The Contingent Role of Interproject Connectedness in Cultivating Open Source Software Projects

Sutanto, Juliana; Jiang, Qiqi ; Tan, Chuan-Hoo

Document Version
Accepted author manuscript

Published in:
The Journal of Strategic Information Systems

DOI:
10.1016/j.jsis.2020.101598

Publication date:
2021

The Contingent Role of Interproject Connectedness in Cultivating Open Source Software Projects

Abstract: A better understanding of the key to successful open-source software (OSS) development continues to motivate research. Aligned with work that builds on the notion that an OSS development is tightly interrelated with its social environment (i.e., the OSS community), this study examines the relationship between interproject structure and OSS project success. OSS project success is reflected in two forms: popularity and knowledge creation. Extending the extant OSS literature, we theorize a contingent role of interproject connectedness. In particular, we posit three points: (1) an OSS project with more structural holes achieves higher popularity; (2) an OSS project with fewer structural holes yields higher knowledge creation; and (3) these two relationships are enhanced by an increase in project maturity. Using a dataset longitudinally collected from SourceForge.net, we found that OSS projects with widespread connectedness are more popular. This is especially so for those OSS projects in the mid-mature stage. We also found that OSS projects with a cohesive network achieve higher knowledge creation, irrespective of their maturity. Findings from our study can contribute to OSS literature by identifying OSS projects that are more likely to be successful.

Keywords: open source software, interproject connectedness, maturity, popularity, knowledge creation

1 Introduction

A recent report estimates that the economic value of open-source software (OSS) development could exceed US$32 billion by the year 2023[1]. OSS development forges, web-based collaborative software platforms for both developing and sharing OSS such as Sourceforge and Github, are an integral part of software innovation. Major technological titans, such as Amazon, Facebook, Apple, Alibaba, and Microsoft[2], have also tapped into OSS development forges for their software innovation. Unique to OSS development forges is that OSS projects are formed by globally distributed people; this enables the projects to gain access to an unlimited pool of IT talents. Unfortunately, few OSS projects achieve success (Chengalur-Smith and Sidorova 2003; Lin et al. 2017). OSS project success can be reflected in terms of popularity and knowledge creation (Crowston et al. 2007; Subramaniam et al. 2009). The question is then what kinds of OSS projects are more likely to be successful?

To gain an understanding of the key to successful OSS development, it is important to recognize that contribution to OSS projects is voluntary (von Hippel and von Krogh 2003) and contributors can freely participate in any and even multiple OSS projects[3] (Grewal et al. 2006; Tan et al. 2007). When developers contribute to multiple OSS projects, they connect these projects. OSS projects that are connected with common developers benefit from having these developers who facilitate the sharing of knowledge and expertise. Specifically, the shared people promote the diffusion of knowledge by having more people know about an OSS project (which

[1] https://www.marketresearchengine.com/open-source-services-market [Last access 10th Sep 2019]
[2] https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/ [Last access 10th Sep 2019]
[3] Isolated developers (i.e., developers with a single project) exist in the OSS context (Gao and Madey 2007), and developers do tend to interact only with prominent developers (Shen and Monge 2011). Hence, not all projects are interlinked or intensively interlinked. In this research, we focus on projects that are connected through common contributors.

could influence popularity). The shared people also facilitate the reuse of source codes from other OSS projects (which could influence knowledge creation).

To the best of our knowledge, the connectedness of an OSS project with other projects (i.e., the network structural characteristic of an OSS project) has been less studied despite its importance[4]. We found three studies looking at the connectedness among OSS projects (i.e., Grewal et al. 2006; Singh 2010; Singh et al. 2011). Although these three studies applied different social network measures, the fundamental argument in these studies is the importance of embracing a structural social capital perspective (Grewal et al. 2006; Singh 2010; Singh et al. 2011). In social network analysis, the vertexes are OSS projects and the vertexes are connected because of shared contributors (i.e., administrators or developers). The structural social capital perspective suggests that information access hinges upon the overall pattern of social network connections (Seibert et al. 2001); an OSS project's connectedness to others gives the project access to certain types of information. However, despite the three studies mentioned above, how connectedness can translate to project success in terms of increasing the popularity of the OSS project and creating knowledge remains unclear. A primary reason for this lack of clarity is the different possible trajectories of popularity and knowledge creation.

A popular OSS project (i.e., one that is successful in the marketplace) can reach out to more users by meeting users' various requirements for features and functionalities. To gain popularity, an OSS project needs to enrich its spectrum of generated ideas and have more heterogeneous contributors who, as a whole, promote debate and discourse (Harrison and Klein 2007; Ren et al. 2016; van Knippenberg et al. 2004). In contrast to gaining popularity, knowledge creation

[4] Prior studies have identified several contributing factors for OSS project success, which include project-specific characteristics such as the types of OSS license, the leader-follower relationship, the availability of company sponsorship, the project activity, and the popularity of the programming language in which an OSS is developed (Jiang et al. 2019).

depends more upon the homogeneity of resources. An OSS project with homogeneous contributors can benefit from consistent beliefs about its development and innovation priorities (Ren et al. 2016; van Knippenberg et al. 2004). From the software development perspective, strong team cohesion is conducive to both the on-time delivery and the technological quality of software projects (Lindsjørn et al. 2016). In short, heterogeneous resources increase popularity and homogeneous resources promote knowledge creation.

Two contesting theories, namely structural holes theory and network closure theory (Burt 1992; Coleman 1988), explain the rise of heterogeneous and homogeneous resources respectively. A structural hole refers to an "empty space" which exists when a vertex or an object provides the only connection between two or more vertexes or objects in the social network. The proponents of structural holes theory believe the vertex at such a position has access to a greater variety of information, which brings about heterogeneity in resources (Burt 1992). On the other hand, network closure theory argues that a closed network can cultivate coherent beliefs and collective actions, which foster resource homogeneity (Gargiulo and Benassi 2000).

Applying these two theoretical contentions onto the OSS context, we posit an overarching proposition that the social network structure (and therefore OSS project connectedness) plays a contingent role in facilitating OSS project success. Considering the two forms of OSS project success (popularity and knowledge creation), we propose:

1. OSS projects with more structural holes can enjoy greater popularity; however,
2. OSS projects with fewer structural holes can achieve better knowledge creation.

We further develop the proposition by recognizing that while the network structure prioritizes the position of the OSS project for accessing resources like contributors (Zaheer and Soda 2009), how far these network-related benefits can be harnessed to contribute to OSS project

success hinges upon the extent to which that OSS project can assimilate such resources (Daniel et al. 2013; Setia et al. 2012). Previous studies contend that an OSS project's maturity[5] (i.e., whether it is in a pre-beta, beta, or post-beta phase) could indicate the project's ability to harness abundant resources (Garriga et al. 2011). That means that the maturity of the OSS project could potentially moderate the relationship between OSS project connectedness (as manifested by structural holes) and OSS project performance (as manifested by popularity and knowledge creation).

By empirically analyzing a large longitudinal dataset from Sourceforge.net, our study makes several contributions to OSS literature, of which two are presented here. First, our findings reveal that OSS interproject connectedness has a significant impact on OSS project success in terms of popularity and knowledge creation. This research adds to the few OSS studies that take a social network perspective to OSS development (Grewal et al. 2006; Singh 2010; Singh et al. 2011) by extending the understanding that one OSS project can gain popularity through gaining more structural holes while another OSS project situated in a cohesive network is more likely to have better knowledge creation. Second, we provide evidence that OSS project maturity helps attract more resources but the relationship is not straightforward. Only the OSS projects which progress from a very nascent stage to developmental stage and are saturated with abundant structural holes can benefit from project maturity for greater popularity. This nuanced finding regarding maturity fills in an important gap in the extant literature that has disproportionately esteemed the positive role of maturity in promoting OSS project success (Daniel et al. 2013; Setia et al. 2012).

[5] OSS project age is an adjacent measure of maturity. Assuming that organizations accumulate innovation capabilities at the same rate, older organizations should outperform the younger ones (Schoonhoven 2015). However, this assumption has been challenged by several studies because an organization's age may not be a reliable proxy for maturity in terms of innovation capabilities (Coad et al. 2016). We therefore believe that innovation maturity is a more appropriate measurement of OSS project maturity.

2 Related OSS Literature

OSS research attracts considerable attention due to its intriguing and counterintuitive model of innovation, in which large numbers of IT talents voluntarily contribute to the creation, maintenance, and support of a public good (Lerner and Tirole 2002). Most of the early studies investigated the individual's motivation to participate in or contribute to the OSS project. These works discussed various participatory motives, such as enjoyment, self-efficacy, need for competence, community reputation, learning opportunities, and social identity (Feller et al. 2008; Shah 2006). Extending these works, von Hippel and von Krogh (2003) proposed a model of innovation to explain individual motives in participating in the OSS innovation activities. These authors found that although innovators do not gain proprietary benefit from the OSS per se, the free revealing (of source codes) promotes innovation diffusion and eventually the diffusion of such innovation-related information benefits the innovators.

Another stream of OSS research focuses on the success factors of OSS projects (Daniel et al. 2013; Garriga et al. 2011; Stewart et al. 2006; Subramaniam et al. 2009). These works mainly examined the intrinsic characteristics of the OSS project. Several software-specific characteristics have been identified as contributing to OSS project success, such as the restrictiveness of an OSS license[6] (Subramaniam et al. 2009), software type (Daniel et al. 2013), OSS project team size (Garriga et al. 2011), and organizational sponsorship[7] (Stewart et al. 2006).

OSS development involves the orchestrated and collective action of contributors who are related through their interactions, thereby forming a network of relationships and ties (Hahn et al.

[6] A type of license that allows or prevents the source code, blueprint or design in OSS project to be used, modified and/or shared under defined terms and conditions.

[7] An OSS project may receive financial or non-financial support from companies, foundations, non-profit organizations, etc.

2008). Prior literature classified these contributors into two groups: the development group and the management group (Subramaniam et al. 2009). While the development group consists of individuals who mainly contribute to the software coding, the management group (also known as product administrators or leaders) consists of individuals who create the OSS project and make the decisions on version releases . The innovation-related resources (i.e., source codes, bug fixes, project administration structure, etc.) are shared across connected projects due to common contributors.

Although OSS projects are connected in nature, we found only three studies that had investigated the network structural characteristic of an OSS project (i.e., its connectedness with other projects) and discussed its impact on OSS project success: Grewal et al. (2006); Singh (2010); and Singh (2011). And even these studies were not complete. Grewal et al. (2006) assessed whether the centrality of an OSS project based on the affiliations of its developers in other projects could predict the OSS project's success. However, this study did not consider that the centrality of an OSS project depends on how the other OSS projects are connected. Singh (2010) investigated the impact of macro-level network attributes such as the clustering coefficient (i.e., the degree of clustering in a network) on the success of the OSS projects residing in such networks. However, that study did not consider project-level network attributes such as the connectedness of the projects, hence limiting the implications of the study for the management of an OSS project. Singh et al. (2011) employed the collaboration network of the OSS project developers to reflect the internal and external cohesion of an OSS project and unveiled an inverted U-shape relationship between external cohesion and OSS project performance [measured as the number of concurrent versioning system (CVS) commits, a proxy for the closure of modification requests]. However, solely measuring the OSS project success by

the number of CVS commits is controversial because a large number of CVS commits may also imply poor software quality (Bird et al. 2009).

To fill these gaps, in our study we do the following: (1) analyze an OSS project's connectedness instead of the connectedness of the whole network; and (2) measure the OSS project success via two forms, i.e. popularity and knowledge creation. Our findings can inform OSS projects on how to strategically position themselves to achieve success.

Also, there is room for improvement of the theoretical foundations of the aforementioned three exceptional works (Grewal et al. 2006; Singh 2010; Singh 2011). Specifically, structural social capital only accounts for how the network structure affects the variance of information access but not the innovation outcome (Burt 1992). We attempt to further theorize the role of interproject connectedness in OSS project success. We argue that the configuration of the network structure not only affects information access but also, as a consequence, polarizes the nature of the accessible resources, i.e., heterogeneity vs. homogeneity (Nerkar and Paruchuri 2005; Ahuja 2000).

Besides resource accessibility, how effectively such resources can be harnessed should also significantly influence an OSS project's success (Daniel et al. 2013; Setia et al. 2012). Previous literature employed OSS project maturity as a proxy indicator of the capability of harnessing the resources. More specifically, mature projects with better project governance can effectively raise the productivity of the OSS development teams (Setia et al. 2012). Compared to projects at a nascent stage, mature projects have established team cognition and shared understanding, which reduce misunderstanding and disagreement in the course of OSS development (He et al. 2007). The advanced code management in a mature project is also helpful to internalize the knowledge and information collected from the other projects (Daniel et al. 2013; Setia et al. 2012).

Essentially, the maturity of the OSS project facilitates better use of various resources including contributors, their knowledge, experiences, and ideas. Thus, OSS project maturity should moderate the relationship between the interproject connectedness and project success. We will explain our hypotheses in detail in the next section.

3 Hypotheses Development

Within the OSS community, an OSS project's connectedness to other projects via common contributors defines its ego network. Visually, an OSS project's ego network positions it as the central vertex (ego) and the neighboring vertexes are the other OSS projects with ties to the ego (Everett and Borgatti 2005). Thus, whether and to what extent an OSS project has access to resources depends on its position in the network. To this end, social network analysis reveals the relationship between the structural position of a network vertex [which in the context of this study is the OSS project] and its access to resources. Several studies (Ahuja 2000; Austin 2003; Balkundi et al. 2007; Beckman and Haunschild 2002; Harrison and Klein 2007) discuss structural hole theory. In the OSS context, we ask whether the presence of more structural holes in an interproject network is beneficial to OSS project success.

Applying structural hole theory in the OSS context, we can infer that the OSS projects which are connected to other projects via contributors with non-redundant (non-overlapping) external network ties have access to a greater variety of resources (Austin 2003; Beckman and Haunschild 2002; Harrison and Klein 2007). This conjecture agrees with the thesis that socioeconomic opportunities increase with the number of structural holes in an ego network due to increased access to diversified information (Eagle et al. 2010). Conversely, in the absence of structural holes, vertexes in an ego network are less likely to generate new ideas (Balkundi et al. 2007). When contributors draw from different pools of resources, they are more likely to have

diverse viewpoints and opinions and could, therefore, deliver more creative products than those who draw from the same pool of resources (Harrison and Klein 2007; Jackson et al. 1995).

On the other hand, Podolny and Baron (1997) argued that "a cohesive network [a network with few structural holes] conveys a clear normative order within which the individual [OSS project] can optimize performance, whereas a diverse . . . network [network with many structural holes] exposes the individual [OSS project] to conflicting preferences and allegiances within which it is much harder to optimize" (p. 676). In a network with many structural holes, organizations must reconcile opposing views, which could reduce innovation performance (Van Knippenberg and Schippers 2007). In other words, an organization (i.e., an OSS project in our case) with many structural holes faces potential problems (Ahuja 2000) – such as coordination difficulty (Balkundi et al. 2007) and decreased production – even though conflict may contribute to more complete and careful analysis of the task at hand.

Conversely, the connected vertexes with few structural holes may benefit from the shared resources and beliefs about project priorities and how the work should be carried about. Tan et al. (2007) studied OSS developers' ego networks and found that OSS developers at brokerage positions may not benefit more than the rest of the community because they incur the cost of sharing and relating knowledge across heterogeneous projects. This empirical evidence supports arguments in favor of a cohesive network structure (i.e., few structural holes).

To reconcile these views, Ahuja (2000) evaluated competing hypotheses on the consequences of the number of structural holes on an organization's innovation performance. Ahuja observed that for an inter-organizational network that is focused on collaboration, cohesive networks (i.e., those with few structural holes) are likely to be beneficial because they foster the development of the fine-grained information and crucial resource exchange. However,

organizations that rely on diverse resources are likely to benefit from many structural holes. The two seemingly contradictory viewpoints could be reconciled by considering multiple dimensions of OSS project success. OSS project success cannot be evaluated using a single criterion but must be considered in light of multiple dimensions representing various stakeholders. There are also problems with some currently used measures of project success. For instance, the number of downloads is a widely-adopted index of popularity, but this index may be biased towards particular types of projects[8] (Crowston et al. 2006). Likewise, the number of CVS commits is used as a proxy of developers' vitality may also indicate poor software quality (Crowston et al. 2006).

Subramaniam et al. (2009) employed a multidimensional construct (user interest, developer interest and program activitiy) to represent OSS project success. They found the same antecedents to have different effects on different aspects of OSS project success. Peng et al. (2013) used software downloads and code released to measure OSS project success. In this study, we consider OSS project success based on two indicators: *popularity* and *knowledge creation*. The former reflects the interest in the OSS project by the users at large, which SourceForge.net bases on the number of OSS downloads and the number of OSS project site and page visits (Setia et al. 2012). The latter reflects the development intensity, which SourceForge.net bases on the number of CVS commits, the frequency of the released files (project output), and the project administrators' activity levels (Crowston et al. 2006). Next, we will examine these two performance indicators and deduce how the configuration of interproject connectedness and OSS project maturity affect them.

[8] In most cases, OSS projects that are designed as end-user applications are downloaded more often than those that serve as fundamental systems, such as game engines or frameworks.

3.1 The Controversy Surrounding Network Structure

The OSS projects' ego networks are comprised of the common contributors' links and ties, which facilitate information and resource exchange. Since in our study 'common contributors' refers to common administrators and common developers, we consider OSS projects' ego networks based on common administrators and common developers. This approach echoes previous studies that categorized OSS project participants according to their various roles (Aberdour 2007; Crowston and Howison 2006; Jiang et al. 2019; Setia et al. 2012). As OSS project leaders, administrators play an essential role in setting up projects, communicating with the OSS community, and recruiting and managing developers (Heckman et al. 2007). Unlike developers, who contribute specialized technical knowledge, administrators are often generalists who must be familiar with the projects' overall development and have the ability to integrate specialized knowledge. They govern and motivate developers to achieve a common goal: innovation development (Chen and Dietrich 2009). Administrators with preexisting developer contacts (e.g., from managing other projects) are more likely to attract developers to a focal project (Hahn et al. 2008), which could suggest the importance of the administrators' connection to other projects. As the primary contributors of innovation, developers who participate in multiple projects provide two knowledge benefits to a focal project: resource sharing and knowledge spillover (Ahuja 2000; Grewal et al. 2006). Resource sharing allows the developers to integrate knowledge within projects, whereas knowledge spillover provides the project participants with information about failed and successful approaches, breakthroughs, and opportunities (Ahuja 2000).

The number of structural holes in an OSS project's ego network is based on the focal project's connections with other projects due to common contributors. An OSS project's ego

network with many structural holes has greater access to diversified information, which is an essential resource in information retrieval activities (Nerkar and Paruchuri 2005). Administrators perform several vital functions, including communicating information about their projects to the OSS community (Heckman et al. 2007). Increasing the number of structural holes in an OSS project's ego network can enhance project visibility (Shipilov 2009). The administrators will be able to understand the broader interests of the other projects' contributors and make strategic decisions to popularize particular OSS projects. For instance, the administrators could reprioritize tasks in the pipeline because certain features are popular in the other projects. Likewise, developers who participate in multiple projects have close contact with the diversified demands of various OSS projects' users. Consequently, these developers could consider and gratify these various demands when developing the focal OSS, which then attracts broader interest in the project. Accordingly, we present the following hypothesis:

Hypothesis 1: An OSS project with a greater number of structural holes in its ego networks has higher popularity.

An OSS project's ego network with many structural holes may face communication and coordination challenges. Examining the patenting frequency in the chemical industry, Ahuja (2000) observed that the networks with many structural holes exhibited decreased innovation output in terms of the number of patents filed. Also, in a study of workgroups in a global organization, Cummings and Cross (2003) observed that workgroups with many structural holes exhibited diminished performance efficiency, poor scheduling, and poor budget adherence. Although increasing an OSS project's popularity is a matter of generating a spectrum of ideas based on shared knowledge and learning from other projects' failures and successes, OSS project development involves more than possessing such knowledge. Administrators' and developers'

experiences working on other projects must be coordinated and integrated during software development (Tullio and Staples 2013). From the social categorization perspective in network closure theory, a cohesive network with connected OSS projects (i.e., fewer structural holes) has shared mental models related to how administrators and developers should work together; this, in turn, could facilitate innovation output (Ren et al. 2016; Van Knippenberg et al. 2004). Besides, organizations with few structural holes benefit from access to shared resources and knowledge spillovers, enabling the contributors to efficiently leverage their shared intellectual property to develop the OSS. For these reasons, we hypothesize as follows:

Hypothesis 2: An OSS project with a lesser number of structural holes in its ego networks has higher knowledge creation.

3.2 OSS Project's Maturity

How much the resources from the connected projects could contribute to an OSS project success hinges on the internalization capability of the focal project (Zahra and George 2002). It is recognized that new product teams face the dilemma of limited resources or capital, which restricts their capability to synergize the internal dynamics with external resources (Patel et al. 2015; Schoonhoven 2015). Teams at an early phase need to bear the costs of learning new rules and creating new roles (in the workgroup) as well as establishing social relationships among internal and external stakeholders (Gulati and Higgins 2003; Li et al. 2008). This restricts the growth of innovation, which results in the high likelihood of mortality of projects in their early phases of development. Consequently, participants in immature OSS projects, even those with advantageous resource access, might still not be able to appropriately utilize such resources to boost their innovation output. Conversely, established teams comprised of members with long-

standing relationships could outperform “fresh” teams (Harrison et al. 1998, 2002) because the former could more easily absorb external information into their innovation output.

Operationally, the OSS developmental stages consist of three levels of maturity: pre-mature phase (pre-beta), mature phase (beta), and post-mature phase (post-beta) (Daniel et al. 2013; Setia et al.2012). An OSS project’s development stage (and hence its maturity) determines whether its participants can effectively incorporate the resources accessed from the community into innovation activities (Daniel et al. 2013; Setia et al.2012). The established governance mechanisms, well-written codes, and stable collaboration structure of a mature OSS project is likely to be attractive to contributors and the better defined stages make the particiants’ contributions more effective for yielding innovation output (Setia et al. 2012).

In OSS projects with many structural holes, the maturity of the project affects participants’ ability to internalize the diversified resources into innovation outputs. Diversified resources could include knowledge of various user demands, inspirational ideas, and novel technologies from other OSS projects (Nerkar and Paruchuri 2005; Shipilov 2009). According to Jones (2006), mature organizations emphasize knowledge exploitation. Knowledge exploitation entails the effective application of resources by emphasizing the “refinement, routinization, production and elaboration of existing experience” (Holmqvist 2003, page 99). Relating this to our context, OSS projects residing in a less cohesive network, i.e. with more structural holes, can achieve higher popularity because more diverse user requirement or ideas can be accessed and incoporated. The mature projects are more experienced in both management and development, thereby more effectively and efficiently completing tasks to meet various demands. OSS projects residing in a more cohesive network, i.e. with fewer structural holes, can advance knowledge creation because of the shared mental models among contributors. Thus, the mature OSS projects

in cohesive a network ought to have more experiences in collaboration of development than those less mature ones, thereby faciliating knowledge creation. In summary, we hypothesize:

Hypothesis 3: The positive relationship between the number of structural holes and popularity will be stronger with the increase of maturity of OSS projects.

Hypothesis 4: The positive impact of a cohesive network on knowledge creation will be stronger for more mature OSS projects.

4 Research Methodology

4.1 Background and Ego Network

We conducted a longitudinal investigation using the dataset available on SourceForge.net. We collected the data twice in 18 months to separate the antecedents from their outcomes and to allow for a more extended observation period in order to determine the effect of brokerage positions on popularity and knowledge creation. The 18-month time frame is referred to as the OSS project's observation period in previous OSS literature (Crowston et al. 2008; Daniel et al. 2018; Ghosh 2006). Furthermore, we excluded OSS projects with the status "inactive" or "planning" because such projects would not have been released to the public or the participants would have had little opportunity to contribute to such projects.

With the dataset obtained, we constructed each OSS project's ego network to discover each project's position in the overarching network. Each OSS project was recorded as a vertex; projects were linked to one another through contributors across the various project categories[9]. Such a network is defined as an affiliation network in prior literature (Wasserman and Faust 1994). In our research setting, OSS projects are linked with one another if they have a common

[9] 18 main project categories were found on SourceForge at the time of data collection: development, games, Internet, scientific, system, education, desktop, communications, security, editors, multimedia, formats and protocols, database, office, printing, religion, and mobile apps.

contributor (such as an administrator or a developer). Hence, we constructed two affiliation networks – one based on common administrators and one based on common developers.

4.2 Dependent Variables

Previous studies attempted to evaluate OSS project performance using multiple dimensions (e.g., number of downloads, number of CVS commits, and size of developer team) (Crowston et al. 2006; Healy and Schussman 2003; Subramaniam et al. 2009). SourceForge.net employ composite indexes to rank each project[10]. We employed two composite indexes, namely *traffic intensity* and *development intensity*, to indicate the OSS project's popularity and knowledge creation respectively. Traffic intensity included downloading intensity (the extent of adoption among end-users), logo-hitting intensity (the number of visits to the project page), and page-view intensity (the number of visits within the project page). Development intensity was composed of CVS commits (the extent of contribution from contributors to the focal OSS project), the history of recently released files (the extent of the overall contributors' recent vitality), and administrators' login information (the extent of administrators' activities). The detailed equations for traffic intensity (TRF) and development intensity (DEV), following Sourceforge.net's equations, are presented below. The descriptions of the components of the equations are given in Table 1.

$$TRF_{it} = \left(\ln(P7DT_{it}+1)\Big/\ln(HIDT_t+1) + \ln(P7LT_{it}+1)\Big/\ln(HILT_t+1) + \ln(P7ST_{it}+1)\Big/\ln(HIST_t+1)\right)\Big/_3$$

$$DEV_{it} = \left(\ln(P7CT_{it}+1)\Big/\ln(HICT_t+1) + DALFR_{it}\Big/_{100} + DADML_{it}\Big/_{100}\right)\Big/_3$$

Table 1. Definitions of intensity components	
Components name	Description

[10] A Sourceforge blog reported the parameters used to rank the hosted OSS projects [https://sourceforge.net/blog/sourceforge-stats-demystified]. The formula is in Sourceforge documentation: https://web.archive.org/web/20070213152144/http://sourceforge.net/docman/display_doc.php?docid=14040&group_id=1#rankings. [last access: 2020 March]

TRF_{it}	Traffic intensity of project *i* at time *t*. This variable was defined by SourceForge, which included downloading, logo hitting, and site-hitting traffic.
$P7DT_{it}$	The total downloading counts of project *i* in the last 7 days since time *t*
$HIDT_t$	The most downloaded counts at time *t*
$P7LT_{it}$	The total logo hit counts of project *i* in the last 7 days since time *t*
$HILT_t$	The most logo hit counts at time *t*
$P7ST_{it}$	The total site hit counts of project *i* in the last 7 days since time *t*
$HIST_t$	The most site hit counts at time *t*
DEV_{it}	Development intensity of project *i* at time *j*. This variable was defined by SourceForge, which included CVS commits, history of most recent file released, and history of administrator logins.
$P7CT_{it}$	The total CVS commit counts of project i in the last 7 days since time *t*
$HICT_t$	The most CVS-committed counts at time *t*
$DALFR_{it}$	The absolute value of the difference between 100 and the days (maximally 100) of the latest file released since time *t*
$DADML_{it}$	The absolute value of the difference between 100 and the days (maximally 100) of last project administrator login since time *t*

Using the above equations, we computed each OSS project's traffic intensity and development intensity at t_1 and t_2. Then, we computed the traffic ratio (i.e. the traffic intensity at t_2 over the traffic intensity at t_1) for lag specification. Similarly we computed the development ratio. We used the traffic ratio and the development ratio as our dependent variable to investigate the incremental or decremental change. To avoid missing values resulting from denominators of zero (the intensity at t_1 may be zero), we added 1 to all values at t_1. Below are the equations:

$$traffic\_ratio_{it2} = {TRF_{it2}}/{(TRF_{it1} + 1)}$$

$$development\_ratio_{it2} = {DEV_{it2}}/{(DEV_{it1} + 1)}$$

4.3 Predictors and Control Variables

We adopted Burt's constraint index (Burt 1992) The value is a reverse indicator of the number of structural holes. In other words, for any focal OSS project, a high constraint index denotes few structural holes. The equation for Burt's constraint index is presented below,

$$C_i = \sum_j \left(P_{ij} + \sum_q P_{iq} P_{qj} \right)^2 , i \neq j \neq q$$

where C_i is Burt's constraint index of vertex *i* (OSS project *i*) and P_{ij} is the proportion of OSS project *i*'s resources spent on its contact, *j*.

Suppose vertex *i* has four direct linkages and the strength of its linkage to vertex *j* and the other vertexes are 2, 1, 1, and 1, respectively, then the value of P_{ij} is 2/5. In our case, the strength of the linkage between two projects is measured by the number of common contributors (administrators or developers). We computed Burt's constraint indexes for each OSS project at T_1, denoted by $admin\_c_{it1}$ (administrator-affiliated network) and $developer\_c_{it1}$ (developer-affiliated network), respectively. We then created two variables, $\boldsymbol{admin\_sh_{it1}}$ and $\boldsymbol{developer\_sh_{it1}}$, which are computed as 1-$admin\_c_{it1}$ and 1-$developer\_c_{it1}$, to *represent* ***the number of structural holes in the administrator-affiliated network and developer-affiliated network respectively*** (Tortoriello 2015).

We determined OSS projects' maturity by referring to their developmental phases (i.e., Pre-alpha, Alpha, Beta, Production, and Mature in Sourceforge). Referring to previous studies (Daniel et al. 2013; Setia et al. 2012), we categorized the developmental stages into three phases, namely Pre-beta (including Pre-alpha and Alpha), Beta, and Post-beta (Production and Mature). The ***maturity*** of an OSS project *i* at T_1 was denoted by a categorical variable, $\boldsymbol{Dev\_stage_{it1}}$.

Besides the key predictors, we considered several covariates to control for variance across the affiliation networks and the OSS projects' characteristics. We grouped the control variables into seven main categories: evenness of work distribution, IT-enabled administration, knowledge control, programming language popularity, team-based characteristics, project license, and project category.

Evenness of work distribution: In previous literature, researchers have argued that the uniformity of work distribution among the contributors would have an impact on project success

(Woolley et al. 2010). To measure the extent of work distribution, we employed the idea of the Gini coefficient and constructed the generalized inequality indicator for our work (Kuk 2006; Thon 1982), denoted as $InEqual_i$. Instead of measuring the work distribution solely by considering the contributor's commitment to the OSS project, we acknowledged contributions more comprehensively. In other words, the contributions, such as a bug report, feature improvement suggestions, and debugging solutions, are all included. The equation is as follow:

$$InEqual_i = \frac{\sum_{j=1}^{n}\left((2m - n + 1)y_m\right)}{n^2\bar{y}}$$

where n is the number of contributors to the OSS project i, y_m is the count of developer m contributions, and $\bar{y}$ is the average number of contributions expected per contributor.

Notably, the value of y_m, m=1 to n, should be indexed in non-descending order (i.e., $y_m <= y_{m+1}$). The value of this indicator ranges from 0 to 1: all contributors contributing equally make this indicator approach 0, while only a few contributors contributing to the focal OSS project makes it approach 1. We employed the aforementioned formula to calculate a variable representing the evenness of work distribution in each OSS project *i at* T_1, which is denoted by $InEqual_{it1}$.

IT-enabled administration: SourceForge.net provided several IT artifacts for various purposes, including communication and assistance. In prior literature, researchers have argued that the adoption of IT communication tools can not only increase the efficiency of project teamwork but can also promote quality assurance (Jurison 1999). Accordingly, we checked whether the sampled OSS projects used the available IT tools. In doing so, we included binary variables: use_mail_{it1} to denote whether e-mail notification was enabled in the OSS project i at T_1, use_pm_{it1} to denote whether personal messaging was enabled for the OSS project i at T_1, and use_forum_{it1} to denote whether forum was enabled in the OSS project.

Knowledge control: In addition to the IT tools to support the contributors, several IT artifacts are available to the public from SourceForge.net, from which the end-users can obtain their desired knowledge about the focal OSS project. For instance, (a) users can receive updates on their OSS projects when the project news function (use_news_i) is enabled; (b) the software screenshots ($use\_screenshots_i$) can provide the users with the first impressions of the software, which could be extremely important for software that relies on graphics (e.g., games or multimedia software); and (c) the project wiki (use_wiki_i) provides tutorials or advanced knowledge for end-users and those who may be interested in engaging in further development. All three IT artifacts (use_news_{it1}, $use\_screenshots_{it1}$, and use_wiki_{it1}) constitute the knowledge controls for the project i at T_1.

Programming-Language Popularity: Extant literature demonstrates that programming languages and project types are important considerations in an OSS project (Zhu and Zhou 2012). For example, more developers may have some knowledge in popular programming languages such as Java or PHP than in less popular languages. The data set used in this research comprised 79 programming languages. The categorical variable $Lang_i$ was used to denote the programming language for OSS project i. Also, we referred to the TIOBE index to control for each language's popularity in OSS project i at T_1, ($Lang\_Pop_{it1}$) because more people are attracted to OSS projects that are written in more popular programming languages. The TIOBE index is widely recognized for measuring the popularity of programming languages (Paulson 2007); the higher values refer to higher popularity.

Team-Based Characteristics: Each OSS project is developed and maintained by a group of participants. Therefore, team-based characteristics may also influence the OSS project's performance (Singh et al. 2011). For instance, the tenure of an OSS project team served as an

important representation of the extent of collaborative experiences and relationships (Hahn et al. 2008; Tan et al. 2007); the network size determined the extent to which miscellaneous information (other than work/project-related information) could flow into the team knowledge base, which could, in turn, affect the innovation output (Hahn et al. 2008; Tan et al. 2007). To this end, we employed two variables, $Team\_Tenure_{it1}$ and Net_Size_{it1}, to indicate the team tenure and network size, respectively, of OSS project *i* at T_1. The former was measured as the mean value of team member tenure (by years), and the latter was measured as the number of participants affiliated with a particular OSS project.

OSS License: Various OSS licenses limit copyright: licenses range from permissive licenses (e.g., MIT or BSD) to protective licenses (e.g., GPL) (Wen et al. 2013). Restrictions on the use and distribution of the software may affect the diffusion of innovation (e.g., code distribution) (Wen et al. 2013). Therefore, we created a categorical control variable, $License_i$, indicating the type of license used in a particular OSS project *i*.

Project Category: Our sample included 18 categories of OSS projects. Previous researchers claimed that the nature of OSS projects also affected the innovation output's evolution. For instance, the projects creating applications attracted more end-users than the projects creating an OSS framework (Dong et al. 2019). Therefore, we created a categorical control variable, $Category_i$, indicating the category of OSS project *i*.

5 Data Analysis

5.1 Main Results

The unit of analysis is the OSS-project. Considering that the dependent variable is a fractional value (i.e., the value between 0 and 1), the generalized linear model (GLM) with a canonical logit link in a binomial family was employed (Wooldridge 2010). We constructed two regression

models to depict two types of intensity-change ratio: traffic intensity-change ratio and development intensity-change ratio. The descriptive data analysis and the description of each variable are given in Table 2. The correlation table is displayed in Table 3, in which all the coefficients are less than 0.6. We used a variance inflation factor (VIF) to test for multicollinearity. According to the rule of thumb, a VIF value that exceeds five is considered evidence of multicollinearity, and a VIF value that exceeds ten is regarded as serious evidence of multicollinearity. No multicollinearity concerns were found in our models.

Table 2. Descriptive Statistics

	Projects with common administrators (13305 observations)				Projects with common developers (12898 observations)			
Continuous Variables	Mean.	S.D.	Min.	Max.	Mean.	S.D.	Min.	Max.
Traffic intensity-change ratio of OSS project i at t_2 (*traffic_ratio*$_{it2}$)	0.093	0.077	0	0.473	0.094	0.079	0	0.473
Development intensity-change ratio of OSS project i at t_2 (*development_ratio*$_{it2}$)	0.329	0.118	0	0.737	0.332	0.116	0	0.737
Number of structural holes in common administrator network of project i at t_1 (*admin_sh*$_{it1}$)	0.280	0.272	0	0.937	--	--	--	--
Number of structural holes in common developer network of project i at t_1 (*developer_sh*$_{it1}$)	--	--	--	--	0.285	0.276	0	0.975
Generalized inequality indicator of work distribution between all contributors at t_1 (*InEqul*$_{it1}$)	0.437	0.143	0	0.954	0.442	0.145	0	0.956
Popularity of programming language (*Lang_Pop*$_{it1}$)	0.117	0.068	0.0001	0.205	0.117	0.067	0.0001	0.205
Team tenure (*Team_Tenure*$_{it1}$), in years	5.871	2.330	0.003	9.6	5.961	2.330	0.003	9.6
Team network size (*Net_Size*$_{it1}$)	3.152	7.336	1	430	3.289	7.678	1	430
Categorical Variables								
Developmental Stages (*Dev_stage*$_{it1}$)								
Dev_stage$_{it1}$*=0 (Pre-beta phase)*	7,915				7,542			
Dev_stage $_{it1}$*=1 (Beta phase)*	2,088				2,061			
Dev_stage $_{it1}$*=2 (Post-beta phase)*	3,302				3,295			
Whether project *i* enables email function (*use_mail*$_i$)								
use_mail$_i$*=0 (Disabled email function)*	2,026				1,935			
use_mail$_i$*=1 (Enabled email function)*	11,279				10,963			
Whether project *i* enables internal messages function (*use_pm*$_i$)								
use_pm$_i$*=0 (Disabled internal message function)*	2,546				2,479			
use_pm$_i$*=1 (Enabled internal message function)*	10,759				10,419			
Whether project *i* enables forum function (*use_forum*$_i$)								
use_forum$_i$*=0 (Disabled forum function)*	2,766				2,692			

use_forum$_i$=1 (Enabled forum function)	10,539	10,206
Whether project *i* uses newsletters (*use_news$_i$*)		
use_news$_i$=0 (Disabled newsletters function)	926	901
use_news$_i$=1 (Enabled newsletters function)	12,379	11,997
Whether project *i* uses screenshots (*use_screenshots$_i$*)		
use_screenshots$_i$=0 (Disabled screenshots function)	911	859
use_screenshots$_i$=1 (Enabled screenshots function)	12,394	12,039
Whether project *i* uses project wiki (*use_wiki$_i$*)		
use_wiki$_i$=0 (Disabled project wiki)	12,162	11,775
use_wiki$_i$=1 (Enabled project wiki)	1,143	1,123

Programming languages (Lang$_i$): 79 programming languages were considered (Java, PHP, Python, C#, C++, C, Visual Basic, ASP.NET, Perl, Assembly, Lisp, XSL (XSLT/XPath/XSL-FO), Visual Basic .NET, JavaScript, Unix Shell, Fortran, S/R, ActionScript, AppleScript, BASIC, Pascal, Tcl, AspectJ, Prolog, Objective C, Ruby, Object Pascal, Euphoria, Standard ML, Oberon, Smalltalk, PL/SQL, MATLAB, OCaml (Objective Caml), Free Pascal, ASP, Logo, Delphi/Kylix, APL, IDL, JSP, D, Erlang, Lazarus, XBase/Clipper, VBScript, Visual FoxPro, Emacs-Lisp, MUMPS, Flex, Scheme, Ada, Groovy, COBOL, Lua, Forth, Mathematica, Eiffel, REALbasic, XBasic, haXe, Haskell, Curl, AWK, Kaya, Visual Basic for Applications (VBA), Modula, Clean, LPC, Rexx, Common Lisp, LabVIEW, VHDL/Verilog, PROGRESS, Pike, Cold Fusion, Boo, Oz, other).

OSS licenses (License$_i$): 56 OSS licenses were considered (apache, gpl, lgpl, apache2, python, bsd, website, artistic, zlib, publicdomain, mit, public, ibmcpl, nethack, educom, afl, apsl, eclipselicense, wxwindows, mpl, cddl, psfl, sleepycat, ibm, osl, mpl11, qpl, zope, adaptive, none, sissl, php-license, fair, gplv3, w3c, boostlicense, cpal, rpl15, ncsa, historical, php, attribut, agpl, iosl, sunpublic, real, opengroup, osi, ms-rl, datagrid, eiffel, jabber, eiffel2, rscpl, rpl, other).

OSS project categories (Category$_i$): 18 categories were considered (development, games, Internet, scientific, system, education, desktop, communications, security, editors, multimedia, formats and protocols, database, office, printing, religion, mobile apps, other).

Table 3. Correlation Matrices and VIFs

Projects with common administrators												
	$admin\_sh_{it1}$	$InEqul_{it1}$	$Lang\_Pop_{it1}$	use_mail_i	use_pm_i	use_forum_i	use_news_i	$use\_screenshots_i$	use_wiki_i	$Team\_Tenure_{it1}$	Net_Size_{it1}	*VIFs*
$admin\_sh_{it1}$	1											1.04
$InEqul_{it1}$	0.146	1										1.48
$Lang\_Pop_{it1}$	0.008	0.001	1									1.00
use_mail_i	-0.023	0.030	0.023	1								1.44
use_pm_i	-0.074	-0.062	-0.003	0.505	1							1.81
use_forum_i	-0.105	-0.089	0.0002	0.434	0.552	1						1.61
use_news_i	-0.066	-0.021	-0.001	0.316	0.396	0.378	1					1.27
$use\_screenshots_i$	-0.036	-0.043	-0.008	0.243	0.369	0.216	0.216	1				1.20
use_wiki_i	-0.004	0.015	-0.026	0.049	0.057	0.066	0.022	0.022	1			1.06
$Team\_Tenure_{it1}$	0.134	0.0387	0.030	-0.093	-0.122	-0.142	-0.135	-0.012	-0.179	1		1.08
Net_Size_{it1}	0.144	0.233	0.014	0.012	-0.107	-0.153	-0.042	-0.087	0.037	-0.013	1	1.10
Projects with common developers												
	$developer\_sh_{it1}$	$InEqul_{it1}$	$Lang\_Pop_{it1}$	use_mail_i	use_pm_i	use_forum_i	use_news_i	$use\_screenshots_i$	use_wiki_i	$Team\_Tenure_{it1}$	Net_Size_{it1}	
$developer\_sh_{it1}$	1											1.07
$InEqul_{it1}$	0.196	1										1.48
$Lang\_Pop_{it1}$	0.005	-0.001	1									1.00
use_mail_i	-0.021	0.033	0.028	1								1.42
use_pm_i	-0.076	-0.059	-0.001	0.495	1							1.80
use_forum_i	-0.076	-0.0940	-0.0002	0.431	0.547	1						1.61
use_news_i	-0.065	-0.022	-0.005	0.299	0.403	0.378	1					1.27
$use\_screenshots_i$	-0.035	-0.034	-0.012	0.236	0.360	0.274	0.211	1				1.19
use_wiki_i	0.002	0.018	-0.024	0.050	0.055	0.045	0.066	0.017	1			1.06
$Team\_Tenure_{it1}$	0.158	0.045	0.028	-0.092	-0.122	-0.145	-0.14	-0.01	-0.182	1		1.09
Net_Size_{it1}	0.194	0.256	0.012	0.017	-0.113	-0.151	-0.045	-0.078	0.046	-0.006	1	1.13

The findings depicting the change in traffic intensity consist of seven models and are summarized in Table 4[11]. Model 1 is the base model with $traffic\_ratio_{it2}$ as the dependent variable, in which only the control variables are included. In Model 2, the number of structural holes computed from the affiliated network with common administrators was entered to test Hypothesis 1a. The significantly positive coefficient of $admin\_sh_{it1}$ supports **Hypothesis 1a**. In Model 3, the number of structural holes ($developer\_sh_{it1}$) in the network constructed with common developers was positively significant. Hence, **Hypothesis 1b** is also supported.

To test the moderating effect in Hypothesis 3, the project maturity, Dev_stage_{it1}, and the interaction terms – $admin\_sh_{it1}XDev\_stage_{it1}$ and $developer\_sh_{it1}XDev\_stage_{it1}$ – are entered into Model 4 through Model 7. We first tested the moderating effect between project maturity and the number of structural holes computed from the affiliated network with common administrators in Models 4 and 5. In Model 4, we set the OSS projects at a pre-beta phase as the investigation base. The positive relationship between the number of structural holes and the traffic intensity is strengthened in beta projects but not in post-beta projects, partially supporting our hypothesis. In Model 5, we changed the investigation base from the pre-beta phase to the beta phase to check the moderating effect of pre-beta and post-beta projects. The estimated coefficient is negatively significant in both phases, again partially supporting our hypothesis. Similarly, we conducted the same empirical testing for the affiliated network with common developers in Model 6 and Model 7 and obtained similar results. Therefore, we could conclude that **Hypothesis 3** is partially supported.

[11] We did not include the $admin\_sh_{it1}$ and $developer\_sh_{it1}$ in the same model because of multicollinearity. These two independent variables are highly correlated.

Table 4. Results with Change in Traffic Intensity as the Dependent Variable							
DV	Traffic intensity-change ratio (*traffic_ratio*$_{it2}$)						
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
admin_sh$_{it1}$	--	0.114*** (0.031)	--	0.077* (0.039)	0.287*** (0.063)	--	--
developer_sh$_{it1}$	--	--	0.149*** (0.035)	--	--	0.077+ (0.041)	0.282*** (0.063)
Dev_stage$_{it1}$*< Beta*	--	--	--	--	-0.26*** (0.032)	--	-0.261*** (0.032)
Dev_stage$_{it1}$*= Beta*	--	--	--	0.26*** (0.032)	--	0.261*** (0.032)	--
Dev_stage$_{it1}$*> Beta*	--	--	--	0.496*** (0.027)	0.236*** (0.032)	0.468*** (0.027)	0.207*** (0.033)
admin_sh$_{it1}$*XDev_stage*$_{it1}$*< Beta*	--	--	--	--	-0.21** (0.072)	--	--
admin_sh$_{it1}$*XDev_stage*$_{it1}$*= Beta*	--	--	--	0.21** (0.072)	--	--	--
admin_sh$_{it1}$*XDev_stage*$_{it1}$*> Beta*	--	--	--	0.013 (0.061)	-0.198** (0.076)	--	--
developer_sh$_{it1}$*XDev_stage*$_{it1}$*< Beta*	--	--	--	--	--	--	-0.205** (0.072)
developer_sh$_{it1}$*XDev_stage*$_{it1}$*= Beta*	--	--	--	--	--	0.205** (0.072)	--
developer_sh$_{it1}$*XDev_stage*$_{it1}$*> Beta*	--	--	--	--	--	0.071 (0.062)	-0.134+ (0.076)
InEqul$_{it1}$	0.932*** (0.082)	0.796*** (0.09)	0.868*** (0.089)	0.644*** (0.076)	0.644*** (0.076)	0.707*** (0.077)	0.707*** (0.077)
use_mail$_{it1}$	0.158*** (0.026)	0.152*** (0.027)	0.149*** (0.027)	0.143*** (0.026)	0.143*** (0.026)	0.144*** (0.027)	0.144*** (0.027)
use_pm$_{it1}$	-0.269*** (0.024)	-0.281*** (0.025)	-0.263*** (0.025)	-0.244*** (0.024)	-0.244*** (0.024)	-0.229*** (0.024)	-0.229*** (0.024)
use_forum$_{it1}$	-0.186*** (0.023)	-0.178*** (0.026)	-0.176*** (0.024)	-0.167*** (0.024)	-0.167*** (0.024)	-0.164*** (0.023)	-0.164*** (0.023)
use_news$_{it1}$	0.119*** (0.031)	0.127*** (0.034)	0.107** (0.034)	0.148*** (0.033)	0.148*** (0.033)	0.124*** (0.033)	0.124*** (0.033)
use_screenshots$_{it1}$	-0.134*** (0.029)	-0.118*** (0.032)	-0.116*** (0.031)	-0.132*** (0.031)	-0.132*** (0.031)	-0.124*** (0.031)	-0.124*** (0.031)
use_wiki$_{it1}$	0.146*** (0.026)	0.134*** (0.029)	0.138*** (0.028)	0.194*** (0.028)	0.194*** (0.028)	0.196*** (0.027)	0.196*** (0.027)
Lang_Pop$_{it1}$	-206.83** (65.338)	-199.277* (85.514)	-206.954** (76.792)	-170.758* (73.066)	-170.758* (73.066)	-182.183** (67.321)	-182.183** (67.321)
Team_Tenure$_{it1}$	0.021*** (0.003)	0.02*** (0.004)	0.021*** (0.004)	-0.012** (0.004)	-0.012** (0.004)	-0.011* (0.004)	-0.011* (0.004)
Net_Size$_{it1}$	0.015*** (0.004)	0.015*** (0.005)	0.015*** (0.004)	0.011** (0.004)	0.011** (0.004)	0.011** (0.004)	0.011** (0.004)
constant	-3.319*** (0.191)	-3.28*** (0.193)	-3.383*** (0.199)	-3.125*** (0.187)	-2.864*** (0.189)	-3.211*** (0.193)	-2.95*** (0.195)
Lang$_i$*, License*$_i$*, Category*$_i$*: Included but not reported*							
Log-pseudo likelihood	-3337.346	-3018.902	-2949.184	-2998.722	-2998.722	-2930.249	-2930.249
+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*							

In Table 5, the dependent variable was replaced with $development\_ratio_{it2}$, which is the development intensity-change ratio, to test Hypotheses 2a, 2b, and 4. The estimated coefficients are shown in Model 8 to Model 14. Model 8 is the base model, which only includes the control variables. The number of structural holes from the administrator-affiliated network was entered in Model 9. Model 9 shows that the $admin\_sh_{it1}$ has a significant negative effect on the change in development intensity. Therefore, **Hypothesis 2a** is supported. In Model 10, the estimated coefficient of $developer\_sh_{it1}$ is found to be significantly negative as well, which supports **Hypothesis 2b**.

The project maturity, Dev_stage_{it1}, and the interaction terms, $admin\_sh_{it1}XDev\_stage_{it1}$ and $developer\_sh_{it1}XDev\_stage_{it1}$, are entered from Model 11 to Model 14. Interestingly, the estimated coefficients of the interaction terms are not significant in either the administrator- or developer-affiliated networks. Therefore, we conclude that **Hypothesis 4** is not supported.

Table 5. Results with Change in Development Intensity as the Dependent Variable							
DV	Development intensity change ratio (*development_ratio$_{it2}$*)						
	Model 8	Model 9	Model 10	Model 11	Model 12	Model 13	Model 14
admin_sh$_{it1}$	--	-0.175*** (0.018)	--	-0.170*** (0.025)	-0.182*** (0.038)	--	--
developer_sh$_{it1}$	--	--	-0.125*** (0.018)	--	--	-0.142*** (0.025)	-0.100** (0.037)
Dev_stage$_{it1}$< Beta	--	--	--	--	-0.176*** (0.017)	--	-0.166*** (0.017)
Dev_stage$_{it1}$= Beta	--	--	--	0.176*** (0.017)	--	0.166*** (0.017)	--
Dev_stage$_{it1}$> Beta	--	--	--	0.176*** (0.015)	0.0003 (0.017)	0.168*** (0.015)	0.002 (0.017)
admin_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	0.011 (0.045)	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	-0.011 (0.045)	--	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	-0.001 (0.038)	0.01 (0.047)	--	--
developer_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	--	--	-0.041 (0.044)
developer_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	--	--	0.041 (0.044)	--
developer_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	--	--	0.029 (0.038)	-0.013 (0.046)
InEqul$_{it1}$	-0.128*** (0.031)	-0.117*** (0.032)	-0.098** (0.032)	-0.15*** (0.032)	-0.15*** (0.032)	-0.136*** (0.033)	-0.136*** (0.033)
use_mail$_{it1}$	-0.036* (0.015)	-0.036* (0.016)	-0.038* (0.015)	-0.037* (0.016)	-0.037* (0.016)	-0.038* (0.015)	-0.038* (0.015)
use_pm$_{it1}$	-0.023 (0.015)	-0.028+ (0.016)	-0.019 (0.016)	-0.017 (0.016)	-0.017 (0.016)	-0.01 (0.016)	-0.01 (0.016)
use_forum$_{it1}$	0.003 (0.014)	-0.00002 (0.015)	0.002 (0.015)	0.003 (0.015)	0.003 (0.015)	0.005 (0.015)	0.005 (0.015)
use_news$_{it1}$	0.01 (0.02)	0.009 (0.021)	0.016 (0.022)	0.021 (0.021)	0.021 (0.021)	0.027 (0.021)	0.027 (0.021)
use_screenshots$_{it1}$	0.006 (0.02)	0.002 (0.021)	0.006 (0.021)	-0.005 (0.021)	-0.005 (0.021)	0.002 (0.021)	0.002 (0.021)
use_wiki$_{it1}$	-0.05** (0.018)	-0.055** (0.019)	-0.042* (0.019)	-0.031 (0.019)	-0.031 (0.019)	-0.018 (0.019)	-0.018 (0.019)
Lang_Pop$_{it1}$	-382.364*** (24.136)	-385.485*** (25.36)	-390.679*** (24.621)	-362.686*** (28.841)	-362.686*** (28.841)	-370.934*** (27.374)	-370.934*** (27.375)
Team_Tenure$_{it1}$	0.012*** (0.002)	0.012*** (0.002)	0.013*** (0.002)	-0.001 (0.002)	-0.001 (0.002)	0.0002 (0.002)	0.0002 (0.002)
Net_Size$_{it1}$	-0.005*** (0.001)	-0.004*** (0.001)	-0.004*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)
constant	-0.284+ (0.171)	-0.223 (0.168)	-0.238 (0.169)	-0.18 (0.171)	-0.004 (0.172)	-0.187 (0.173)	-0.021 (0.174)
Lang$_i$, License$_i$, Category$_i$: Included but not reported							
Log-pseudo likelihood	-6255.396	-5741.350	-5574.505	-5733.573	-5733.573	-5566.659	-5566.659
+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*							

Overall, the results indicate that OSS projects with a greater number of structural holes enjoy higher popularity for both administrator-affiliated networks (coefficient [$admin\_sh_{it1}$] = 0.114, p-value < 0.001) and developer-affiliated networks (coefficient [$developer\_sh_{it1}$] = 0.149, p-value < 0.001). However, OSS projects with a greater number of structural holes also suffer from less knowledge creation for both administrator-affiliated networks (coefficient [$admin\_sh_{it1}$] = −0.175, p-value < 0.001) and developer-affiliated networks (coefficient [$developer\_sh_{it1}$] = −0.170, p-value < 0.001). This finding suggests the OSS project that is tightly connected with the other projects enjoys more intensive software development.

The OSS project's maturity is found to strengthen the positive relationship between the number of structural holes and traffic intensity in administrator- and developer-affiliated networks. However, such a positive moderating effect can only be observed between pre-beta and beta phases. There is no significant difference in the moderating effect between pre-beta and post-beta phases. This interesting finding is not counterintuitive. In the software release life cycle, the beta version is used to gather feedback on bugs or possible new features (MacCormack 2001). Therefore, more information ought to intensely flow into the OSS projects during the beta phase through their connected projects. Finally, project maturity was not found to strengthen the negative relationship between the number of structural holes and development intensity.

To further validate our empirical findings, we plotted the estimations of our primary predictors in Figure 1 below. Figures 1(a) and 1(b) show the results estimated in Table 4. The gradient of the red line, representing the OSS projects at beta phase, is steeper than those of the other two lines. The blue line (pre-beta phase) is almost parallel to the green line (post-beta phase), implying the positive moderation effect between the beta phase and the two other phases.

In addition, all three lines are almost parallel to each other in Figures 1(c) and 1(d), implying the insignificant moderation effect.

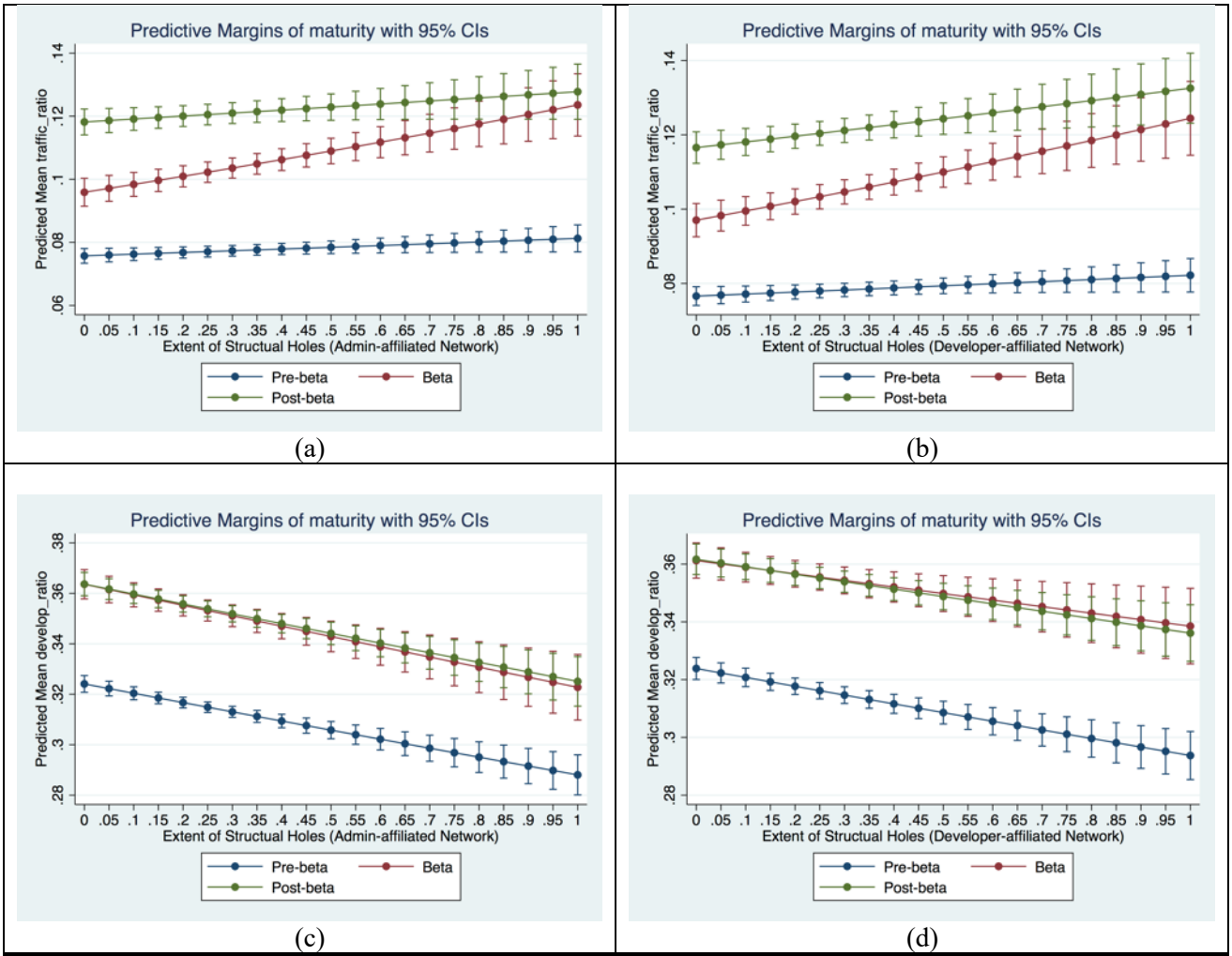


Figure 1. The Interaction Plots

5.2 Post Hoc Investigations

We further conducted the post-hoc investigations by considering the case that a developer may be involved in multiple projects across different project categories. We adopted the Blau's heterogeneity index (1977) – which has been employed in previous studies (Knight et al. 1999) – to represent the categorical heterogeneity, which is mathematically expressed below.

$$\text{Blau Index} = 1 - \sum_{k=1}^{s} p_k^2$$

where p is the proportion of connected projects in category k, and s is the number of project categories (s is 18 in our context).

A higher value of the Blau index implies a greater extent of resource heterogeneity. For example, suppose that one OSS project (in the Multimedia category) is connected with five other projects, P1–P5, from five different project categories (Communication, Database, Desktop, Development, and Editors); in this case, the Blau index[12] is equal to 0.8. In our study, two Blau indexes were computed for OSS project *i* at T_1 — one from the administrator-affiliated network (*admin_blau*$_{it1}$) and the developer-affiliated network *(developer_blau*$_{it1}$). The maximum values of *admin_blau*$_{it1}$ and *developer_blau*$_{it1}$ are 0.898 and 0.906 respectively. Next, the extent of structural holes was regressed on the Blau indexes in each network with GLM. The coefficients of both *admin_blau*$_{it1}$ (coefficient = 0.390 and standard error = 0.070, p-value = 0.001) and *developer_blau*$_{it1}$ (coefficient = 0.142 and standard error = 0.082, p-value = 0.082) were found to be significantly positive, thereby indicating that the OSS project with higher categorical heterogeneity in its connected projects indeed possessed more structural holes in both the administrator- and developer-affiliated networks. Compared with developers, the administrators

[12] This value is calculated as 1- $[(1/5)^2 + (1/5)^2 + (1/5)^2 + (1/5)^2 + (1/5)^2]$.

can more easily switch across different types of projects as soon as they have sharpened their project management or administration skills. In other words, the administrators can engage in more types of projects than the developers because of the flexibility of their knowledge. Thus, $admin\_blau_{it1}$ has a stronger significance level than $developer\_blau_{it1}$.

Two methods were used to test the robustness of our analysis results. First, we replaced the GLM with beta regression to test the robustness of our results. The beta regression can be used to estimate the proportional values between 0 and 1 but excluding the 0 and 1 (Wooldridge 2010). By referring to the descriptive statistics in Table 2, the minimum value of the two dependent variables is 0. To maintain the consistency of the sample size, we referred to the transformation proposition by Smithson and Verkuilen (2006) and made the following transformation.

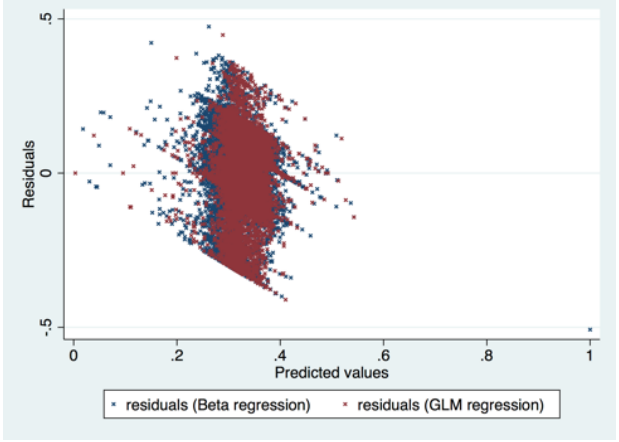
$$traffic\_ratio'_{it2} = {(traffic\_ratio_{it2} * (N-1) + 0.5)}/{N}$$

$$development\_ratio'_{it2} = {(development\_ratio_{it2} * (N-1) + 0.5)}/{N}$$

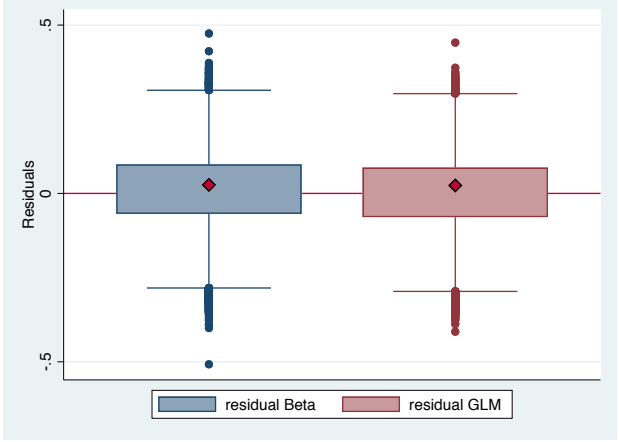
where N is the total number of observations in the sample

The results of the robustness tests are given in Tables 6 and 7. The estimated coefficients for $traffic\_ratio'_{it2}$ are presented from Models 1 to 7 in Table 6. The findings agree with those in Table 4. The results for $development\_ratio'_{it2}$, as the dependent variable are presented in Table 7 from Models 8 to 14. There is a minor exception in the estimated coefficients of the interaction terms presented in Model 13, where the project maturity alleviated the negative relationship between the extent of structural holes and development intensity. Such an exceptional difference may result from the bias introduced by the transformation of the dependent variables. By referring to the log-likelihood, the estimation from GLM (Tables 4 and 5) outperformed those from beta regressions. To further evaluate the results, we calculated the residuals of Model 13 in

Table 5 and Model 13 in Table 7 and visualized them in Figure 2 below. The scatter plot (Figure 2a) indicates substantial overlap, although the residuals from the GLM estimation have a smaller projection area, the substantial overlap between red dots and blue dots (Figure 2a) indicates better goodness-of-fit. The box plot (Figure 2b) confirms the scatter plots.



(a)



(b)

Figure 2. Residual Plots

Table 6. Results of Robustness Test (Change in Traffic Intensity as the Dependent Variable, Beta Model)							
DV	Transformed Traffic intensity change ratio (*traffic_ratio'$_{it2}$*)						
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
admin_sh$_{it1}$	--	0.079** (0.029)	--	0.036 (0.039)	0.212** (0.071)	--	--
developer_sh$_{it1}$	--	--	0.083** (0.03)	--	--	0.010 (0.040)	0.206** (0.07)
Dev_stage$_{it1}$< Beta	--	--	--	--	-0.225*** (0.033)	--	-0.216*** (0.034)
Dev_stage$_{it1}$= Beta	--	--	--	0.225*** (0.033)	--	0.216*** (0.034)	--
Dev_stage$_{it1}$> Beta	--	--	--	0.391*** (0.028)	0.166*** (0.036)	0.385*** (0.029)	0.169*** (0.037)
admin_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	-0.176* (0.08)	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	0.176* (0.08)	--	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	0.087 (0.064)	-0.088 (0.087)	--	--
developer_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	--	--	-0.196* (0.08)
developer_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	--	--	0.196* (0.08)	--
developer_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	--	--	0.126* (0.064)	-0.069 (0.086)
InEqul$_{it1}$	0.647*** (0.063)	0.484*** (0.066)	0.544*** (0.067)	0.415*** (0.064)	0.415*** (0.064)	0.458*** (0.065)	0.458*** (0.065)
use_mail$_{it1}$	0.205*** (0.025)	0.192*** (0.027)	0.195*** (0.027)	0.192*** (0.026)	0.192*** (0.026)	0.199*** (0.027)	0.199*** (0.027)
use_pm$_{it1}$	-0.224*** (0.025)	-0.236*** (0.026)	-0.213*** (0.027)	-0.202*** (0.026)	-0.202*** (0.026)	-0.18*** (0.026)	-0.18*** (0.026)
use_forum$_{it1}$	-0.134*** (0.023)	-0.106*** (0.025)	-0.128*** (0.025)	-0.107*** (0.025)	-0.107*** (0.025)	-0.128*** (0.025)	-0.128*** (0.025)
use_news$_{it1}$	0.269*** (0.032)	0.275*** (0.034)	0.238*** (0.035)	0.303*** (0.034)	0.303*** (0.034)	0.263*** (0.034)	0.263*** (0.034)
use_screenshots$_{it1}$	-0.165*** (0.031)	-0.149*** (0.032)	-0.154*** (0.033)	-0.166*** (0.032)	-0.166*** (0.032)	-0.167*** (0.032)	-0.167*** (0.032)
use_wiki$_{it1}$	0.115*** (0.027)	0.084** (0.028)	0.125*** (0.028)	0.138*** (0.028)	0.138*** (0.028)	0.181*** (0.028)	0.181*** (0.028)
Lang_Pop$_{it1}$	-234.216 (215.382)	-254.612 (215.209)	-255.455 (215.215)	-208.857 (212.087)	-208.857 (212.087)	-242.765 (208.715)	-242.765 (208.715)
Team_Tenure$_{it1}$	0.032*** (0.003)	0.03*** (0.004)	0.035*** (0.004)	0.005 (0.004)	0.005 (0.004)	0.01** (0.004)	0.01** (0.004)
Net_Size$_{it1}$	0.024*** (0.001)	0.025*** (0.001)	0.024*** (0.001)	0.021*** (0.001)	0.021*** (0.001)	0.02*** (0.001)	0.02*** (0.001)
constant	-3.362*** (0.213)	-3.327*** (0.214)	-3.324*** (0.22)	-3.239*** (0.212)	-3.014*** (0.215)	-3.165*** (0.217)	-2.949*** (0.219)
Lang$_i$, License$_i$, Category$_i$: Included but not reported							
Log-pseudo likelihood	21318.134	19896.251	19164.078	20105.381	20105.381	19372.085	19372.085
+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*							

Table 7. Results of Robustness Test (Change in Development Intensity as the Dependent Variable, Beta Model)							
DV	Transformed Development intensity change ratio (*development_ratio'$_{it2}$*)						
	Model 8	Model 9	Model 10	Model 11	Model 12	Model 13	Model 14
admin_sh$_{it1}$	--	-0.256*** (0.022)	--	-0.285*** (0.029)	-0.223*** (0.052)	--	--
developer_sh$_{it1}$	--	--	-0.179*** (0.022)	--	--	-0.233*** (0.029)	-0.129* (0.052)
Dev_stage$_{it1}$< Beta	--	--	--	--	-0.201*** (0.024)	--	-0.195*** (0.024)
Dev_stage$_{it1}$= Beta	--	--	--	0.201*** (0.024)	--	0.195*** (0.024)	--
Dev_stage$_{it1}$> Beta	--	--	--	0.213*** (0.021)	0.012 (0.026)	0.199*** (0.021)	0.004 (0.027)
admin_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	-0.062 (0.059)	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	0.062 (0.059)	--	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	0.088+ (0.049)	0.027 (0.066)	--	--
developer_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	--	--	-0.104+ (0.059)
developer_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	--	--	0.104+ (0.059)	--
developer_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	--	--	0.131** (0.048)	0.026 (0.064)
InEqul$_{it1}$	-0.154*** (0.04)	-0.141*** (0.042)	-0.111** (0.042)	-0.183*** (0.042)	-0.183*** (0.042)	-0.159*** (0.042)	-0.159*** (0.042)
use_mail$_{it1}$	-0.033+ (0.018)	-0.032+ (0.019)	-0.046* (0.019)	-0.033+ (0.019)	-0.033+ (0.019)	-0.046* (0.019)	-0.046* (0.019)
use_pm$_{it1}$	-0.037* (0.019)	-0.046* (0.02)	-0.026 (0.02)	-0.033+ (0.019)	-0.033+ (0.019)	-0.014 (0.019)	-0.014 (0.019)
use_forum$_{it1}$	0.021 (0.017)	0.021 (0.018)	0.024 (0.018)	0.027 (0.018)	0.027 (0.018)	0.03+ (0.018)	0.03+ (0.018)
use_news$_{it1}$	0.054* (0.024)	0.057* (0.025)	0.061* (0.025)	0.074** (0.025)	0.074** (0.025)	0.078** (0.025)	0.078** (0.025)
use_screenshots$_{it1}$	0.013 (0.024)	0.007 (0.025)	0.013 (0.025)	0.001 (0.025)	0.001 (0.025)	0.011 (0.025)	0.011 (0.025)
use_wiki$_{it1}$	-0.066*** (0.02)	-0.08*** (0.021)	-0.066** (0.021)	-0.05* (0.021)	-0.05* (0.021)	-0.035+ (0.021)	-0.035+ (0.021)
Lang_Pop$_{it1}$	-196.529 (248.034)	-215.185 (247.516)	-217.211 (246.111)	-177.077 (246.182)	-177.077 (246.182)	-190.493 (244.696)	-190.493 (244.696)
Team_Tenure$_{it1}$	0.02*** (0.003)	0.021*** (0.003)	0.022*** (0.003)	0.005+ (0.003)	0.005+ (0.003)	0.005+ (0.003)	0.005+ (0.003)
Net_Size$_{it1}$	-0.006*** (0.001)	-0.005*** (0.001)	-0.004*** (0.001)	-0.008*** (0.001)	-0.008*** (0.001)	-0.008*** (0.001)	-0.008*** (0.001)
constant	-0.458** (0.152)	-0.371* (0.151)	-0.426** (0.153)	-0.316* (0.15)	-0.115 (0.151)	-0.361* (0.152)	-0.166 (0.154)
Lang$_i$, License$_i$, Category$_i$: Included but not reported							
Log-pseudo likelihood	7848.265	7303.360	7108.576	7449.9903	7449.9903	7257.874	7257.7546
+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*							

In addition, we computed the extent of structural holes by each project category to further confirm the contingent role of interproject connectedness in influencing the change in popularity and knowledge creation. For each OSS project, the projects from other project categories that it is connected to are excluded. Eventually, 36 distinct affiliation networks were constructed based on the two roles of the contributors, namely administrator (*admin_intra_sh*$_{it1}$) and developer (*developer_intra_sh*$_{it1}$). Table 8 shows the overall results for the change in traffic intensity as the dependent variable. In Table 9, the dependent variable is the change to the development intensity. The base models are not presented as the results are generally consistent with the previous findings. Interestingly, we found that the negative relationship between the degree of structural holes and development intensity was strengthened when comparing projects at the post-beta phase with those at the pre-beta phase (Model 9 in Table 9). The most mature OSS projects might be reluctant to make use of the externally sourced information due to their inertia and sunk costs in ongoing developments or operations (Zahra and Hayton 2008).

Table 8. Results of Robustness Test (Extent of structural holes computed from same project category)						
DV	Traffic intensity change ratio (*traffic_ratio*$_{it2}$)					
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
admin_intra_sh$_{it1}$	0.25*** (0.051)	--	0.052 (0.043)	0.269*** (0.067)	--	--
developer_intra_sh$_{it1}$	--	0.235*** (0.058)	--	--	0.061 (0.045)	0.29*** (0.067)
Dev_stage$_{it1}$*< Beta*	--	--	--	-0.275*** (0.034)	--	-0.265*** (0.034)
Dev_stage$_{it1}$*= Beta*	--	--	0.275*** (0.034)	--	0.265*** (0.034)	--
Dev_stage$_{it1}$*> Beta*	--	--	0.479*** (0.029)	0.204*** (0.035)	0.46*** (0.029)	0.195*** (0.035)
admin_sh$_{it1}$*XDev_stage*$_{it1}$*< Beta*	--	--	--	-0.217** (0.078)	--	--
admin_sh$_{it1}$*XDev_stage*$_{it1}$*= Beta*	--	--	0.217** (0.078)	--	--	--
admin_sh$_{it1}$*XDev_stage*$_{it1}$*> Beta*	--	--	0.067 (0.065)	-0.15+ (0.081)	--	--
developer_sh$_{it1}$*XDev_stage*$_{it1}$*< Beta*	--	--	--	--	--	-0.229** (0.078)
developer_sh$_{it1}$*XDev_stage*$_{it1}$*= Beta*	--	--	--	--	0.229** (0.078)	--
developer_sh$_{it1}$*XDev_stage*$_{it1}$*> Beta*	--	--	--	--	0.085 (0.068)	-0.144+ (0.08)
InEqul$_{it1}$	0.821*** (0.093)	0.92*** (0.093)	0.678*** (0.079)	0.678*** (0.079)	0.755*** (0.081)	0.755*** (0.081)
use_mail$_{it1}$	0.155*** (0.03)	0.14*** (0.03)	0.14*** (0.029)	0.14*** (0.029)	0.131*** (0.029)	0.131*** (0.029)
use_pm$_{it1}$	-0.285*** (0.028)	-0.258*** (0.027)	-0.246*** (0.027)	-0.246*** (0.027)	-0.227*** (0.026)	-0.227*** (0.026)
use_forum$_{it1}$	-0.168*** (0.028)	-0.171*** (0.026)	-0.156*** (0.026)	-0.156*** (0.026)	-0.161*** (0.026)	-0.161*** (0.026)
use_news$_{it1}$	0.108** (0.037)	0.091* (0.037)	0.135*** (0.036)	0.135*** (0.036)	0.117*** (0.035)	0.117*** (0.035)
use_screenshots$_{it1}$	-0.103** (0.034)	-0.112*** (0.034)	-0.116*** (0.033)	-0.116*** (0.033)	-0.116*** (0.033)	-0.116*** (0.033)
use_wiki$_{it1}$	0.157*** (0.031)	0.139*** (0.031)	0.215*** (0.03)	0.215*** (0.03)	0.193*** (0.03)	0.193*** (0.03)
Lang_Pop$_{it1}$	-158.107+ (81.124)	-205.529** (77.221)	-135.315* (67.399)	-135.315* (67.398)	-181.888** (66.949)	-181.888** (66.949)
Team_Tenure$_{it1}$	0.025*** (0.004)	0.025*** (0.004)	-0.01* (0.004)	-0.01* (0.004)	-0.01* (0.005)	-0.01* (0.005)
Net_Size$_{it1}$	0.013** (0.004)	0.013** (0.004)	0.01** (0.004)	0.01** (0.004)	0.01** (0.004)	0.01** (0.004)
constant	-3.288*** (0.226)	-3.265*** (0.227)	-3.16*** (0.224)	-2.885*** (0.226)	-3.12*** (0.225)	-2.855*** (0.227)
Lang$_i$*, License*$_i$*, Category*$_i$*: Included but not reported*						
Log-pseudo likelihood	-2507.366	-2497.641	-2490.906	-2490.906	-2481.859	-2481.859
+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*						

Table 9. Results of Robustness Test (Extent of structural holes computed from same project category)

DV	Development intensity change ratio (*development_ratio$_{it2}$*)					
	Model 7	Model 8	Model 9	Model 10	Model 11	Model 12
admin_intra_sh$_{it1}$	-0.154*** (0.033)	--	-0.098* (0.046)	-0.193* (0.077)	--	--
developer_intra_sh$_{it1}$	--	-0.139*** (0.032)	--	--	-0.141** (0.047)	-0.123+ (0.069)
Dev_stage$_{it1}$< Beta	--	--	--	-0.184*** (0.015)	--	-0.182*** (0.015)
Dev_stage$_{it1}$= Beta	--	--	0.184*** (0.015)	--	0.182*** (0.015)	--
Dev_stage$_{it1}$> Beta	--	--	0.186*** (0.013)	0.002 (0.015)	0.171*** (0.013)	-0.012 (0.015)
admin_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	0.095 (0.089)	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	-0.095 (0.089)	--	--	--
admin_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	-0.143* (0.068)	-0.049 (0.09)	--	--
developer_sh$_{it1}$XDev_stage$_{it1}$< Beta	--	--	--	--	--	-0.018 (0.082)
developer_sh$_{it1}$XDev_stage$_{it1}$= Beta	--	--	--	--	0.018 (0.082)	--
developer_sh$_{it1}$XDev_stage$_{it1}$> Beta	--	--	--	--	-0.004 (0.066)	-0.021 (0.081)
InEqul$_{it1}$	-0.137*** (0.035)	-0.091** (0.035)	-0.168*** (0.035)	-0.168*** (0.035)	-0.129*** (0.035)	-0.129*** (0.035)
use_mail$_{it1}$	-0.045** (0.017)	-0.055*** (0.017)	-0.047** (0.017)	-0.047** (0.017)	-0.056*** (0.017)	-0.056*** (0.017)
use_pm$_{it1}$	-0.025 (0.017)	-0.019 (0.017)	-0.012 (0.017)	-0.012 (0.017)	-0.01 (0.017)	-0.01 (0.017)
use_forum$_{it1}$	0.006 (0.016)	0.01 (0.016)	0.008 (0.016)	0.008 (0.016)	0.012 (0.016)	0.012 (0.016)
use_news$_{it1}$	0.041+ (0.024)	0.034 (0.023)	0.053* (0.023)	0.053* (0.023)	0.047* (0.023)	0.047* (0.023)
use_screenshots$_{it1}$	-0.002 (0.022)	0.012 (0.022)	-0.01 (0.022)	-0.01 (0.022)	0.007 (0.022)	0.007 (0.022)
use_wiki$_{it1}$	-0.056** (0.021)	-0.053* (0.021)	-0.034 (0.021)	-0.034 (0.021)	-0.03 (0.021)	-0.03 (0.021)
Lang_Pop$_{it1}$	-378.693*** (26.261)	-379.869*** (26.382)	-356.879*** (29.402)	-356.879*** (29.402)	-362.038*** (29.016)	-362.038*** (29.016)
Team_Tenure$_{it1}$	0.01*** (0.002)	0.013*** (0.003)	-0.003 (0.003)	-0.003 (0.003)	-0.001 (0.003)	-0.001 (0.003)
Net_Size$_{it1}$	-0.004*** (0.001)	-0.004*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)	-0.006*** (0.001)
constant	-0.322* (0.159)	-0.362* (0.159)	-0.285+ (0.164)	-0.101 (0.165)	-0.317+ (0.164)	-0.135 (0.165)
Lang$_i$, License$_i$, Category$_i$: Included but not reported						
Log-pseudo likelihood	-4747.1604	-4683.228	-4740.445	-4740.445	-4676.736	-4676.736

+p-value < 0.1; *p-value < 0.05; **p-value < 0.01; ***p-value < 0.001; values are displayed in terms of *coefficient (standard error)*

To rule out other potential alternative explanations and strengthen the findings, we conducted several ad hoc tests. First, we conducted two correlation analyses to rule out the possibility of interdependency between the two dependent variables. This concern arises because the extent of structural holes in both administrator- and developer-affiliated networks have an opposite impact on the changes in traffic intensity and development intensity respectively. We referred to our measurements of traffic intensity (TRF_{it}) and development intensity (DVP_{it}) in section 4.2 to calculate their values at T1, namely TRF_{it1} and DVP_{it1}, and calculated their correlation. Their correlation value is 0.053, which indicates that there is no correlation between traffic intensity and development intensity. To further confirm this, we calculated the correlation between the two dependent variables, namely $traffic\_ratio_{it2}$ and $development\_ratio_{it2}$, used in the preceding analyses. The low correlation value (0.039) between these two variables confirms our conclusion.

Second, we calculated the extent of structural holes in the administrator-affiliated network ($admin\_sh_{it2}$) and the developer-affiliated network ($developer\_sh_{it2}$) at T_2 (Dec 2010), and statistically compared them with those at T1, namely $admin\_sh_{it1}$ and $developer\_sh_{it1}$. This approach can rule out the alternative explanation that changes in the dependent variables were influenced by the phases of the network (i.e., network growth or attenuation). The results from the paired sample t-test[13] indicate that there is no difference in the extent of structural holes between T_1 and T_2. There may be two possible explanations for this. On the one hand, some unconnected OSS projects may be bridged by newly affiliated contributors, which reduces the extent of structural holes and increases the network closure. On the other hand, the network of a

[13] We employed the paired sample t-test to statistically test the difference between two timestamps. In the administrator-affiliated network, the difference in mean values between two timestamps is -0.184 and the 95% confidence interval is from -0.191 to -0.176. Thus, the null hypothesis is not rejected. In a similar vein, the difference in mean values of the developer-affiliated network between the two timestamps is -0.126, which is also located within the 95% confidence interval (i.e., from -0.131 to -0.121). Thus, the null hypothesis is not rejected either.

focal OSS project can be expanded by connecting to new projects that otherwise will not be connected, which increases the extent of the structural holes. As both circumstances may concurrently exist, the network of a focal OSS project does not fluctuate drastically over time.

6 Discussions

6.1 Theoretical and Practical Implications

Compared to previous studies that briefly explained the relationship between network structure and OSS project success (Grewal et al. 2006; Singh 2010; Singh et al. 2011), we comprehensively assessed how network structures contributed to OSS project success by taking into consideration the OSS innovation mechanisms. We complement the OSS innovation model with two contributions that add to a more holistic perspective. First, previous studies place an overwhelming emphasis on the role of motivation or incentives in constructing the innovation model of OSS (von Hippel and von Krogh 2003). While acknowledging the importance of accounting for individual participation in OSS innovation, we urge the researchers to take into account interproject connectedness in the theorization of the OSS innovation model. Second, rather than following the extant OSS studies that assessed the success of an OSS project from a single dimension – such as the number of downloads or the number of CVS commits (Subramaniam et al. 2009; Singh et al. 2011) – we used a more holistic investigation of OSS project performance. To the best of our knowledge, this research is the first work that considers both the network structure and the composite assessments of OSS performance. We found that an OSS project could strategically aim to have many structural holes in its ego-network [constructed through its shared contributors with other projects] to gain popularity or it could aim to be situated in the cohesive network to advance its knowledge creation. In view of the network structure, our findings are significant for strategizing the composition of OSS contributors to

align with the divergent definitions of OSS success – i.e., to reach out to more people or to improve the software.

Besides contributing to knowledge on OSS innovation mechanisms, our findings reveal how the OSS project's maturity plays a role in facilitating OSS project popularity. Only the OSS projects that have progressed from a very nascent stage to a developmental stage and are saturated with abundant structural holes appear to benefit from the maturity of the project for greater popularity. This complements the finding on the positive linear relationship between project maturity (i.e., pre-beta, beta, and post-beta) and OSS project success in previous literature (Daniel et al. 2013; Setia et al. 2012). Projects with fewer structural holes had better knowledge creation and this was not affected by project maturity. One potential explanation for this finding is that, in a cohesive network structure, the circulated resources tend to be homogeneous, which means they are relatively easy to harness regardless of the maturity of the OSS project. Another potential explanation is the less mature projects might offer more learning opportunities and more spaces for original creations which attract talented contributors who can effectively co-create even when there is less stable governance and an immuture collaboration structure. Both of these possible explanations warrant a closer look to better understand our finding of the non-significant moderation role of project maturity on knowledge creation.

Besides contributing to theory, an immediate practical implication of this research is that OSS project administrators should consider not only recruiting and sharing essential resources (such as sharing developers with their other projects), but also managing other projects. By being linked to other projects, an OSS project could gain greater diversity of information flow that, in turn, increases its popularity. On the other hand, by encouraging a cohesive network structure, an OSS project could harness the homogenous resources that, in turn, increase knowledge creation.

It is a trade-off that OSS project administrators should carefully assess in the process of OSS project development.

Our findings also have important implications for IT companies. Previous studies have found that several IT companies sponsored OSS projects in order to maintain a steady stream of innovation or achieve novel creations (Watson et al. 2008; Chen et al. 2012; Daniel et al. 2018). For example, IBM initiated the foundation for supporting Linux projects while Oracle invested in MySQL to expand its service lines. In this regard, our findings provide constructive suggestions for the sponsor companies to leverage the OSS project team composition to accomplish their desired outputs. In particular, organizations wanting to achieve innovation creation by engaging the OSS community should sponsor OSS projects whose contributors (i.e., administrators or developers) work in overlapping projects. In contrast, those organizations wanting to leverage OSS projects to promote their products or services should sponsor those OSS projects whose teams are composed of people who work in heterogeneous OSS projects.

OSS development forges – such as SourceForge.net or Github – could also benefit from our findings. We found the contributors' collaborative ties resulted in a trade-off between popularity and knowledge creation. In this regard, those projects with significant knowledge creation achievement might suffer from low popularity. To resolve such a dilemma, the OSS forge operators should adjust their recommendation mechanism in consideration of the collaborative ties among OSS project teams; this could effectively prevent those OSS projects with high innovation potential from being lost among many mediocre ones.

Our findings also shed light on the practice of open innovation in general. Unlike organizations in the IT industry that have frequent and close collaboratiions with the OSS community to co-create concrete products or services, the organizations in conventional

industries mainly leverage open innovation for idea generation (King and Lakhani 2013): achieving an innovative output is not the main purpose of innovation campaigns in organizations in conventional industries. Thus, recruiting external innovators from different backgrounds or from diversified online communities that are engaged in multiple and intersectional open innovation projects is recommended. Such broad participant engagement will also provide greater public exposure for the campaign.

6.2 Limitations and Suggestions for Future Research

Like any other study, this one has several limitations. First, it focuses on the network metric of Burt's constraint index, which reflects the structural holes of an OSS project. Hence, our findings should not be directly integrated with the findings from other OSS studies that also adopt the network perspective but use different network metrics such as direct and indirect ties (Hahn et al. 2008; Tan et al. 2007). An immediate extension of this study would be to triangulate the findings by adopting different network metrics, such as centrality and tie strength.

Second, our measurement of popularity and knowledge creation are based on the marginal differences between the two time periods of data collection, which enable us to temporally separate network instantiations and performance outcome. Although this method is the best approach to address commonly raised concerns (e.g., endogeneity), the interval time gap between the two time periods of data collection could also raise concerns that relate to the continuous evolvement of the projects. Other than the researchers' judgment regarding the time gap, a primary reason for the interval time gap is that adequate time is required for people to know (i.e., gain interest) about a project. Different time gaps could be considered to test the robustness of our findings. Also, future studies could vary the weight of each the components used to measure popularity and knowledge creation.

Third, we chose to anchor our empirical investigation on SourceForge.net to align with the stream of OSS research, in which many studies use the same data source. In recent years, other OSS communities such as Github have emerged. Future research could consider extending our study to test the robustness of our findings in these other emerging OSS communities (Medappa and Srivastava 2019).

The opportunities for future research stemming from our study are many. We hope our study serves as a beginning and encourages more evolved research ideas and inquiries into this topic.

7 Concluding Remarks

Nowadays, open source software projects are more connected than before. The emergence of OSS development forges allows these projects to share valuable resources. Our findings reveal the contingent role of interproject connectedness (i.e., of an OSS projects' ego network structure) on the OSS project success. A project with more structural holes in its ego network is more likely to gain popularity, however, a cohesive network structure (i.e., with fewer structural holes) is associated with better knowledge creation. We also observed that the positive relationship between structural holes and project popularity could be further enhanced by the maturity of the OSS project. However, the negative relationship between the extent of structural holes and knowledge creation was not affected by the maturity of the OSS projects. We hope our findings will inspire subsequent research that considers an OSS project as a vertex in a network of OSS projects and its connectedness as a determinant of its success.

8 Acknowledgement

This study is supported by the National Natural Science Foundation of China (71702133, 71704078, 71801217, and 71532015) and Singapore Ministry of Education Academic Research Fund Tier 1, R-253-000-121-133.

9 References

Aberdour, M. 2007. Achieving quality in open-source software. IEEE Software 24, 58-64. https://doi.org/10.1109/MS.2007.2

Ahuja, G. 2000. Collaboration networks, structural holes, and innovation: A longitudinal study. Administrative Science Quarterly 45, 425-455. https://doi.org/10.2307%2F2667105

Austin, J. R. 2003. Transactive memory in organizational groups: the effects of content, consensus, specialization, and accuracy on group performance. Journal of Applied Psychology 88, 866-878. http://dx.doi.org/10.1037/0021-9010.88.5.866

Autio, E., Sapienza, H. J., and Almeida, J. G. 2000. Effects of age at entry, knowledge intensity, and imitability on international growth. Academy of Management Journal 43, 909-924. https://doi.org/10.5465/1556419

Balkundi, P., Kilduff, M., Barsness, Z. I., and Michael, J. H. 2007. Demographic antecedents and performance consequences of structural holes in work teams. Journal of Organizational Behavior 28, 241-260. https://doi.org/10.1002/job.428

Beckman, C. M., and Haunschild, P. R. 2002. Network learning: The effects of partners' heterogeneity of experience on corporate acquisitions. Administrative Science Quarterly 47, 92-124. https://doi.org/10.2307/3094892

Bird, C., Bachmann, A., Aune, E., Duffy, J., Bernstein, A., Filkov, V., & Devanbu, P. 2009. Fair and balanced?: bias in bug-fix datasets. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (pp. 121-130). ACM. https://doi.org/10.1145/1595696.1595716

Blau, P. M. 1977. *Inequality and heterogeneity: A primitive theory of social structure*. New York: Free Press.

Burt, R. S. 1992. *Structural holes: The social structure of competition*. Cambridge, MA: Harvard Business School Press.

Chen, X., and Dietrich, G. 2009. Knowledge Location, Differentiation, Credibility and Coordination in Open Source Software Development Teams, In, *15th Americas Conference on Information Systems (AMCIS 2009)*, August 6th-9th, San Francisco, California, US, 2009.

Chen, L., Marsden, J. R., and Zhang, Z. 2012. Theory and analysis of company-sponsored value co-creation. Journal of Management Information Systems 29, 141-172. https://doi.org/10.2753/MIS0742-1222290206

Chengalur-Smith, S., and Sidorova, A. 2003. Survival of open-source projects: A population ecology perspective. In, 24th International Conference on Information Systems (ICIS 2003), December 14th-17th, Seattle, Washington, US, 2003.

Coad, A., Segarra, A., and Teruel, M. 2016. Innovation and firm growth: Does firm age play a role?. Research Policy 45, 387-400. https://doi.org/10.1016/j.respol.2015.10.015

Coleman, J. S. 1988. Free riders and zealots: The role of social networks. Sociological Theory, 6, 52-57. http://doi.org/10.2307/201913

Crowston, K., and Howison, J. 2006. Hierarchy and centralization in free and open source software team communications. Knowledge, Technology & Policy 18, 65-85. https://doi.org/10.1007/s12130-006-1004-8

Crowston, K., Howison, J., and Annabi, H. 2006. Information systems success in free and open source software development: Theory and measures. Software Process: Improvement and Practice 11, 123-148. https://doi.org/10.1002/spip.259

Crowston, K., Li, Q., Wei, K., Eseryel, U. Y., and Howison, J. 2007. Self-organization of teams for free/libre open source software development. Information and Software Technology, 49, 564-575. https://doi.org/10.1016/j.infsof.2007.02.004

Crowston, K., Wei, K., Howison, J., and Wiggins, A. 2008. Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys* 44, Article 7. https://doi.org/10.1145/2089125.2089127

Cummings, J. N., and Cross, R. 2003. Structural properties of work groups and their consequences for performance. Social Networks 25, 197-210. https://doi.org/10.1016/S0378-8733(02)00049-7

Daniel, S., Agarwal, R., and Stewart, K. J. 2013. The effects of diversity in global, distributed collectives: A study of open source project success. Information Systems Research 24, 312-333. https://doi.org/10.1287/isre.1120.0435

Daniel, S., Midha, V., Bhattacherhjee, A., and Singh, S. P. 2018. Sourcing knowledge in open source software projects: The impacts of internal and external social capital on project success. The Journal of Strategic Information Systems, 27, 237-256. https://doi.org/10.1016/j.jsis.2018.04.002

Dong, J. Q., Wu, W., and Zhang, Y. S. 2019. The faster the better? Innovation speed and user interest in open source software. Information & Management, 56, 669-680. https://doi.org/10.1016/j.im.2018.11.002

Eagle, N., Macy, M., and Claxton, R. 2010. Network diversity and economic development, Science 328, 1029-1031. https://doi.org/10.1126/science.1186605

Everett, M., and Borgatti, S. P. 2005. Ego network betweenness. Social Networks 27, 31-38. https://doi.org/10.1016/j.socnet.2004.11.007

Feller, J., Finnegan, P., Fitzgerald, B., and Hayes, J. 2008. From peer production to productization: A study of socially enabled business exchanges in open source service networks. Information Systems Research, 19, 475-493. https://dx.doi.org/10.1287/isre.1080.0207

Gao, Y., and Madey, G., 2007. Network analysis of the SourceForge. net community, In, J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti. (eds.), *Open Source Development, Adoption and Innovation*, US, Springer US: pp. 187-200. https://doi.org/10.1007/978-0-387-72486-7_15

Garriga, H., Spaeth, S., & Von Krogh, G. 2011. Open Source Software Development: Communities' Impact on Public Good. In International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction (pp. 69-77). Springer, Berlin, Heidelberg.

Ghosh, R. 2006. *Collaborative ownership and the digital economy*. Cambridge, MA: The MIT Press.

Grewal, R., Lilien, G. L. and Mallapragada, G. 2006. Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. Management Science 52, 1043-1056. https://doi.org/10.1287/mnsc.1060.0550

Gargiulo, M., and Benassi, M. 2000. Trapped in your own net? Network cohesion, structural holes, and the adaptation of social capital. Organization Science, 11, 183-196. https://doi.org/10.1287/orsc.11.2.183.12514

Gulati, R., and Higgins, M. C. 2003. Which ties matter when? The contingent effects of interorganizational partnerships on IPO success. Strategic Management Journal 24, 127-144. https://www.jstor.org/stable/20060517

Hahn, J., Moon, J. Y., and Zhang, C. 2008. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. Information Systems Research 19, 369-391. https://doi.org/10.1287/isre.1080.0192

Harrison, D. A., and Klein, K. J. 2007. What's the difference? Diversity constructs as separation, variety, or disparity in organizations. Academy of Management Review 32, 1199-1228. https://doi.org/10.5465/amr.2007.26586096

Harrison, D. A., Price, K. H., and Bell, M. P. 1998. Beyond relational demography: Time and the effects of surface-and deep-level diversity on work group cohesion. Academy of Management Journal 41, 96-107. https://doi.org/10.5465/256901

Harrison, D. A., Price, K. H., Gavin, J. H., and Florey, A. T. 2002. Time, teams, and task performance: Changing effects of surface-and deep-level diversity on group functioning. Academy of Management Journal 45, 1029-1045. https://doi.org/10.5465/3069328

He, J., Butler, B. S., and King, W. R. 2007. Team cognition: Development and evolution in software project teams. Journal of Management Information Systems, 24, 261-292. https://doi.org/10.2753/MIS0742-1222240210

Healy, K., and Schussman, A. 2003. The ecology of open-source software development. Technical report, University of Arizona, USA. https://pdfs.semanticscholar.org/2c89/092af57b5c4508dd65863df5602c90d7bbb6.pdf

Heckman, R., Crowston, K., Eseryel, U. Y., Howison, J., Allen, E., and Li, Q., Emergent decision-making practices in free/libre open source software (FLOSS) development teams, In, , J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti. (eds.), *Open Source Development, Adoption and* Innovation, US, Springer US: 2007, pp. 71-84. https://doi.org/10.1007/978-0-387-72486-7_6

Holmqvist, M. 2003. A dynamic model of intra-and interorganizational learning. Organization Studies 24. 95-123. https://doi.org/10.1177/0170840603024001684

Jackson, S. E., May, K. E., and Whitney, K. 1995. Understanding the dynamics of diversity in decision-making team, In, Guzzo, R.A., Salas, E., and Associates, *Team Effectiveness and Decision Making in Organizations*, San Francisco: Jossey-Bass: 1995, 204-261.

Jiang, Q., Tan, C. H., Sia, C. L., & Wei, K. K. 2019. Followership in an Open-Source Software Project and its Significance in Code Reuse. Mis Quarterly, 43, 1303-1319. http://doi.org/10.25300/MISQ/2019/14043.

Jones, O. 2006. Developing absorptive capacity in mature organizations: The change agent's role. Management Learning 37. 355-376. https://doi.org/10.1177/1350507606067172

Jurison, J. 1999. Software project management: the manager's view. Communications of the AIS 2, article 2. https://doi.org/10.17705/1CAIS.00217

King, A., and Lakhani, K. R. 2013. Using open innovation to identify the best ideas. MIT Sloan Management Review 55, 41-48.

Knight, D., Pearce, C. L., Smith, K. G., Olian, J. D., Sims, H. P., Smith, K. A., and Flood, P. 1999. Top management team diversity, group process, and strategic consensus. Strategic Management Journal 20, 445-465. https://doi.org/10.1002/(SICI)1097-0266(199905)20:5<445::AID-SMJ27>3.0.CO;2-V

Kuk, G. 2006. Strategic interaction and knowledge sharing in the KDE developer mailing list. Management Science 52, 1031-1042. https://doi.org/10.1287/mnsc.1060.0551

Lerner, J., and Tirole, J. 2002. Some simple economics of open source. The Journal of Industrial Economics, 50, 197-234. https://doi.org/10.1111/1467-6451.00174

Li, X., Hess, T. J., and Valacich, J. S. 2008. Why do we trust new technology? A study of initial trust formation with organizational information systems. The Journal of Strategic Information Systems 17. 39-71. https://doi.org/10.1016/j.jsis.2008.01.001

Lin, B., Robles, G., and Serebrenik, A. 2017. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)* (pp. 66-75). IEEE.

Lindsjørn, Y., Sjøberg, D. I., Dingsøyr, T., Bergersen, G. R., and Dybå, T. 2016. Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software, 122, 274-286. https://doi.org/10.1016/j.jss.2016.09.028

MacCormack, A. 2001. How internet companies build software. MIT Sloan Management Review 42. 75-84.

Medappa, P. K., and Srivastava, S. C. 2019. Does Superposition Influence the Success of FLOSS Projects? An Examination of Open-Source Software Development by Organizations and Individuals. Information Systems Research, 30, 764-786. https://doi.org/10.1287/isre.2018.0829

Nerkar, A., and Paruchuri, S. 2005. Evolution of R&D Capabilities: The Role of Knowledge Networks within a Firm, Management Science 5, 771-785. https://doi.org/10.1287/mnsc.1040.0354

Patel, P. C., Kohtamäki, M., Parida, V., and Wincent, J. 2015. Entrepreneurial orientation-as-experimentation and firm performance: The enabling role of absorptive capacity. Strategic Management Journal 36, 1739-1749. https://doi.org/10.1002/smj.2310

Paulson, L. D. 2007. Developers shift to dynamic programming languages. Computer 40, 12-15. https://doi.org/10.1109/MC.2007.53

Peng, G., Wan, Y., and Woodlock, P. 2013. Network ties and the success of open source software development. The Journal of Strategic Information Systems 22, 269-281. https://doi.org/10.1016/j.jsis.2013.05.001

Podolny, J. M., and Baron, J. N. 1997. Resources and relationships: Social networks and mobility in the workplace. American Sociological Review 62, 673-693. http://dx.doi.org/10.2307/2657354

Ren, Y., Chen, J., & Riedl, J. 2016. The impact and evolution of group diversity in online open collaboration. Management Science 62, 1668-1686. https://doi.org/10.1287/mnsc.2015.2178

Schoonhoven, C. B. Liability of newness, In, Cooper, C., *Wiley Encyclopedia of Management*, Wiley: 2015, https://doi.org/10.1002/9781118785317.weom030067

Seibert, S. E., Kraimer, M. L., and Liden, R. C. 2001. A social capital theory of career success. Academy of management journal 44, 219-237. https://doi.org/10.5465/3069452

Setia, P., Rajagopalan, B., Sambamurthy, V., and Calantone, R. 2012. How peripheral developers contribute to open-source software development. Information Systems Research 23, 144-163. https://doi.org/10.1287/isre.1100.0311

Shah, S. K. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. Management science, 52, 1000-1014. http://dx.doi.org/10.1287/mnsc.1060.0553

Shipilov, A.V. 2009. Firm Scope Experience, Historic Multimarket Contact with Partners, Centrality, and the Relationship Between Structural Holes and Performance. Organization Science 20, 85-106. https://doi.org/10.1287/orsc.1080.0365

Singh, P. V. 2010. The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success. ACM Transactions on Software Engineering and Methodology, 20, Article 6. http://doi.org/10.1145/1824760.1824763

Singh, P. V., Tan, Y., and Mookerjee, V. 2011. Network effects: the influence of structural capital on open source project success. Management Information Systems Quarterly 35, 813-830. http://doi.org/10.2307/41409962

Shen, C., and Monge, P. 2011. Who connects with whom? A social network analysis of an online open source software community. *First Monday* 16, 6, https://www.firstmonday.dk/ojs/index.php/fm/article/view/3551/2991

Smithson, M., and Verkuilen, J. 2006. A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. Psychological Methods 11. 54-71. http://dx.doi.org/10.1037/1082-989X.11.1.54

Stewart, K. J., Ammeter, A. P., and Maruping, L. M. 2006. Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. Information Systems Research, 17, 126-144. https://doi.org/10.1287/isre.1060.0082

Subramaniam, C., Sen, R., and Nelson, M. L. 2009. Determinants of open source software project success: A longitudinal study. Decision Support Systems 46, 576-585. https://doi.org/10.1016/j.dss.2008.10.005

Tan, Y., Mookerjee, V., and Singh, P.V. 2007. Social Capital, Structural Holes and Team Composition: Collaborative Networks of the Open Source Software Community, In, *2007 International Conference on Information Systems (ICIS 2007)*, December 9th–12th, Montreal, Quebec, Canada: 2007.

Thon, D. 1982. An axiomatization of the Gini coefficient. Mathematical Social Sciences 2, 131-143. https://doi.org/10.1016/0165-4896(82)90062-2

Tortoriello, M. 2015. The social underpinnings of absorptive capacity: The moderating effects of structural holes on innovation generation based on external knowledge. Strategic Management Journal 36, 586-597. https://doi.org/10.1002/smj.2228

Tullio, D. D., and Staples, D. S. 2013. The Governance and Control of Open Source Software Projects. Journal of Management Information Systems 30, 49-80. https://doi.org/10.2753/MIS0742-1222300303

Van Knippenberg, D., De Dreu, C. K. W., and Homan, A. C. 2004. Work group diversity and group performance: An integrative model and research agenda. Journal of Applied Psychology 89, 1008-1022. http://doi.org/10.1037/0021-9010.89.6.1008

Van Knippenberg, D., and Schippers, M. C. 2007. Work group diversity. Annual Review of Psychology. 58, 515-541. https://doi.org/10.1146/annurev.psych.58.110405.085546

von Hippel, E., and von Krogh, G. 2003. Open source software and the "private-collective" innovation model: Issues for organization science. Organization Science 14, 209-223. https://doi.org/10.1287/orsc.14.2.209.14992

Wasserman, S., and Faust, K. 1994. *Social network analysis: Methods and applications (Vol. 8)*, Cambridge, UK: Cambridge University Press. https://doi.org/10.1017/CBO9780511815478

Watson, R. T., Boudreau, M. C., York, P. T., Greiner, M. E., and Wynn Jr, D. 2008. The business of open source. Communications of the ACM 51, 41-46. https://doi.org/10.1145/1330311.1330321

Wen, W., Forman, C., and Graham, S. J. 2013. Research note—The impact of intellectual property rights enforcement on open source software project success. Information Systems Research 24. 1131-1146. https://doi.org/10.1287/isre.2013.0479

Wooldridge, J. M. *Econometric analysis of cross section and panel data*, Cambridge, US: MIT Press, 2010.

Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N., and Malone, T. W. 2010. Evidence for a Collective Intelligence Factor in the Performance of Human Groups. Science 330, 686-688. https://doi.org/10.1126/science.1193147

Xiao, Z., and Tsui, A. S. 2007. When brokers may not work: The cultural contingency of social capital in Chinese high-tech firms. Administrative Science Quarterly 52, 1-31. https://doi.org/10.2189/asqu.52.1.1

Zaheer, A., and Soda, G. 2009. Network evolution: The origins of structural holes. Administrative Science Quarterly, 54, 1-31. http://doi.org/10.2189/asqu.2009.54.1.1

Zahra, S. A., and George, G. 2002. Absorptive capacity: A review, reconceptualization, and extension. Academy of Management Review 27. 185-203. https://doi.org/10.2307/4134351

Zahra, S. A., and Hayton, J. C. 2008. The effect of international venturing on firm performance: The moderating influence of absorptive capacity. Journal of Business Venturing 23. 195-220. https://doi.org/10.1016/j.jbusvent.2007.01.001

Zhu, K. X., and Zhou, Z. Z. 2012. Research note—Lock-in strategy in software competition: Open-source software vs. proprietary software. Information Systems Research 23, 536-545. https://doi.org/10.1287/isre.1110.0358