Information Systems Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Does Superposition Influence the Success of FLOSS Projects? An Examination of Open-Source Software Development by Organizations and Individuals

Poonacha K. Medappa, Shirish C. Srivastava

To cite this article:
Poonacha K. Medappa, Shirish C. Srivastava (2019) Does Superposition Influence the Success of FLOSS Projects? An Examination of Open-Source Software Development by Organizations and Individuals. Information Systems Research

Published online in Articles in Advance 02 Aug 2019

. https://doi.org/10.1287/isre.2018.0829

http://pubsonline.informs.org/journal/isre/

INFORMATION SYSTEMS RESEARCH
Articles in Advance, pp. 1–23
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Does Superposition Influence the Success of FLOSS Projects? An Examination of Open-Source Software Development by Organizations and Individuals

Poonacha K. Medappa,[a] Shirish C. Srivastava[a]

[a] HEC Paris, Jouy-en-Josas 78350, France

Contact: poonacha.medappa@hec.edu, http://orcid.org/0000-0001-8872-6161 (PKM); srivastava@hec.fr, http://orcid.org/0000-0001-7586-5314 (SCS)

Received: January 26, 2017
Revised: December 22, 2017; August 20, 2018
Accepted: October 2, 2018
Published Online in Articles in Advance: August 2, 2019

https://doi.org/10.1287/isre.2018.0829

Abstract. Collaboration through open superposition describes the dominant work orchestration mechanism observed in free (Libre) and open-source (FLOSS) software, 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 toward cowork and concurrent development facilitated by a modular software design architecture. Our study 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 use 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 6,500 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 that (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.

History: Jonathon Cummings, Senior Editor; Likoebe Maruping, Associate Editor.
Funding: This work was supported by the HEC Paris Foundation.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/isre.2018.0829.

Keywords: open-source software • collaboration • virtual teams • motivation • task structure • information systems (IS) development • superposition • project success • econometrics

Introduction

In the current digitally enabled collaborative environment, free (Libre) and open-source (FLOSS) software projects have become ubiquitous. This model of development allows 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 work by Howison and Crowston (2014) and Lindberg et al. (2016) has 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 toward cowork 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% of tasks involved no cowork (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 paper 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:

Research Question 1. *How does the extent of task superposition influence FLOSS project success?*

Given that organizations are increasingly shifting their strategic focus toward FLOSS development (Microsoft News Center 2018), it is imperative to understand whether 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 us to our second research question:

Research Question 2. *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, we contribute to the theoretical understanding of the different ways in which superposition influences project success 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 (e.g., Fitzgerald 2006, Capra et al. 2011). 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). We 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 2, a and b).

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 the information systems (IS) success model of Delone and McLean (1992) 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 with the number of bugs, the number of participants in the bug trackers (Crowston et al. 2006), and the total number of developers overall (Krishnamurthy 2005, 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 (Rebeca and García 2009, Fershtman and Gandal 2011), commercial success (Grewal et al. 2006, Midha and Palvia 2012), user interest (Stewart et al. 2006, Subramaniam et al. 2009), open-source software (OSS) product awareness (Setia et al. 2012), and project viewership and downloads (Crowston 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 perspectives of codebase architecture (e.g., modularity; Baldwin and Clark 2006) and coordination mechanisms (e.g., Mockus et al. 2002, Crowston et al. 2005, Chua and Adrian 2010), 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: (1) 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, and (2) 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 this 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 (1) individual task work and (2) incremental layering of motivationally independent tasks. This unique work breakdown structure has been found to accomplish complex work through a process of "productive deferral," which 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 completing 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 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* because the layering of tasks on the work of others and the potential support from other contributors provide connectedness in a manner that does not undermine contributors' autonomy. In the next 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 (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 cowork), becomes a concern at higher degrees of superposition, where contributors tend to predominantly defer complex/big tasks rather than engage in cowork. This increased adoption of the productive deferral mechanism and avoidance of cowork 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), resulting in some frustration (Selvidge et al. 2002). Overall, this may create 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, in turn, may 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 resulting from avoidance of cowork 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 (1) the influence of superposition in increasing the positive affective state of the contributor owing to increased autonomy and (2) the influence of superposition in increasing the negative affective state of the contributor as a result of increased loss of control and frustration. We propose that the relationship between degree of superposition and the positive affective state of the contributor resulting from 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. By contrast, we propose that loss of control and frustration caused by avoidance of cowork and productive deferral mechanisms tends to manifest 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 off, whereas 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 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[1] (Figure 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 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 cowork. Hence, we hypothesize the following:

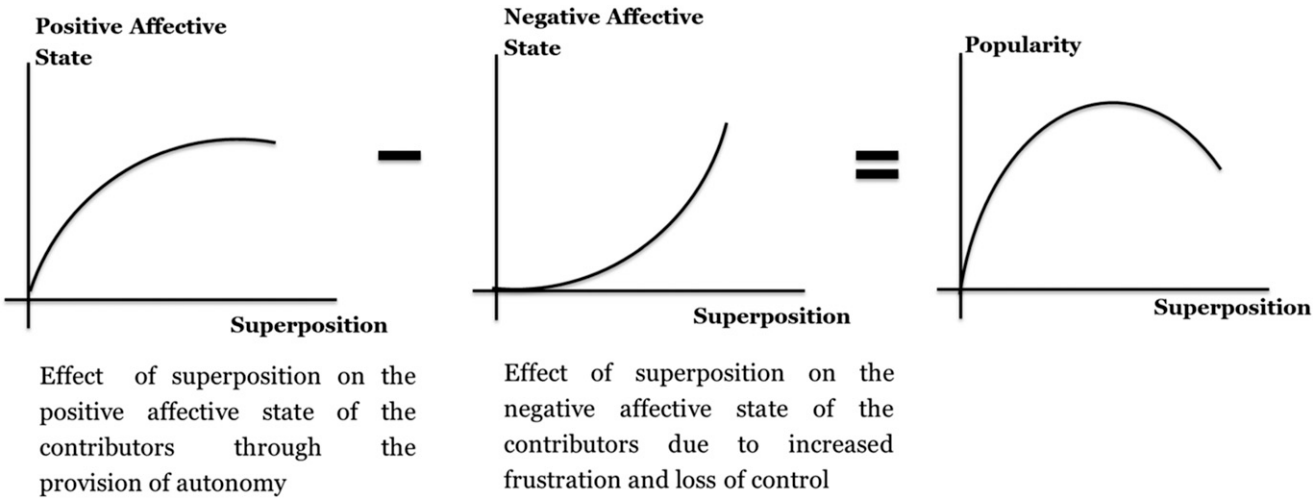
Hypothesis 1. *In the context of FLOSS projects, the degree of superposition has a nonlinear relationship with project popularity such that project popularity increases with an increase in the degree of superposition up to a particular value (the turning point). Beyond this optimal degree of superposition,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

Organizational Ownership of FLOSS Projects

By early 2000, organizations began to realize the commercial potential of FLOSS projects (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 not only has been found to provide extrinsic motivation through rewards and signaling (Lerner and Tirole 2003) but also can 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).[2] 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 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 9 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.

Figure 1. Effect of Superposition on the Popularity of FLOSS Projects Through Its Latent Influence on the Affective State of the Contributors



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 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 cowork rather than relying on productive deferral to complete the task, making such deferral less of a factor.[3] 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 because of 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 (Nondevelopment) Model. In the support model, organizations provide financial, infrastructure, marketing, and/or customer support (Capra et al. 2011). Each of the nondevelopmental 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 resulting from productive deferral and avoidance of cowork have a larger negative influence on project popularity. Thus, in the case of organization-owned projects, increased productive deferral and avoidance of cowork associated with a high degree of superposition not only may lead to a negative affective state for the contributors, because of frustration and loss of control, but also may increase the time cost of money for their investments. The increase in the time cost of money can deincentivize the organization owner from making future investments in the project, leading to a reduced focus toward adding new functionality and improving the 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 with 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 compared with projects 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 in troduction of management practices that tend to transform the adopted model of development from a bazaar into a cathedral-like form. By contrast, 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 the increased time cost of money and that there are extrinsically motivated contributors who have less need for autonomy, 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 cowork 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 are observed (1) as a flattening of the inverted U-shaped curve and (2) as a left shift of the turning point. Table 1 summarizes the moderating influence of organizational ownership through the different models of engagement and its expected influence on the shape of the curve.

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, 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 the following:

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.*

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.[4] Inclusion of the negative linear mechanism largely results in the left shift of the turning point of the curve (Figure 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

Table 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 cowork in the presence of extrinsically motivated contributors and an increased time cost of money	Facilitates Hypotheses 2a and 2b

individual-owned FLOSS projects. Hence, we hypothesize the following:

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.*

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 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

Figure 2. Effect of Increase in Net Extrinsic Motivation of the Contributors When Organizations Own FLOSS Projects

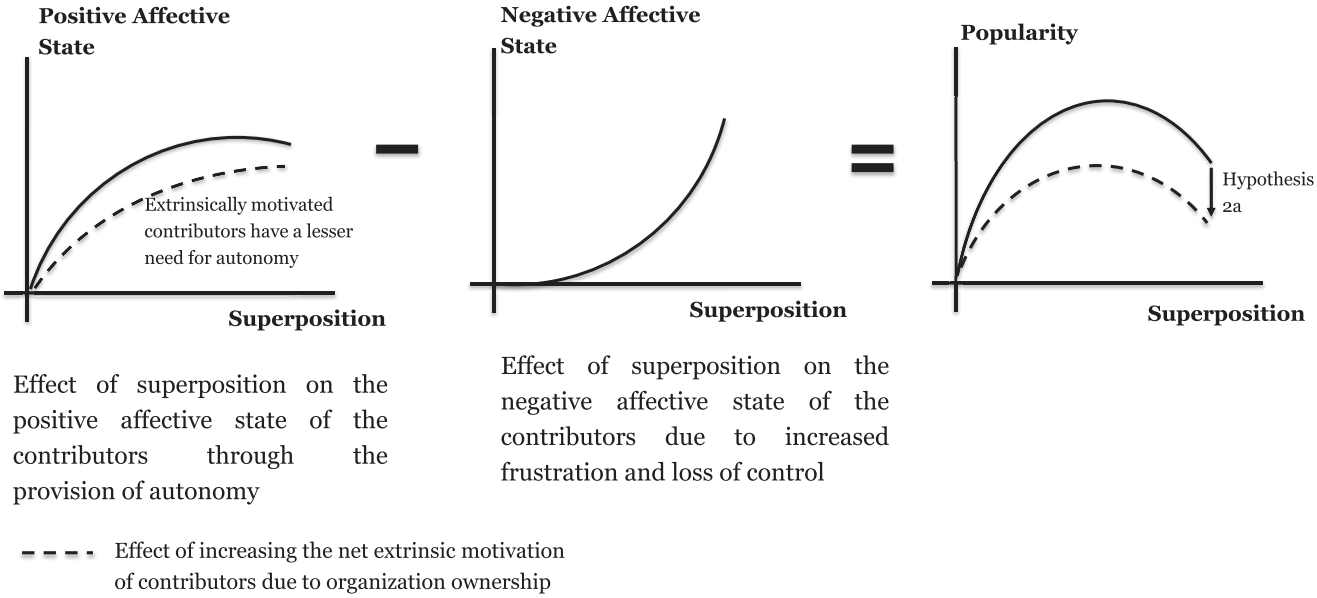
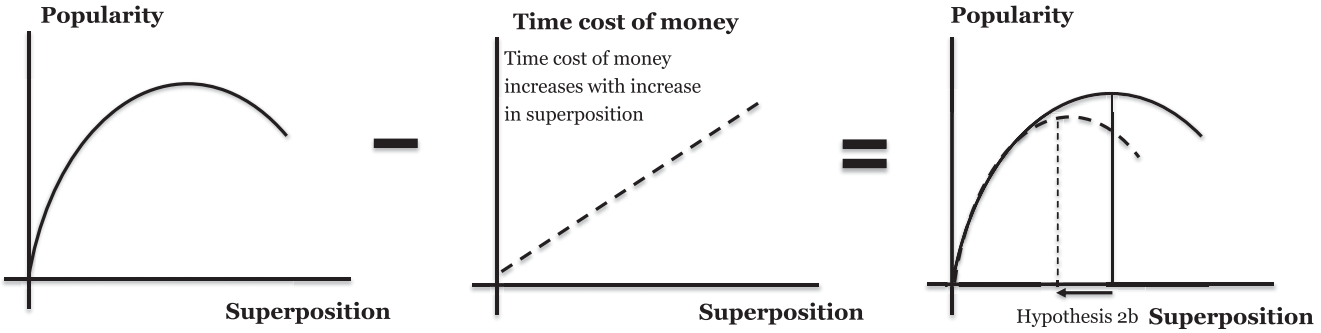


Figure 3. Effect of Time Cost of Money when Organizations Own FLOSS Projects



Non-linear effect of superposition on popularity of the project (Hypothesis 1)

Increasing effect of superposition on the time cost of money

Effect of introducing the time cost of money due to organization ownership

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 into 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. Online Appendix A1 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 such as GitHub, we can define *version* as the snapshot of a 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 with 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. Online Appendix A2 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: (1) task work is accomplished by individual contributors, (2) individual tasks are sequentially layered on top of one another, and (3) productive deferral occurs in order to accomplish complex tasks through (1) and (2). 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 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 that 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 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 15. The addition of tasks A and B results in version 1 of the project on July 17. Task C, developed by three contributors, builds on the version that was available on July 17 (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 July 21 (version 2), creating version 3 of the project. This example has six tasks (tasks A–F) and a total of 12 individual task contributions that result in the three versions; hence, the degree of superposition is 3/12, or 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 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 GitHub (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

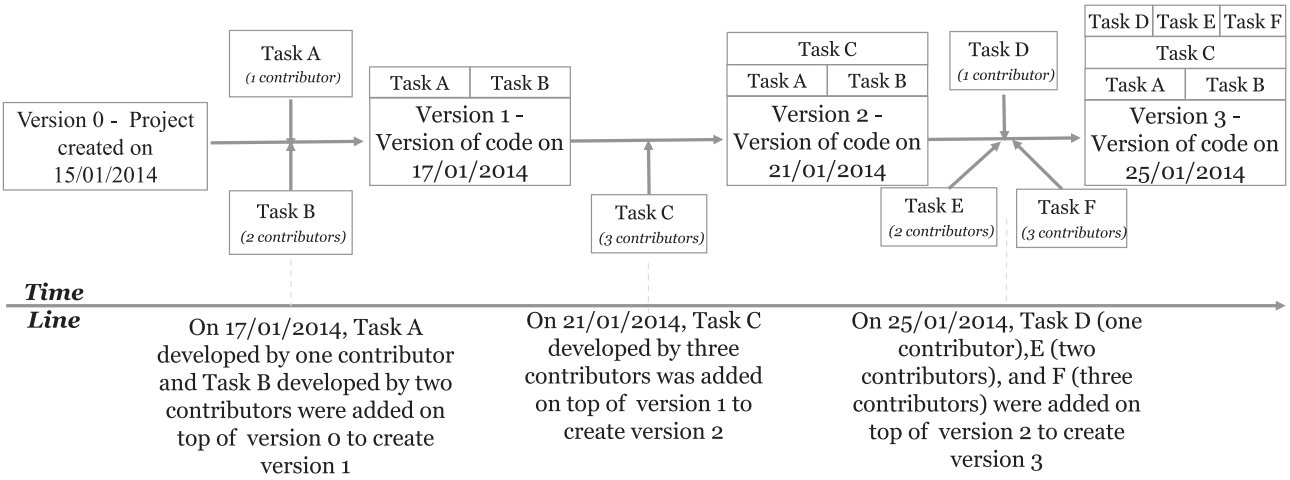
We employed Google's bigquery tool to query the archived project log data available in the GH Archive database (Grigorik 2012). Because this database is large (~432 GB, with 134 million rows for the year 2014 alone; Google 2014), we needed the bandwidth provided by a tool like Google's bigquery to run 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 data set, care needs to be taken to overcome common perils in using this data set (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 Online Appendix A3. After filtering the data set to address the perils, we were left with a sample of 6,571 FLOSS projects, of which 3,086 are individual owned and 3,485 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.

Figure 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

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 to keep track of those they find interesting and also to show their appreciation for these projects (GitHub 2017a, Borges and Tulio Valente 2018). 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 sees 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 to which they belong. Online Appendices A1 and A2 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 of 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 a 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*, that 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 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 application programming interface. By accessing this GitHub-determined project attribute, we identify whether a project is owned by an organization or an individual.

Control Variables. GitHub maintains detailed project- and user-level data, which allow 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 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 in which the owner participates. The existence and density of prior ties between the project owner and contributors have 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 cowork 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 with 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, by contrast, 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 versus 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 the 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 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 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, 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.

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 3 provides the results of the regression models we employed. Models 1a–c provide the results of the stepwise log-linear OLS regression analysis. Models 2a–c 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 owing 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, 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 our models (Hayes and Cai 2007).

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

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

	Variable	Mean	Standard deviation	1	2	3	4	5	67	7	8	9	10	11	12
1	*Article I. Stars*	194.687	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	*Number 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 contributor*	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**	−0.017	−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**	−0.015	−0.087**	0.030*

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

independent variable (*degree of superposition*) and its squared term. It is essential to retain the linear term *degree of superposition* (Aiken and West 1991) because 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 *degree of superposition*. From Table 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 (Lind and Mehlum 2010, Haans et al. 2015). 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, whereas those at the high end of the range (i.e., when the degree of superposition is 1) should be negative and significant (Lind and Mehlum 2010, Haans et al. 2015,). We tested Model 1b to confirm that the slope when the degree of superposition is 0 is positive and significant ($dy/dx_{x=0} = +2.866, p < 0.01$) and that when the degree of superposition is 1, the slope is negative and significant ($dy/dx_{x=1} = -2.023, p < 0.01$). 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 $-(\beta_1/2\beta_2)y = a^*x^2 + b^*x + c - b/(2^*a)$.[5] Because 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 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

Table 3. Results of Moderated Regression Analysis

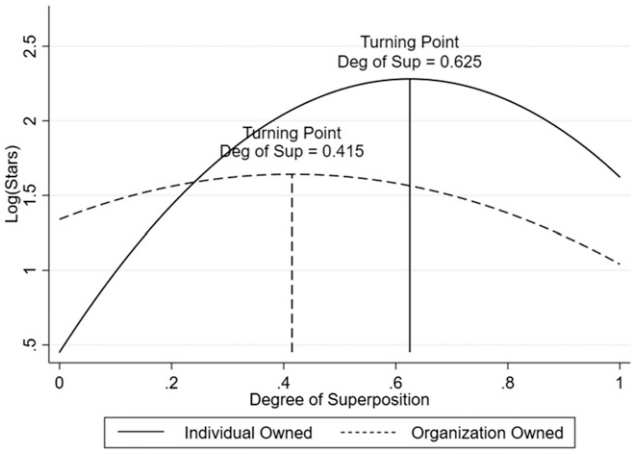
	Model: OLS			Model: Negative binomial		
	Dependent variable: Log(*stars*)			Dependent variable: *Stars*		
	Model 1a: Control variables	Model 1b: Main effects	Model 1c: Interaction effects	Model 2a: Control variables	Model 2b: Main effects	Model 2c: Interaction effects
Number of programming languages	−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 (p_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 × *project ownership flag* (β_4)			−4.398 (0.00)**			−6.726 (0.00)**
Degree of superposition squared × *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^2	0.133	0.137	0.1797	—	—	—
ΔR^2		0.004**	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

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

type with the degree of superposition and its squared term in Model 1c (Table 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 a 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 *degree of superposition* (β_5; Haans et al. 2015). From Model 1c (Table 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 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 Model 1c (Table 3). The flattening effect can be observed in the plots, with

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



organization-owned FLOSS projects exhibiting a relatively flat curve compared with 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). Because the relationship is quadratic, the turning points are given by $-\beta_1/(2*\beta_2)$ for individual-owned projects ($z = 0$) and $-(\beta_1 + \beta_4)/(2*(\beta_2 + \boldsymbol{\beta}_5))$ for organization-owned projects ($z = 1$).[6] 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 shown in Figure 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).

Negative Binomial Regression Model

Taking into account the overdispersed nature of the number of stars, we estimated the negative binomial model (Model 2; Table 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 Hypotheses 1, 2a, and 2b. We find support for the three hypotheses in the negative binomial model. Because 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, we present three robustness tests in the appendix: (1) tests for model specification, (2) a test for the dependent variable, and (3) a test for the nature and treatment of data. In tests for model specification, we test whether 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 include 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.

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 have received only limited attention from previous researchers (Howison and Crowston 2014). Considering the steady shift toward FLOSS projects 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, whereas for organization-owned projects it is found to increase from 28.26 to 38.08 (see Model 1c, Table 3). As mentioned in the initial part of this 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 et al. 2006). And some of these users, over time, may progress to become active developers for the project (Krishnamurthy 2005, 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 cowork 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,[7] 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.

Implications

Our research contributes to IS and organization theory in three ways. First, our study advances the existing literature on motivation (e.g., Ryan and Deci 2000, Ke and Zhang 2010, von Krogh et al. 2012) and work structures (e.g., Howison and Crowston 2014, Lindberg et al. 2016) in FLOSS projects because 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 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., Lakhani and Wolf 2005, Roberts et al. 2006, Ke and Zhang 2010), 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 because 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 (Fitzgerald 2006, Stewart et al. 2006, Wagstrom 2009, Capra et al. 2011, Spaeth et al. 2015) 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"[8] 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), which, 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 on 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, the 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 (Sutton and Staw 1995; Suddaby 2010). As a step toward 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—that is, 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 not only will help researchers to advance the theory of superposition and unearth new relationships but also can 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 data set 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 data set and the checks we adopted to overcome them are listed in Online Appendix A3. 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 Online Appendix A1 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 Online Appendix A2 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.[9] 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 appendix) 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 the popularity of a project. 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 with each other but that project lifespan was unique in not correlating with 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 the appendix 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 such as 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, 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. Because 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.

Acknowledgments

The authors gratefully acknowledge the excellent guidance received from the senior editor and the associate editor. Their deep engagement with our work and the constructive feedback from the entire review team have been instrumental enhancing the value of our work.

Equal contribution by both authors, names listed alphabetically.

Appendix. 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 (see Table 3). We found that inclusion of the cubic term does not increase the model fit ($\Delta R^2 = 0.000, p > 0.10$) and that the coefficient of the cubic term is not significant ($p > 0.10$).

Test for Outliers

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 reestimate the model (Haans et al. 2015). We adopted Cook's distance (Cook 1977) to eliminate outliers from Models 1b and 1c (see Table 3). On reestimating 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 with each other but that project lifespan was unique because it did not correlate with 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. 2008). In this model (Table A.1), 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.

We use the semiparametric Cox regression model for the survival analysis because it relaxes the specification requirement for the baseline hazard function (Hosmer et al. 2008). Because 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 A.1) 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 among 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, cowork does not undermine the autonomy afforded to the contributor because 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

Table A.1. Results of the Survival Analysis

	Model—Survival analysis (dependent variable: *likelihood of project inactivity*)		
	Model 3a: Control variables	Model 3b: Main effects	Model 3c: Interaction effects
Number 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 × project ownership flag (β_4)			4.402 (0.00)**
Degree of superposition squared × 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^2	—	—	—
ΔR^2	—	—	—
A\C	51851.89	51854.33	51772.97
N	6,571	6,571	6,571

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

from GitHub. In this sample, the presence of an inverted U-shaped relationship between degree of superposition and project popularity would indicate that either (1) the nature of the data and the operationalization of the constructs could cause an inverted U-shaped relationship or (2) 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 because 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 A.2 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 ($\Delta R^2 = 0.005, p > 0.10$) and that the coefficient of the quadratic term is not significant ($p > 0.10$). 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 (see Table 3, Models 1b and 1c) being due to the theorized mechanisms rather than an outcome of the nature and treatment of the data.

Table A.2. Results of the Falsification Test

	Dependent variable: Log(*stars*)	
	Model 4a: Linear model	Model 4b: Quadratic model
Total contributors	0.007	0.009
	(0.04)*	(0.04)*
Log(*size of project*)	0.086	0.089
	0.205	0.187
Average commits per contributor	–0.002	–0.002
	(0.02)*	(0.04)*
Project completed flag	0.617	0.572
	(0.03)*	(0.05)*
Restrictive license regime	0.266	0.354
	0.436	0.305
Log(*average task complexity*)	0.035	0.021
	0.821	0.894
Owner experience	0	0
	0.655	0.505
Average idle time	0.005	0.007
	0.394	0.271
Project ownership flag	–1.049	–1.059
	(0.00)**	(0.00)**
Degree of superposition (β_1)	–2.233	1.37
	(0.00)**	0.638
Degree of superposition squared (β_2)		–2.74
		0.194
Fixed effects of month of creation	Yes	Yes
R^2	0.2342	0.2395
ΔR^2	—	0.005
N	260	260

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

Endnotes

[1] 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.

[2] We thank one of the anonymous reviewers for suggesting this reference for enriching our theory development.

[3] We thank the AE for helping us expand the theoretical understanding of this model of organizational engagement.

[4] 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: $PV = FV - FVtr$. In our case, we consider the time cost of money as the value of $FVtr$. 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.

[5] For a quadratic equation of the form $y = ax^2 + bx + c$, the turning point is given by $-b/2a$.

[6] The full regression model with interaction (see Table 3, Model 1c) can be represented by

$$y = \beta_1 x + \beta_2 x^2 + \beta_3 z + \beta_4 xz + \beta_5 x^2 z + \beta_i \times Controls_i + Const.,$$

where
y = *popularity of the project*
x = *degree of superposition*
z = *project ownership flag* (1 if organizational owned, 0 if individual owned) βs = *regression coefficients*.

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

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

[8] We thank one of the reviewers for helping us better articulate the contribution of our research.

[9] We thank one of the anonymous reviewers for pointing out the limitation with our operationalization of versions and suggesting ways to improve the clarity surrounding the construct.

References

Aiken LS, West SG (1991) *Multiple Regression: Testing and Interpreting Interactions* (SAGE Publications, Newbury Park, CA).

Baldwin CY, Clark KB (2006) The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Sci.* 52(7):1116–1127.

Borges H, Tulio Valente M (2018) What's in a GitHub Star? Understanding starring practices in a social coding platform. *J. Systems Software* 146:112–129.

Cameron AC, Trivedi PK (2013) *Regression Analysis of Count Data* (Cambridge University Press, Cambridge, UK).

Capra E, Francalanci C, Merlo F, Rossi-lamastra C (2011) Firms' involvement in open source projects: A trade-off between software structural quality and popularity. *J. Systems Software* 84:144–161.

Carte TA, Russel CJ (2003) In pursuit of moderation: Nine common errors and their solutions. *MIS Quart.* 27(3):479–501.

Chua CEH, Adrian YKY (2010) Artifacts, actors, and interactions in the cross- project coordination practices of open-source communities. *J. Assoc. Information Systems* 11(12):838–867.

Colazo JA, Fang Y, Neufeld D (2005) Development success in open source software projects: Exploring the impact of copylefted licenses. *Americas Conf. Inform. Systems* (AIS, Atlanta), 929–936.

Cook RD (1977) Detection of influential observation in linear regression. *Technometrics* 19(1):15–18.

Coverity Inc. (2013) Coverity Scan 2013 open source report. Accessed January 4, 2017, http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf.

Crowston K, Howison J, Annabi H (2006) Information systems success in free and open source software development: Theory and measures. *Software Process Improv. Pract.* 11(2):123–148.

Crowston K, Scozzi B (2004) Coordination practices within FLOSS development teams: The bug fixing process. *Proc. 1st Internat. Workshop Comput. Supported Activity Coordination, CSAC 2004, Porto, Portugal*, 21–30.

Crowston K, Wei K, Howison J, Wiggins A (2012) Free/Libre open-source software development. *ACM Comput. Survey* 44(2):1–35.

Crowston K, Wei K, Li Q (2005) Coordination of free/libre and open source software development. *26th Internat. Conf. Inform. Systems* (AIS, Atlanta), 181–193.

Deci EL, Koestner R, Ryan RM (1999) A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psych. Bull.* 125(6):627–668.

Delone WH, Mclean ER (1992) Information systems success: The quest for the dependent variable. *Information Systems Res.* 3(1):60–95.

Engeström Y (2007) From communities of practice to mycorrhizae. Hughes J, Jewson N, Unwin L, eds. *Communities of Practice—Critical Perspectives* (Routledge, London), 1–20.

Fershtman C, Gandal N (2004) The determinants of output per contributor in open source projects: An empirical examination. CEPR Discussion Papers 4329, Centre for Economic Policy Research, London.

Fershtman C, Gandal N (2011) Direct and indirect knowledge spillovers: The "social network" of open-source projects. *RAND J. Econom.* 42(1):70–91.

Fitzgerald B (2006) The transformation of open source software. *MIS Quart.* 30(3):587–598.

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

GitHub (2017b) Glossary—Commit. Retrieved January 1, 2017, https://help.github.com/articles/github-glossary/#commit.

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

Gousios G, Pinzger M, Van Deursen A (2014) An exploratory study of the pull-based software development model. *Proc. 36th Internat. Conf. Software Engrg.* (ACM, New York), 345–355.

Greenberger DB, Strasser S (1986) Development and application of a model of personal control in organizations. *Acad. Management Rev.* 11(1):164–177.

Grewal R, Lilien GL, Girish M (2006) Location, location, location: How network embeddedness affects project success in open source systems. *Management Sci.* 52(7):1043–1056.

Grigorik I (2012) The GitHub archive. Retrieved January 1, 2017, https://www.githubarchive.org/.

Guenter H, van Emmerik IJH, Schreurs B (2014) The negative effects of delays in information exchange: Looking at workplace relationships from an affective events perspective. *Human Resources Management Rev.* 24(4):238–298.

Haans JFR, Pieters C, He ZL (2015) Thinking about U: Theorizing and testing U- and inverted U-shaped relationships in strategy research. *Strategic Management J.* 37:1177–1195.

Hair JF, Black WC, Babin BJ, Anderson RE (2010) *Multivariate Data Analysis*, 7th ed. (Prentice Hall, Upper Saddle River, NJ).

Hayes AF, Cai L (2007) Using heteroskedasticity-consistent standard error estimators in OLS regression: An introduction and software implementation. *Behav. Res. Methods* 39(4):709–722.

Hosmer DW, Lemeshow S, May S (2008) *Applied Survival Analysis. Regression Modeling of Time-to-Event Data* (John Wiley & Sons, New York).

Howison J, Crowston K (2014) Collaboration through open superposition: A theory of the open source way. *MIS Quart.* 38(1):29–50.

Jarczyk O, Gruszka B, Jaroszewicz S, Bukowski L (2014) GitHub projects: Quality analysis of open-source software. *6th Internat. Conf. Soc. Informatics* (Springer International, Cham, Switzerland), 80–94.

Jarvenpaa SL, Lang KR (2011) Boundary management in online communities: Case studies of the nine inch nails and ccMixter music remix sites. *Long Range Planning* 44(5–6):440–457.

Jones C, Hesterly WS, Borgatti SP (1997) A general theory of network governance: Exchange conditions and social mechanisms. *Acad. Management Rev.* 22(4):911–945.

Kalliamvakou E, Gousios G, Singer L, Blincoe K, German DM, Damian D (2014) The promises and perils of mining GitHub. *Proc. 11th Workshop Conf. Mining Software Repositories* (ACM, New York), 92–101.

Ke W, Zhang P (2010) The effects of extrinsic motivations and satisfaction in open source software development. *J. Assoc. Inform. Systems* 11(12):784–808.

Krishnamurthy S (2005) Cave or community? An empirical examination of 100 mature open source projects. Accessed May 1, 2019, https://firstmonday.org/ojs/index.php/fm/article/view/1477/1392.

von Krogh G, Haefliger S, Spaeth S, Wallin MW (2012) Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quart.* 36(2):649–676.

Lakhani K, Wolf RG (2005) Why hackers do what they do: Understanding motivation and effort in free/open source software projects. Feller J, Fitzgerald B, Hissam S, Lakhani KR, eds. *Perspectives on Free and Open Source Software* (MIT Press, Cambridge, MA), 3–22.

Lee SYT, Kim HW, Gupta S (2009) Measuring open source software success. *Internat. J. Management Sci.* 37:426–438.

Lerner J, Tirole J (2003) Some simple economics of open source. *J. Indust. Econom.* 50(2):197–234.

Lind JT, Mehlum H (2010) With or without U? The appropriate test for a U-shaped relationship. *Oxford Bull. Econom. Statist.* 72(1):109–118.

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, Cleveland).

Lindberg A, Berente N, Gaskin J, Lyytinen K (2016) Coordinating interdependencies in online communities: A study of an open source software project. *Inform. Systems Res.* 27(4):751–772.

Malone TW, Crowston K (1994) The interdisciplinary study of coordination. *ACM Comput.* 26(1):87–119.

Michlmayr M, Fitzgerald B (2012) Time-based release management in free and open source (FOSS) projects. *Internat. J. Open Source Software Processes* 4(1):1–19.

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

Midha V, Palvia P (2012) Factors affecting the success of open source software. *J. Systems Software* 85(4):895–905.

Mockus A, Fielding RT, Herbsleb J (2002) Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Software Engrg. Methodology* 11(3):309–346.

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

O'Mahony S (2005) Nonprofit foundations and their role in community-firm software collaboration. Fitzgerald B, Feller J, Hissam SA, Lakhani KR, eds. *Perspectives on Free Open Source Software* (MIT Press, Cambridge, MA), 393–413.

Pearce J (2014) 9.9 Million lines of code and still moving fast—Facebook Open Source in 2014. Retrieved December 11, 2017, https://code.facebook.com/posts/292625127566143/9-9-million-lines-of-code-and-still-moving-fast-facebook-open-source-in-2014.

Phillips PCB, Park JY (1988) On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56(5):1065–1083.

Rao CR (1972) *Linear Statistical Inference and Its Applications*, 2nd ed. (John Wiley & Sons, New York).

Raymond E (1998) The cathedral and the bazaar. *First Monday* 3(3).

Rebeca MD, García CE (2009) Returns from social capital in open source software networks. *J. Evolutionary Econom.* 19(2):277–295.

Roberts JA, Hann IH, Slaughter SA (2006) Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Sci.* 52(7):984–999.

Ryan RM, Deci EL (2000) Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *Amer. Psych.* 55(February):68–78.

Rynes SL, Bartunek JM, Daft RL (2001) Across the great divide: Knowledge creation and transfer between practitioners and academics. *Acad. Management J.* 44(2):340–355.

Schaarschmidt M, Gianfranco W, Von Kortzfleisch H (2015) How do firms influence open source software communities? A framework and empirical analysis of different governance modes. *Inform. Organ.* 25(2):99–114.

Selvidge PR, Chaparro BS, Bender GT (2002) The world wide wait: Effects of delays on user performance. *Internat. J. Indust. Ergonomics* 29(1):15–20.

Sen R, Subramaniam C, Nelson ML (2008) Determinants of the choice of Open Source Software License. *J. Management Inform. Systems* 25(3):207–240.

Setia P, Rajagopalan B, Sambamurthy V, Calantone R (2012) How peripheral developers contribute to open-source software development. *Inform. Systems Res.* 23(1):144–163.

Singh PV, Phelps C (2013) Networks, social influence and the choice among competing innovations: Insights from open source software licenses. *Inform. Systems Res*. 24(3):539–560.

Spaeth S, von Krogh G, He F (2015) Perceived firm attributes and intrinsic motivation in sponsored open source software projects. *Inform. Systems Res.* 26(1):224–237.

Stewart KJ, Ammeter AP, Maruping LM (2006) Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. *Inform. Systems Res.* 17(2):126–144.

Subramaniam C, Sen R, Nelson ML (2009) Determinants of open source software project success : A longitudinal study. *Decision Support Systems* 46(2):576–585.

Suddaby R (2010) Editor 's comments : Construct clarity in theories. *Acad. Management Rev.* 35(3):346–357.

Sutton RI, Staw BM (1995) What theory is not. *Admin. Sci. Quart.* 40(3):371–384.

Teigland R, Di Gangi PM, Flåten BT, Giovacchini E, Pastorino N (2014) Balancing on a tightrope: Managing the boundaries of a firm-sponsored OSS community and its impact on innovation and absorptive capacity. *Inform. Organ.* 24(1):25–47.

Tsay J, Dabbish L, Herbsleb J (2014) Influence of social and technical factors for evaluating contribution in GitHub. *36th Internet. Conf. Software Engrg.* (ACM, New York), 356–366.

Wagstrom PA (2009) *Vertical Interaction in Open Software Engineering Communities* (Carnegie Mellon University, Pittsburgh).

Weiss HM, Cropanzo R (1996) Affective events theory: A theoretical discussion of the structure, causes and consequences of affective experiences at work. *Res. Organ. Behav.* 18:1–74.

Zhang P (2013) The affective response model: A theoretical framework of affective concepts and their relationships in the ICT context. *MIS Quart.* 37(1):247–274.