 2009. *Returns from Social Capital in Open Source Software Networks*, pp. 277–295. (https://doi.org/10.1007/s00191-008-0125-5).

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*. (https://doi.org/10.1109/MSR.2007.35).

Roberts, J. A., Hann, I.-H., and Slaughter, S. A. 2006. “Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects,” *Management Science* (52:7), pp. 984–999. (https://doi.org/10.1287/mnsc.1060.0554).

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. (https://doi.org/10.1016/j.respol.2012.10.008).

Ryan, Richard M., and Deci, E. L. 2000. “Self-Determinaton Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being,” *American Psychologist* (55:February), pp. 68–78. (https://doi.org/10.1037/0003-066X.55.1.68).

Ryan, Richard M, and Deci, E. L. 2000. “Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being.,” *The American Psychologist* (55:1), pp. 68–78. (https://doi.org/10.1037/0003-066X.55.1.68).

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. (https://doi.org/10.2307/3069460).

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. (http://www.jstor.org/stable/40398943).

Samoladas, I., Angelis, L., and Stamelos, I. 2010. "Survival Analysis on the Duration of Open Source Projects," *Information and Software Technology*, pp. 902–922. (https://doi.org/10.1016/j.infsof.2010.05.001).

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.

Scotchmer, S. 2010. "Openness, Open Source, and the Veil of Ignorance," *American Economic Review* (100:2), pp. 165–171. (https://doi.org/10.1257/aer.100.2.165).

Selvidge, P. R., Chaparro, B. S., and Bender, G. T. 2002. "The World Wide Wait: Effects of Delays on User Performance," *International Journal of Industrial Ergonomics* (29:1), pp. 15–20. (https://doi.org/10.1016/S0169-8141(01)00045-2).

Sen, R., Subramaniam, C., and Nelson, M. L. 2008. "Determinants of the Choice of Open Source Software License," *Journal of Management Information Systems* (25:3), pp. 207–240. (https://doi.org/10.2753/MIS0742-1222250306).

Sen, R., Subramaniam, C., and Nelson, M. L. 2011. "Open Source Software Licenses: Strong-Copyleft, Non-Copyleft, or Somewhere in Between?," *Decision Support Systems* (52:1), Elsevier B.V., pp. 199–206. (https://doi.org/10.1016/j.dss.2011.07.004).

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.

Singh, P. V., and Phelps, C. 2013. "Networks , Social Influence , and the Choice Among Competing Innovations : Insights from Open Source Software Licenses Networks , Social Influence , and the Choice Among Competing Innovations : Insights from Open Source Software Licenses," *Information Systems Research* (24:August 2014), pp. 539–560.

Spaeth, S., von Krogh, G., and He, F. 2015. "Perceived Firm Attributes and Intrinsic Motivation in Sponsored Open Source Software Projects," *Information Systems Research* (26:1), pp. 224–237. (https://doi.org/10.1287/isre.2014.0539).

Stallman, R. 1999. "The GNU Operating System and the Free Software Movement," in *Open Sources Voices from the Open Source Revolution*, p. 272. (http://www.oreilly.com/catalog/opensources/book/stallman.html).

Stewart, and Gosain. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *MIS Quarterly* (30:2), p. 291. (https://doi.org/10.2307/25148732).

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. (https://doi.org/10.1287/isre.1060.0082).

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. (https://doi.org/10.1214/088342306000000141).

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. (https://doi.org/10.1016/j.dss.2008.10.005).

Suchman, M. C. 1995. "Managing Legitimacy: Strategic and Institutional Approaches," *The Academy of Management Review* (20:3), p. 571. (https://doi.org/10.2307/258788).

Suddaby, R. 2010. "Editor 's Comments : Construct Clarity in Theories," *Academy of Management Review* (35:3), pp. 346–357. (https://doi.org/10.5465/AMR.2010.51141319).

Sutton, R. I., and Staw, B. M. 1995. "What Theory Is Not," *Administrative Science Quarterly* (40:3), pp. 371–384.

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. (https://doi.org/10.1016/j.infoandorg.2014.01.001).

Tsay, J., Dabbish, L., and Herbsleb, J. 2014. "Influence of Social and Technical Factors for Evaluating Contribution in GitHub," *36th International Conference on Software Engineering*, pp. 356–366. (https://doi.org/10.1145/2568225.2568315).

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. (https://doi.org/Article).

Weiss, H. M., and Cropanzo, R. 1996. "Affective Events Theory: A Theoretical Discussion of the Structure, Causes and Consequences of Affective Experiences at Work," *Research in Organizational Behavior* (18), pp. 1–74. (https://doi.org/1-55938-938-9).

van Wendel de Joode, R. 2004. "Managing Conflicts in Open Source Communities," *Electronic*

Markets (14:2), pp. 104–113. (https://doi.org/10.1080/10196780410001675059).

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

Yong, K., Sauer, S. J., and Mannix, E. A. 2014. *Conflict and Creativity in Interdisciplinary Teams*. (https://doi.org/10.1177/1046496414530789).

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. (https://doi.org/10.1109/ICSE.2013.6606653).

Zellner, A. 1962. "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias," *Journal of the American Statistical Association* (57:298), pp. 348–368. (https://doi.org/10.1080/01621459.1962.10480664).

Zhang, P. 2013. "The Affective Response Model: A Theoretical Framework of Affective Concepts and Their Relationships in the ICT Context," *MIS Quarterly* (37:1), pp. 247–274.

7 Appendices

Appendix 1: Adopted Approach for Identifying Tasks from GitHub Project Log Data

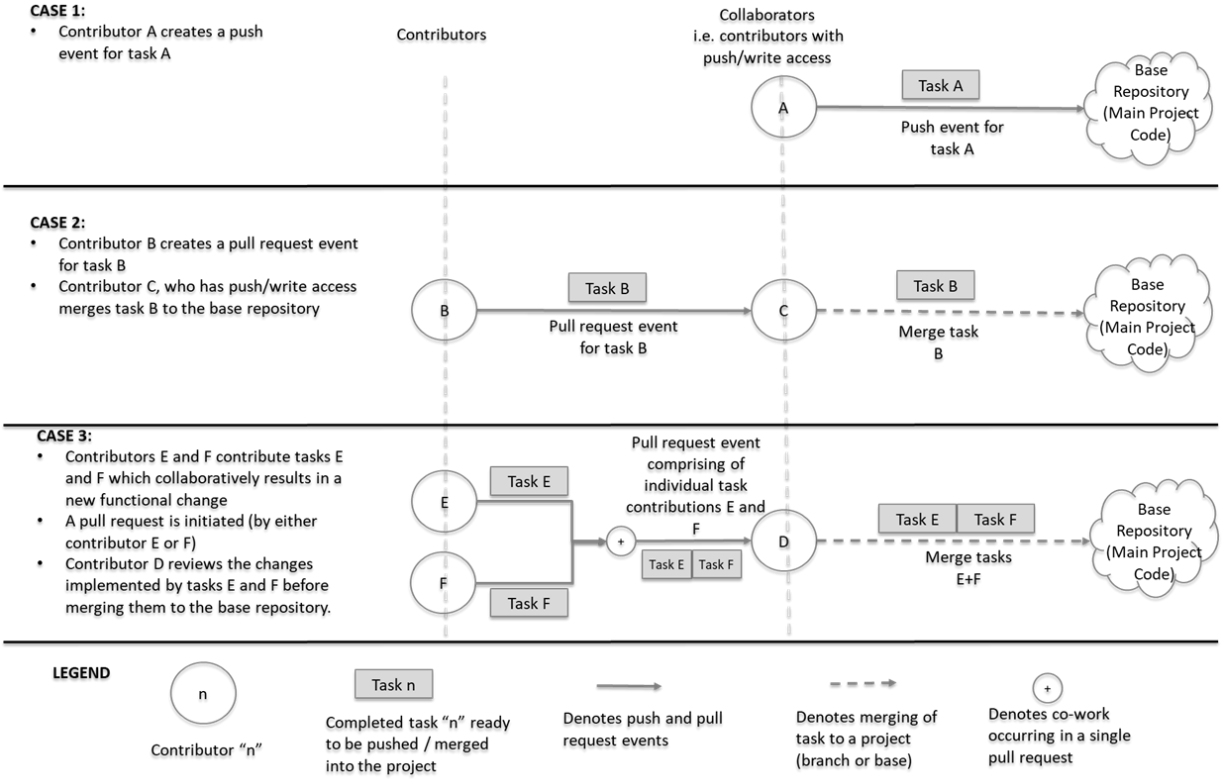
The method we adopted for identifying tasks within a project is based on the understanding that a task is a sequence of actions that leads to a change in the shared output of the project (Howison and Crowston 2014). In GitHub, a change in the shared output of a project can be accomplished through either 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 push access to a contributor by identifying that contributor as a "collaborator" for the project (GitHub 2017c). In contrast, any contributor (regardless of whether they have push access) can propose a change to the project by issuing a pull request event. In this case, a contributor with push access can choose to merge the proposed changes into the project if they are found satisfactory. Such an event can also be seen as a call for a peer review of the suggested changes. In light of this understanding, we identified tasks from both push and pull request events, observing the following criteria:

Tasks based on push events: Each push event was considered to be a new task provided that the changes associated with the push event were unique and significant. To ensure uniqueness and avoid the double counting of tasks, we considered push events only if the changes (commits) had not previously been counted as a task. For example, a push event that was created as a result of merging a pull request event was not considered a new task if the task had already been attributed to the pull request.

To ensure that tasks made significant changes, we decided that at least 24 hours must have passed for a subsequent push event by the *same* contributor to be considered a new task. This minimum time interval between push events by the same contributor was intended to prevent considering small incremental changes to the same task as different tasks. This choice of time interval is in line with the illustrative task undertaken in Howison and Crowston's (2014)

research, which was found to take approximately 20 hours spread over 3 days. As a robustness test for this choice of the time interval, we ran the analysis with 2-day, 10-hours, 5-hours, and 1-hour time windows. We found that the analysis and conclusions remained valid under the different time intervals.

Figure 7-1: Examples of Tasks Based on Push and Pull Request Events



Tasks based on pull request events: The tasks associated with a pull request event were considered only if the pull request was eventually merged into the project. In most cases, pull requests result in peer review by one or more collaborator(s) before a decision is made to merge, update, or reject the pull request. Kalliamvakou et al. (2014) identified a difficulty associated with determining whether a pull request event was merged. Appendix 3 (peril 8) describes this difficulty and the approach we adopted to overcome it.

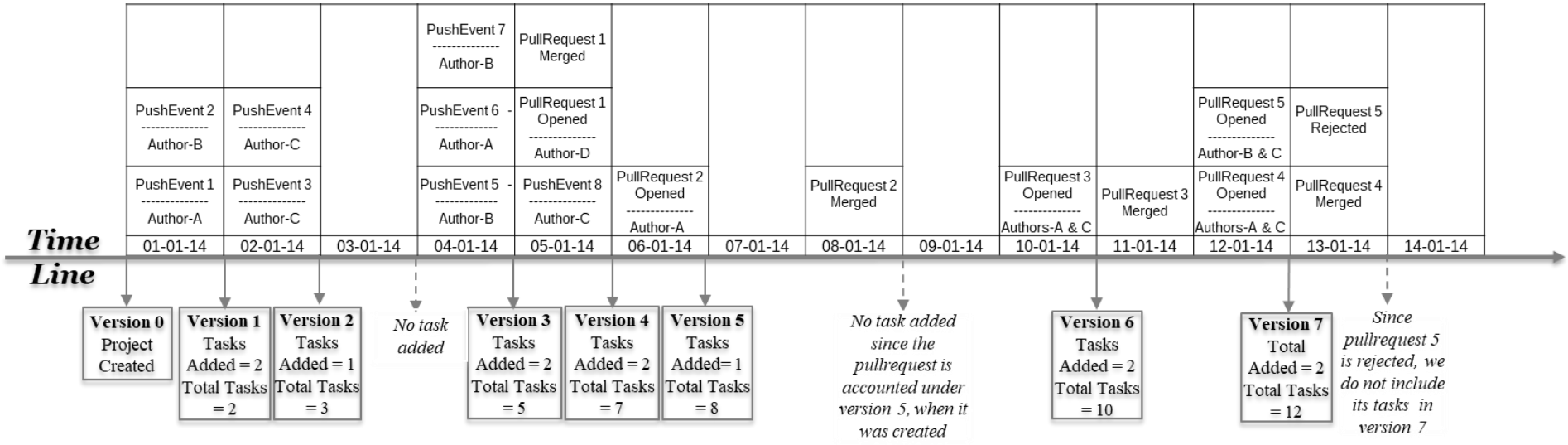
Figure 7-1 depicts three cases of task contributions to a project through push and pull request events. Case 1 is the simple case of a task associated with a push event. Case 2 depicts the task associated with a pull request event and one contributor working on the task. Case 3 depicts

two individual task contributions made by contributors E and F that are merged using a single pull request event.

Appendix 2: Adopted Approach for Identifying Versions from GitHub Project Log Data

Each version is a snapshot of the project at a point in time. On Git based version control systems like GitHub, we can define versions as the snapshot of the project that was available at the end of a specific day. Based on this definition, we can say that a new version of a project is created on any day that sees tasks being added to the project. The new version differs from the previous version of the project by the tasks that have been included to it through push and pull request events created on that day. As a robustness test for the choice of one day time interval for the operationalization of versions, we ran the analysis with 2-day and 10-hrs' time intervals. We found that the analysis and conclusions remained valid under the different time intervals. The following illustration (Figure 7-2) describes the approach that we adopted to identify versions from the project log data.

Figure 7-2: Identifying Versions Based on Push and Pull Request Events



Date 01-01-2014: On this day the project was created on GitHub and contributors A and B created two push events on the project. Each of these push event's, are considered as separate tasks for the project. The state of code at the end of 1st constitute version 1 of the project, which includes two tasks added during the 24-hour period.

Date 02-01-2014: On the 2nd, contributor C creates two push events. Since the two push events are created by the same author on the same day, we consider the two events to belong to the same task (ref. Appendix 2 for details on how we identified tasks from push events). Thus, we see that that at the end of the day, version 2 is created with the addition of one task.

Date 03-01-2014: On the 3rd, there are no tasks added to the project. Hence there is no version change attributed to this day.

Date 04-01-2014: On the 4th, three push events are created. Two of them are created by the same author "B" and the other is created by author A. In total, this results in 2 tasks which are added on top of version 2 to create version 3 of the project.

Date 05-01-2014: On the 5th, one push event is created by contributor C. In addition, a pull request event is opened by contributor D. After necessary review, the open pull request is merged to the project by a contributor who has write access to the project. Thus, we see two tasks created in the day, which get added on top of version 3 to create version 4 of the project.

Date 06-01-2014: On the 6th, one pull request event is created by contributor A to merge their code to the main project code. At a later date (8th), this pull request event is merged after necessary peer review. We consider the task that is attributed to this pull request event to be added on the 6th to create version 5 of the project. We considered this task to be added on the date of creation of pull request event (6th) rather than on the date of merger of pull request (8th), since majority of the development tends to happen before the creation of the pull request. If significant work is required after the creation of the pull request event, the collaborator who is reviewing the code often recommends closing the pull request without merging it and requests that the author to reopen the pull request after making the required changes.

Date 07-01-2014: On the 7th, there are no tasks added to the project. Hence there is no version change attributed to this day.

Date 08-01-2014: On the 8th, the pull request that was opened by contributor A on the 6th is merged to the project. Since this task is attributed to the 6th (on the date of creation of pull request event) we consider no tasks to be added on the 8th and hence no version change is attributed to the day.

Date 09-01-2014: On the 9th, there are no tasks added to the project. Hence there is no version change attributed to this day.

Date 10-01-2014: On the 10th, one pull request event is created which has contributions made by authors A and C. This pull request event is merged on the 11th. Since two contributors were

involved in this pull request event, we consider it to have created two individual tasks which are added on top of version 5 to create version 6 of the project.

Date 11-01-2014: On the 11th, the pull request that was opened the previous day by two contributors - A and C, is merged to the project. Since we account the tasks attributed to this pull request to the 10th, we consider no change to have occurred to the version on the 11th.

Date 12-01-2014: On the 12th, two pull requests are opened. One of the them comprises of commits made by contributors A and C, while the other has commits made by authors B and C. On the 13th, the pull request made by contributors A and C is merged, while the other made by contributors B and C is rejected. Thus, on this day (12th), we consider two tasks attributed to the former pull request to be added on top of version 6 to create version 7 of the project.

Date 13-01-2014: On the 13th, the pull request made by contributors A and C is merged, while the other made by contributors B and C is rejected. Since these tasks are attributed to the 12th, we consider no change to have occurred to the versions on the 13th.

Date 14-01-2014: On the 14th, there are no tasks added to the project. Hence there is no version change attributed to this day.

At the end of two weeks, this project sees a total of 12 individual tasks that are added across 7 versions.

Appendix 3: Perils in Using GitHub Data and the Mitigation Methods We Adopted

Based on a study of the quality and properties of data available on GitHub, Kalliamvakou et al. (2014) listed nine perils that researchers need to consider when using GitHub data. The perils and the methods we adopted to overcome them are as follows.

Peril 1: A repository is not necessarily a project: In GitHub, it is common to *fork* (clone or copy) the base repository, thereby creating a new forked repository in which development

can be done independently. Once development in the forked repository is completed, the changes are merged into the base repository after undergoing peer review(s), using one or more pull requests. Because of this distributed form of development, studies of software development in GitHub projects should include a study not only of the base repositories but also of all the forked repositories that are eventually merged into the base repository (Kalliamvakou et al. 2014). We addressed this peril by analyzing the commits that occurred within each project fork that eventually resulted in a pull request.

Peril 2: Most projects have very few commits: Analyses of GitHub projects indicate that the median number of commits across projects is six (Kalliamvakou et al. 2014). To address the problem of projects with low development activity, we selected projects that had a minimum of 10 tasks. We further mitigated the potential impact of this peril through the way we operationalized the degree of superposition. By using the ratio of the number of versions of the project to the number of tasks, the degree of superposition operationalizes the average number of tasks within each version rather than the total number of tasks in the project.

Peril 3: Most projects are inactive: Our research studied the mechanisms through which tasks are added during the active life of a project. Thus, we measured the degree of superposition only for the duration of the active life of the FLOSS project. Further, in our analysis, we attempted to control the effects of project completion through the project completion flag. This flag indicated whether the project ended during the data collection period (i.e., in 2014) and became inactive.

Peril 4: A large portion of repositories are not for software development: To ensure that our sample comprised only software development projects, we manually analyzed the project description of every project in our sample. During this exercise, we rejected non-software-development projects such as personal stores, mirrors, lists, tutorials, etc. Further, using GitHub metadata, we removed projects that did not list a programming language in the

repository description. Last, we removed projects that did not include lines of code, thus eliminating projects that did not include any programming activity.

Peril 5: Two-thirds of the projects (71.6% of repositories) are personal: As with peril 4, we were able to remove projects that were personal by analyzing the project descriptions of all of the projects in our sample. Further, we selected projects that had a minimum of 3 contributors and at least 1 fork. These filters helped remove projects that were intended to be personal, without any collaborative activity.

Peril 6: Only a fraction of all projects uses pull requests, and among those that use them, the use is very skewed: Depending on how many contributors receive push access, the shares of tasks added by push and pull request events may differ. Thus, research on GitHub data should ideally not restrict itself to either push or pull request events. To overcome this peril, we identified tasks based on both push and pull request events, as contributions to projects on GitHub can occur through both types of event. The method we adopted to identify tasks based on push and pull request event data is detailed in Appendix 1.

Peril 7: If the commits in a pull request are reworked (in response to comments), GitHub records only the commits that are the result of the peer review, not the original commits: Since we operationalized the degree of superposition as the ratio of the number of project versions to the number of tasks, we restricted our analysis to the task level and not the level of commits. According to Howison and Crowston's (2014) definition, a task is expected to result in a change in the shared outcome of the project. If commits within a pull request are reworked, there will be no effect on the identification of tasks, since the reworked commits represent the same task. Thus, reworks of commits in a pull request are counted as activities within the same task.

Peril 8: Most pull requests appear as nonmerged even if they are actually merged: A sample study of GitHub projects found that although more than 80% of all pull requests are

at least partially merged, GitHub identifies only 35 to 65% of the total number of pull requests as having been merged (Kalliamvakou et al. 2014). In light of this, it is not sufficient to depend on GitHub metadata to determine whether a pull request has been merged.

Merging of a pull request (partial or complete) results in the creation of a push event, irrespective of whether GitHub identifies the pull request as merged. By including push events in our analysis, we ensured that tasks associated with merged pull requests were captured even if the pull request was not identified as merged by GitHub. The scripts that we wrote ensured that (i) the task from a merged pull request was captured either through the pull request event or through its corresponding push event, and (ii) there was no double counting when a single task created both a pull request and a push event.

Peril 9: Many active projects do not conduct all their software development in GitHub: Some of the projects in GitHub might have some or all development work occurring outside the GitHub. To remove projects that were mirrors of projects being developed in other hosting environments, we read the project descriptions for an indication that a project was a mirror. Those projects that were identified as mirrors were not included in the main regression model.

Titre : Essais Sur la Création de Valeur Appliquée au Phénomène du Logiciel Libre: Comprendre l'Influence des Structures de Travail, de la Composition des Équipes et des Idéologies de la Communauté

Mots clés : Logiciel open source, collaboration, équipes virtuelles, motivation, structure de travail, idéologies, gouvernance, réussite du projet

Résumé. Cette thèse, composée de trois essais, explore les mécanismes de création de valeur associés aux structures de travail, à la composition d'équipes, ainsi qu'aux idéologies communautaires de projets de logiciels libres. Le premier essai examine la nature unique du travail open source, dominé par la superposition séquentielle de tâches individuelles. Cet essai théorise les mécanismes de motivation associés aux structures de travail des projets open source et examine leur influence sur la réussite des projets. Alors que le premier essai établit l'importance de l'organisation du travail par tâche dans les projets open source, le deuxième essai élargit l'enquête sur le rôle de la composition de l'équipe et de la gouvernance dans la réussite du projet. S'appuyant sur les théories de la coordination et de la gouvernance des réseaux, cet article étudie l'influence des restrictions d'accès au code source imposé aux membres de l'équipe pour atténuer les problèmes de coordination. Le troisième essai poursuit une vision globale de la communauté open source en examinant les fondements idéologiques de la communauté et étudie son influence sur le succès du projet. L'essai examine deux changements idéologiques observés dans la communauté de l'open source qui ont modifié les convictions d'ouverture et de prévention de l'appropriation commerciale, sur lesquels le phénomène de l'open source a été fondé.

Title: Essays on Value Creation in the Open Source Phenomenon: Understanding the Influence of Work Structures, Team Composition, and Community Ideologies

Keywords: Open source software, collaboration, virtual teams, motivation, work structure, ideologies, governance, project success

Abstract. This dissertation comprising three essays explores the value creation mechanisms associated with the work structures, team composition, and community ideologies of open source software projects. The first essay examines the unique nature of open source work which is dominated by the sequential layering of individual tasks. This essay theorizes the motivational mechanisms associated with the work structures of open source projects and examines their influence on project success. While the first essay establishes the importance of task-work organization in open source projects, the second essay expands the inquiry into the role of team composition in the project's success. Building on the theories of coordination and network governance, this essay studies the influence of source code access restrictions imposed on team members in mitigating coordination challenges. The third essay pursues an overarching view of the open source community by examining the ideological foundations of the community and studies its influence on project success. The essay scrutinizes two ideological shifts seen in the open source community that have altered the beliefs of 'openness' and 'prevention of commercial appropriation', on which the open source phenomenon was founded.