journal homepage: www.elsevier.com/locate/emj

Survival factors for Free Open Source Software projects: A multi-stage perspective

Jing Wang [*]

Department of Decision Sciences, Whittemore School of Business and Economics, University of New Hampshire, United States

KEYWORDS
Free Open Source Software;
Survival;
Software characteristics;
Developer characteristics;
Social networks

Summary This paper uses a large data set of Free Open Source Software (FOSS) projects obtained from SourceForge.net to investigate survival factors at various stages of a FOSS project's lifecycle. It distinguishes FOSS projects that are still at the initial stage of development from those at the growth stage, and posits that the relative importance of the identified survival factors changes as FOSS projects evolve from one stage to the next. The results demonstrate the changing effect of factors over time affecting FOSS survival. Restrictive FOSS licenses and large internal and external networks are found to present advantages for projects that are at the initial stage, but the advantages dissipate as the projects move into the growth stage. Projects with high-quality external networks, greater levels of user/developer participation and service quality, and projects targeted at technical users have a higher likelihood of surviving at both stages. These findings show that a FOSS project team needs to be aware of the conditioning effect of time and focus on the appropriate mix of survival factors as the project moves from one stage to the next.

Introduction

Free Open Source Software (FOSS) projects are emerging as a significant economic, social, and cultural phenomenon, drawing considerable attention from academics and practitioners (Lee, Kim, & Gupta, 2009; von Hippel & von Krogh, 2003). Fueling this interest are successful FOSS projects including the GNU/Linux operating system, Mozilla web browser, the Apache web server, MySQL, and the PHP programming language (Lee et al., 2009; von Hippel & von Krogh, 2003). Despite these successful examples, many FOSS projects have ended in failure (Lee et al., 2009; Wu, Goh & Tang, 2007). Thus, an increasing number of studies have been devoted to understanding the factors that significantly affect FOSS project outcomes (Lee et al., 2009; Subramaniam, Sen, & Nelson, 2009; Wu et al., 2007). Nevertheless, these studies have largely been static in nature and have assumed a stable impact of factors on FOSS outcomes throughout the project's lifecycle. This assumption ignores the possibility that the relative importance of factors may change as FOSS projects evolve from one stage to the next. This is problematic as different stages could present different sets of challenges. What works at one stage may not necessarily work at another. Therefore,

[*] Tel.: +1 603 862 3329; fax: +1 603 862 3383.
E-mail address: Jing.wang@unh.edu

0263-2373/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.emj.2012.03.001

if we are to understand what leads to better FOSS project outcomes, we must reexamine this issue through a new conceptual lens, asking how our interpretation on this issue would differ if factors such as project life stage are considered explicitly.

Motivated by high abandon rates among FOSS projects (Comino, Manenti, & Parisi, 2007; Schweik & English, 2007; Schweik & Semenov, 2003), this research focuses on one particular outcome: FOSS project survival. Given the voluntary nature of FOSS development, it is not uncommon that a FOSS project is abandoned prematurely (Schweik & English, 2007; Schweik & Semenov, 2003). Survival is thus a fundamental aspect of FOSS project performance and of utmost concern for many project teams. Despite this, research that explicitly examines FOSS survival remains scarce. Thus, this study addresses these gaps in the literature by identifying survival factors of FOSS projects. Rather than assuming that the criticality of these factors remains stable throughout the life cycle of a FOSS project, this research distinguishes FOSS projects that are at the initial stage from those at the growth stage, and argues for a contingency perspective which emphasizes that the relative importance of the survival factors is subject to change depending on the stage in which the project currently resides.

If these factors are found to reliably predict subsequent survival, they can function as warning indications for FOSS project managers to assess the project's chance of surviving, diagnose likely problems, and take corrective actions. Knowing about and acting on warning signs at the earliest possible stage have been found to significantly improve the probability of successful software project outcomes (Kappelman, McKeeman, & Zhang, 2006). Nevertheless, many well documented warning indications identified for traditional software projects may not be readily extended to the FOSS domain due to the important difference between these two types of projects. FOSS managers may also face key challenges including the lack of project management experience and the dearth of available support. Diagnosing, monitoring, and managing project risks and problems can hence be extremely challenging for many FOSS managers. This study focuses on FOSS projects that are at the initial stage as well as those at the growth stage. The survival indicators identified here will hence enable FOSS project managers to recognize red flags at the earliest possible stage, to assess whether the red flags are severe enough to warrant significant project redirection, and possibly to redirect the project to success.

The article begins with a brief review of the relevant literature and a comparative discussion of the different challenges FOSS projects face at the initial versus growth stage. Building on this discussion, the article then presents theoretical arguments on the changes in the relative importance of the identified survival factors as FOSS projects move through the two distinct stages. The proposed hypotheses are then empirically evaluated by analyzing data from the world's largest FOSS development data repository, SourceForge.net (SF hereafter). Finally, the findings, implications, and limitations of this research are presented and suggestions for future research are discussed.

Theoretical background

The importance of studying FOSS survival factors

Despite the high failure rate among FOSS projects (Comino et al., 2007; Schweik & English, 2007; Schweik & Semenov, 2003), research that explicitly identifies FOSS survival factors is scarce. The only notable exception is the work done by Chengalur-Smith and Sidorova (2003), which posits that project reliability, age, size, and niche focus are likely predictors of FOSS project survival. Valuable as this effort has been, it does not empirically test its propositions.

Identifying factors that reliably predict FOSS project survival is particularly valuable for two reasons. First, these factors can function as warning indications for FOSS project managers to assess their projects chance of surviving, diagnose likely problems, and take corrective actions. FOSS projects differ from traditional software projects in important ways that FOSS project managers cannot rely on many of the warning indicators commonly recommended for managing traditional software projects. For instance, the FOSS practice eschews many of the traditional software engineering and project management principles including plans, system-level design, schedules, and defined process management (Mockus, Fielding, & Herbsleb, 2002). Therefore, although issues including no documented milestone deliverables and due dates, no project status progress monitoring, and no defined processes for user requirement collection have been suggested to be important indicators for the survival and success of traditional projects (Kappelman et al., 2006), they cannot be easily extended to the FOSS domain. Second, a FOSS project typically commences because one or a small group of like-minded programmers, many of whom have little project management experience, have a shared but unfilled personal software challenge to tackle. Its project members are volunteers who have other jobs and hence can only contribute to the project when they have time. The lack of project management experience on the part of the project manager(s) may make the diagnosis of project risks and problems particularly difficult, preventing the managers from taking corrective action and successfully turning the project around when necessary. With time constraints, the volunteer members may be forced to be technically focused, concentrating on the development of the software code rather than monitoring and managing project risks and problems. Given these reasons, studies that systematically identify the predictors of FOSS project survival is vital. With such knowledge, FOSS managers will be able to better assess whether their project stands a chance of surviving. When such assessments can be made with relative ease and accuracy, particularly at an early stage, FOSS project managers can use this information to redirect their projects and possibly steer the projects to success.

Predictors of FOSS project outcome

In identifying FOSS project survival factors, this research turns to the literature on the predictors of FOSS project outcomes for insights as this area of research is relatively more developed.

The predictors of FOSS project outcomes have been examined from a variety of perspectives. Earlier contributions in this area focus on well-known FOSS projects and offer conceptual models, case studies, and anecdotal evidence. For example, Mockus et al. (2002) examine Apache and Mozilla and their findings indicate that a critical mass of devoted core developers and peripheral developers are crucial for the projects' success. Bonaccorsi and Rossi (2003) focus on Apache and Linux and suggest that open-source systems are less successful in the client market compared to the server market because the former has already been dominated by Microsoft products. Therefore, market structure functions as a barrier to entry for the client-end FOSS products.

More recently, studies with larger sample sizes have begun to emerge. For instance, Stewart, Ammeter, and Maruping (2006) concentrate on license restrictiveness and organizational sponsorship and find that users are most attracted to projects that employ nonrestrictive licenses and that are sponsored by organizations. Focusing on social network structure, Grewal, Lilien, and Mallapragada (2006) suggest that project network embeddedness positively influences both the acceptance rate and technical improvements of FOSS projects. Stewart and Gosain (2006) focus on FOSS community ideology and find that team members' adherence to the FOSS community ideology influences FOSS team effectiveness by enhancing trust and communication quality. These studies typically focus on one or two categories of projects such as games, communications etc., and their findings may not be generalizable to a wider population. The work of Subramaniam et al. (2009) and Comino et al. (2007) represents two of the few empirical studies that are based on comprehensive datasets.

Synthesis of the literature helps the identification of possible FOSS survival factors, which can be broadly categorized into three classes: developer, software, and community characteristics (see Table 1). Developer characteristics refer to the attributes of individual developers affiliated with a particular FOSS project; software characteristics refer to the attributes related to the software product; and community characteristics refer to the attributes describing the project team and the interaction between the team and its larger social environment. Among factors listed in Table 1, six have been recurrently found to impact FOSS project outcomes and will be the focus of this research. These factors include user/developer participation, service quality, license restrictiveness, targeted users, network size, and quality of the external ties. The decision to focus on these six factors, and not the others, is also motivated by the fact that data representing such factors are typically readily available to the project managers. Given that most FOSS project managers lack project management experience and are constrained by resources in monitoring and managing project risks and problems, these readily available data will best serve FOSS project managers' efforts in detecting warning signs, assessing the survival probability of their projects, and taking corrective action where necessary.

Synthesis of the literature also helps in the identification of limitations of previous studies. The majority of studies in this area have taken a static view and assumed that the criteria for measuring FOSS project outcomes remain stable throughout the life of the project and so do the outcome predictors. This is problematic because researchers have suggested that FOSS projects go through distinct stages (Capiluppi & Herraiz, 2007; Schweik, 2005; Schweik & Semenov, 2003) and research from the general project management literature has found lifecycle to be a key explanatory construct in influencing a project's outcome (Hoegl & Weinkauf, 2005; Pinto & Prescott, 1988). As a FOSS project proceeds through its life cycle, it may entail distinct characteristics and encounter unique challenges at each stage. Consequently, the criteria for measuring outcomes may change with the different stages that a FOSS project goes through and certain outcome predictors could be more critical at one stage than at another. Thus, the following section discusses the FOSS project lifecycle stages and the characteristics and challenges associated with each stage.

FOSS projects' lifecycle stages

Lifecycle stage models are not new to the FOSS literature (Capiluppi et al., 2007; Schweik, 2005; Schweik & Semenov, 2003). Drawing on the general software lifecycle literature (Rajlich & Bennett, 2000), Schweik (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003) was among the first to propose that FOSS projects typically go through two identifiable stages: initial and growth.

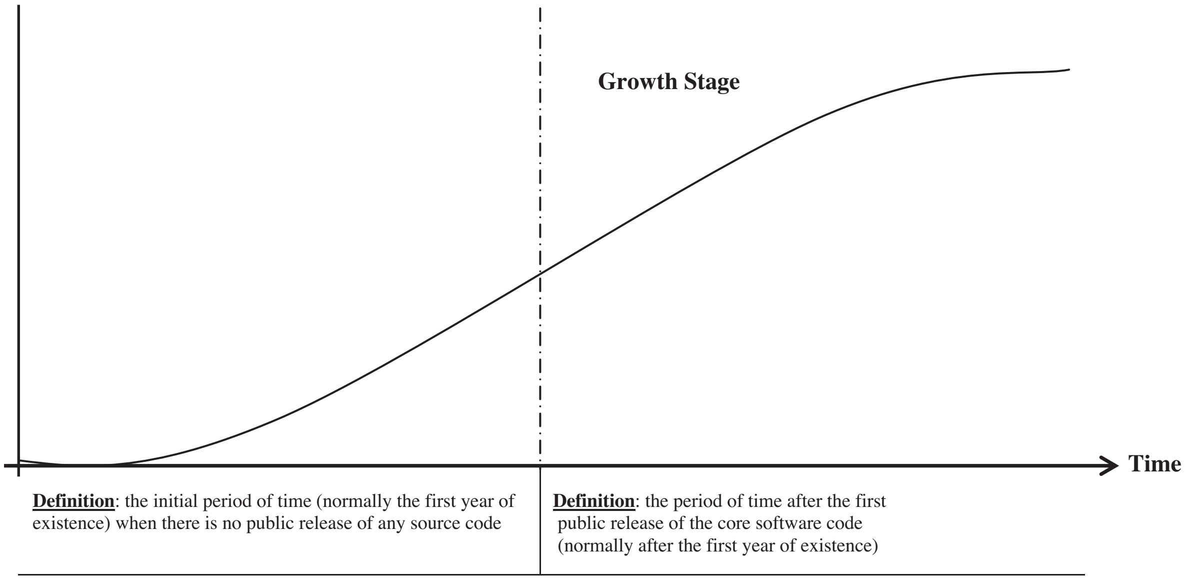
This study utilizes the conceptualization of Schweik's two-stage model and defines the first stage as the initial period of time when the team members of a particular FOSS project are collaborating on the first release of the core software code (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003). A FOSS project typically commences because one or a small group of like-minded programmers decide to tackle a shared but unfilled personal software challenge. Software development at the initial stage is typically confined within this small tightly knit group and not open to outside developers for feedback and contribution. Since FOSS projects that fail to produce the first release within the first year of existence are usually abandoned (Schweik, 2005), the priority for the project team at this stage is to develop and release an initial core piece of software for others to build upon. With the first release of software code being made available on the internet, the project will then have the opportunity to grow by attracting both developers and users. Thereafter, the project enters the growth stage, defined as the period after the first public release of the core software code. At this stage, the priority of the team shifts from focusing solely on the development of the software product to focusing on the growth of both the software product and the user base (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003). Figure 1 shows the FOSS lifecycle stages and their definitions.

A FOSS project faces different sets of challenges depending on which stage it resides in. At the initial stage, a FOSS project must overcome challenges associated with a lack of legitimacy. When a project is created, its value is unproven and the project initiator(s) needs to demonstrate its worthiness and viability. The need to get other people to see the value of the new project suggests the importance of generating legitimacy. Legitimacy is especially critical to FOSS projects at the initial stage since it is an important resource

Table 1 Research on the predictors of FOSS project outcomes.

	Predictors	Definition	Studies
Developer characteristics	User/developer participation effort*	Level of participation from the users/developers	1. Mockus et al. (2002) 2. Stewart and Gosain (2006)
	Service quality*	Developers' capability to promptly provide information, services, and solutions to reported problems	1. Lee et al. (2009) 2. Stewart and Gosain (2006)
	Leadership	A widely accepted leadership setting the project guidelines and driving the decision process	Bonaccorsi and Rossi (2003)
	Adherence to FOSS ideology	The extent to which individuals felt each ideological tenet was adhered to in his or her team	Stewart and Gosain (2006)
Software characteristics	License restrictiveness*	Whether the license of a FOSS software contains highly restrictive terms	1. Stewart et al. (2006) 2. Subramaniam et al. (2009) 3. Wu et al. (2007) 4. Comino et al. (2007)
	Targeted users*	Whether the FOSS software is targeted at general end users or tech-savvy end users	1. Stewart et al. (2006) 2. Subramaniam et al. (2009) 3. Wu et al. (2007) 4. Comino et al. (2007) 5. Chengalur-Smith and Sidorova (2003)
	Software modularity	The degree to which a FOSS community is able to decompose its project into modules and is able to coordinate between the modules based on established communication protocol	Bonaccorsi and Rossi (2003)
	Software quality	The software is useful and easy to use	Lee et al. (2009)
Community attributes	Organizational sponsorship	Whether a project is affiliated with an organization	Stewart et al. (2006)
	Financial support	Whether a project has donors for financial support	Wu et al. (2007)
	Trust	The extent to which a FOSS developer is confident in and willing to act on the basis of, the words, actions, and decisions of another developer	Stewart and Gosain (2006)
	Social network ties*	The number of direct and indirect ties a FOSS project has	1. Grewal et al. (2006) 2. Singh, Tan, and Mookerjee (2007) 3. Wu et al. (2007) 4. Chengalur-Smith and Sidorova (2003) 5. Hahn et al. (2008)
	Quality of social ties*	The extent to which a FOSS project is connected with other important FOSS projects	Grewal et al. (2006)
	Structural holes	The absence of direct ties between two projects	Singh, Tan, and Mookerjee (2007)
	Network externality	The degree to which one user of a proprietary software has on the value of the open source software	Bonaccorsi and Rossi (2003)

Note: The ones marked with * are tested in this paper.



Growth Stage
Time
Definition: the initial period of time (normally the first year of existence) when there is no public release of any source code
Definition: the period of time after the first public release of the core software code (normally after the first year of existence)

Figure 1 FOSS lifecycle stages and their definitions.

for gaining other valuable resources (Zimmerman & Zeitz, 2002) such as developers. Yet at the initial stage, there is no publically released software code and typically a limited, if not non-existent, track record for the assessment of a project's legitimacy. Faced with great uncertainty about the life chances and the quality prospects of a project, FOSS developers have to evaluate the project's legitimacy and decide whether to invest their time, effort, or other resources into the project based on an informal assessment of observable attributes that are thought to co-vary with the project's underlying but unknown or unobservable value. Thus, acquisition of legitimacy at the initial stage largely depends on the establishment of observable conditions that convey to the FOSS community a sense of the project's worthiness and strengthen their confidence in the project. At the growth stage, the projects have released their software code, have existed for a sufficient period, and established a track record, which permits the FOSS community to evaluate their desirability and viability. Thus, the value of the projects is observable at the growth stage and can be measured through unambiguous outcomes such as the ones investigated in this paper (e.g., adoption rate and technical improvements). To achieve survival at the growth stage, FOSS projects must establish conditions that will actually contribute to the measurable outcomes.

Further, projects at the initial stage are more likely to obtain help, support, and interest from FOSS idealistic developers (the idealistic). As the projects move into the growth stage, they can attract help and interest from mainstream FOSS developers as well as the idealistic. This is because developers consider the fit between the project and their motivational goals when participating in a FOSS project (Sen, Subramaniam, & Nelson, 2009). The idealistic are those who are largely motivated to contribute to FOSS projects by altruism and/or the FOSS ideologies (Sen et al., 2009) By contrast, mainstream developers tend to be motivated to contribute to FOSS projects by instrumental reasons such as reputation and career opportunities as well as altruism and FOSS ideologies (Sen et al., 2009). At the initial stage, the value and future prospect of the projects is typically uncertain and it is unclear if participation in these projects will generate status or recognition for the developers. Thus, the projects are more likely to get help and support from the idealistic who participate and contribute for the sake of the greater good. At the growth stage, the projects' value and future prospect are more certain. Projects with a good track record have the potential to attract not only the idealistic but also mainstream developers because such projects can provide the mainstream developers with good exposure to the FOSS community. This could ultimately lead to enhanced visibility and economic opportunities for the participating developers (Sen et al., 2009). Therefore, while aligning itself with the motivational goals of the idealistic is critical for a young FOSS project's survival, projects striving for survival at the growth stage must establish conditions best aligned with the motivational goals of the mainstream developers as well as the idealistic.

Recognizing that the challenges FOSS projects face differ at the initial and growth stages, researchers have suggested that outcome measures and their predictors may be intimately linked to the lifecycle stage of the projects (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003). This notion highlights the need to explicitly consider the contingent effect of lifecycle stage when identifying survival criteria and studying the survival factors of FOSS projects.

Measures of FOSS project survival for the initial and growth stage

In the literature, the widely used outcome measures for FOSS projects include process-level variables (e.g., user interest, number of developers, and team effectiveness in addressing user issues) and project-level variables (e.g., adoption rate and technical improvements). Nevertheless, these measures may be more applicable to projects that are at the growth stage than to those at the initial stage. Each life cycle stage may engender different priorities for a FOSS project team, suggesting that different measures of survival must be employed to reflect the differences in priorities (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003).

At the initial stage, the emphasis of a FOSS project team is to ensure that the project does not lose its momentum and produces its first release as quickly as possible. With no publically released source code, projects at this stage are generally not open to users or non-member developers for adoption, feedback, and technical contributions. Measures such as user interest, adoption rate, and technical improvements are hence less relevant at this stage. By contrast, measures such as the ability to remain active or to produce the first release of source code better reflect the project teams' priorities at this stage and thus give a better and an earlier indication of the projects' chance of survival at this period (Schweik, 2005; Schweik & English, 2007; Schweik & Semenov, 2003). After the first release is produced and the project evolves into the growth stage, the team will need to shift their focus to expanding the user base, enhancing software functionally, accelerating the adoption rate, and improving the user experience etc. Reflecting these priorities, user interest, adoption rate, and technical improvements can give a good indication of survival for FOSS projects at this stage.

Focusing on the project level outcome, this study uses the ability to remain alive on the hosting website and the ability to produce the first release of source code as the measures of survival at the initial stage and the adoption rate and technical improvements as the measures of survival at the growth stage.

Research hypotheses

Developer characteristics (user/developer participation and service quality)

User/developer participation is defined as the extent to which users/ developers are involved in the development of the software. Service quality captures the degree to which the project team satisfies the needs and requirements of the software users/developers through the addition of new features, patches, and fixing of bugs etc.

One major problem with many FOSS projects lies in their unreliable, or non-existent, software updates and technical support (Goode, 2005). At the initial stage, with a high level of user/developer participation and service quality, the project could send a signal to the FOSS community that its development team is accountable and trustworthy and is committed to providing reliable and professional updates and support. Researchers have suggested that organizations that are thought to be reliable, accountable, and trustworthy have higher chances of survival (Stuart, Hoang, & Hybels, 1999). Additionally, while effort does not guarantee survival, a high user/developer participation level indicates the devotion of the developers to the project and could be interpreted by the FOSS community as the first step towards a quality product. Service quality may function as an indicator of the depth of the project team's underlying software development and coordination capability because they are able to respond to users/developers' needs quickly. These signals facilitate young projects in their effort to acquire legitimacy, increasing their chances of survival at the initial stage.

H_{1a}. The level of FOSS projects' user/developer participation will positively impact their survival at the initial stage.

H_{1b}. The level of FOSS projects' service quality will positively impact their survival at the initial stage.

User/developer participation and service quality are also expected to positively impact a FOSS project's survival at the growth stage. In the general information systems literature, researchers have found that user/developer participation in the development of software ensures a more accurate and complete assessment of user/developer requirements and leads to higher system quality (Peppard, 2003). Higher quality in turn leads to a greater adoption rate (Chengalur-Smith & Sidorova, 2003). Participation from users creates a sense of system ownership among the users, leading to greater user loyalty and acceptance of the final software product. In the case of the FOSS development model, as there are no formal protocols for collecting user requirements, user participation enhances a FOSS project's capability to acquire users' knowledge and understand users' needs and requirements, leading to greater technical improvements. The level of user/developer participation could be an indicator of their devotion and support to the project and researchers have suggested that devotion and support from both core developers and users were critical for the success of Apache and Mozilla (Mockus et al., 2002). A higher level of service quality leads to greater user satisfaction, which in turn improves user acceptance of the software. Taken together, user/developer participation and service quality are expected to positively influence a project's survival (measured by technical improvements and users' acceptance of the software) at the growth stage.

H_{1c}. The level of FOSS projects' user/developer participation will positively impact their survival at the growth stage.

H_{1d}. The level of FOSS projects' service quality will positively impact their survival at the growth stage.

Software characteristics

License restrictiveness

FOSS projects could differ in the restrictiveness of their licenses, i.e., the degree of restrictions imposed on the users/developers to re-distribute software derived or modified from FOSS. Strong copy-left licenses are highly restrictive and require any subsequent or derivative software/programs to inherit the original license. Weak-copy-left licenses are less restrictive and only require subsequent software/programs to be licensed similarly. However, the modified software can be released under a different license under certain conditions. Non-copy-left licenses are non-restrictive and the subsequent or derivative work is not obliged to inherit the license of the original (Lerner & Tirole, 2005).

While there is consensus that the degree of license restrictiveness could affect the interest of users and developers in the project (Comino et al., 2007; Subramaniam et al., 2009; West, 2003), the nature of this impact remains a debate. Some argue that restrictive licenses help attract users/developers because they are more likely to attract ''idealistic'' and not for profit programmers (West, 2003). Others suggest that the restrictiveness of the licenses could negatively impact user/developer interest for two reasons. First, many users/developers, motivated by the need to fulfill their unique software requirements, prefer to retain the rights for reuse of the software code in the ways that best serve their needs (Stewart et al., 2006; Subramaniam et al., 2009). Second, given the lack of clarity in the interpretation of the terms imposed by the restrictive licenses, FOSS projects with restrictive licenses will raise concerns and uncertainties among many developers/users about software ownership and the legal implications of the license terms (Stewart et al., 2006; Subramaniam et al., 2009), negatively impacting developer/user interest.

This inconsistency in findings may be reconciled by the incorporation of ''project life cycle stages'', i.e., the positive or negative effects of license restrictiveness are contingent on the life cycle stage in which the project resides. As previously pointed out, at the initial stage, a project is more likely to obtain help, support, and interest from the FOSS idealistic and thus aligning itself with the motivational goals of the idealistic is critical for its survival. Restrictive licenses are better aligned with the idealistic's view of the FOSS world (Sen et al., 2009) and hence, projects released under such licenses are more likely to attract interest and support from the idealistic, leading to a higher chance of surviving at the initial stage. As the project moves into the growth stage, it must establish conditions best aligned with the motivational goals of the mainstream developers as well as the idealistic. At this stage, restrictive licenses could discourage mainstream developers/users from participating because such licenses impose constraints on mainstream developers' rights to reuse the software code in the ways that best serve their needs, negatively impacting survival at the growth stage.

H_{2a}. The restrictiveness of a FOSS project's license will positively impact its survival at the initial stage.

H_{2b}. The restrictiveness of a FOSS project's license will negatively impact its survival at the growth stage.

Targeted users

Research has indicated that FOSS developers are more inclined to contribute their technical expertise to more challenging and sophisticated projects because they desire to be intellectually stimulated, to learn, and/or to improve their reputation and status (Comino et al., 2007). This suggests that sophisticated FOSS projects targeting tech-savvy users are more likely to enjoy a higher status. At the initial stage when there is limited information for the assessment of the project's true value, the higher status associated with a technically sophisticated project could help the project attract the participation and contribution from talented developers. At the very least, it could enhance the legitimacy of the project by sending a signal to the FOSS community that the project is capable of attracting talented developers given its high status. This in turn positively impacts the project's likelihood of survival at this stage. As the project moves into the growth stage, the talented developers it attracted at the initial stage, together with the developers it might continue to attract will help the project achieve greater technical improvements. It is also more likely to be adopted because the majority of the FOSS adopters are currently tech-savvy end users such as software developers (Sohn & Mok, 2007).

H_{3a}. FOSS projects targeting tech-savvy users will have a higher likelihood of surviving at the initial stage than those targeting general end users.

H_{3b}. FOSS projects targeting tech-savvy users will have a higher likelihood of surviving at the growth stage than those targeting general end users.

Community characteristics

Network size

One type of social networks that has been found to impact FOSS success is the affiliation network formed through the overlapping memberships of the developers. Figure 2 is an illustration of an affiliation network in which projects A and B form external networks because developer 3 is a member of both projects. Developers 1, 2, and 3 form internal networks because they are affiliated with the same project A. One network property that has been recurrently found to positively affect FOSS project outcomes is ***network size*** (Grewal et al., 2006; Wu et al., 2007). In an affiliation network, a FOSS project develops an internal network, whose size is captured by the number of developers in the project, and an external network, whose size is captured by the total number of projects in which this particular project's developers are members.

At the initial stage, a FOSS project benefits from its large internal and external networks because these

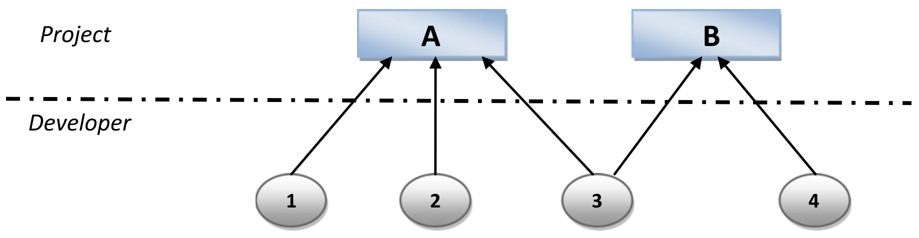


Figure 2 An example of FOSS affiliation network.

networks have the potential to enhance the project's legitimacy and ultimate chance of surviving. As previously pointed out, given its short track record and unproven value, an alternative approach for a FOSS project to prove its value is to enhance attributes that allow the FOSS community to indirectly infer the project's worthiness. One such attribute is the project's social network size. A large internal network could convey a positive message to the FOSS community that the project is perceived as important by the community because it can attract developers, thereby attaining confidence and viability in the FOSS community. This in turn could improve the project's chance of surviving. A large external network can be interpreted by many as an assurance that the focal project will receive support and help from its members' affiliated projects. This is because reciprocity is widely expected in the FOSS community and developers are more inclined to help and exchange information with those whom they have had prior interactions with (Kuk, 2006; von Hippel & von Krogh, 2003). This perceived assurance of help and support from peer projects improves the FOSS community's confidence in the focal project, which could ultimately improve the project's likelihood of surviving. Taken together, a FOSS project's internal and external network sizes enable the FOSS community members to indirectly infer the probability of its survival at the initial stage. This could help the project with a reasonably high-perceived probability of surviving legitimize its importance and viability and attract enough interest from FOSS user/developers to participate in it, thereby fostering actual survival at this stage.

H_{4a}. FOSS projects' internal network size will positively impact their survival at the initial stage.

H_{4b}. FOSS projects' external network size will positively impact their survival at the initial stage.

Nevertheless, the net positive effect of network size on FOSS performance is expected to diminish as the project evolves from the initial to the growth stage. As previously pointed out, projects at the growth stage are expected to demonstrate proven track records through solid and unambiguous performance. Thus, establishing conditions that actually improve their track record is necessary for FOSS project survival at this stage. Some may argue that a FOSS project's network size has intrinsic value that helps to enhance its performance. The logic for this argument is that with a larger internal network, the project team is able to spread complex software development tasks and cognitive strains over a larger number of developers, resulting in greater productivity and technical improvements (Grewal et al., 2006). The team is also more likely to produce high quality software products because there would be more eyeballs to search for and fix bugs (Raymond, 2001). Higher quality could ultimately translate into a greater adoption rate (Chengalur-Smith & Sidorova, 2003). Since advocacy is a widespread phenomenon in the FOSS community where developers work to convince their reference groups to use open-source software, a larger internal network size also implies a larger number of informal marketing staff promoting and campaigning for their products, resulting in a higher acceptance rate (Bonaccorsi & Rossi, 2003). Projects with a large external network have the potential to be exposed to unique sources of task related knowledge and information. Such unique sources can help project members elicit novel ideas and insights and better understand the needs and requirements of more diverse user and developer groups. Thus, a FOSS team with a large external network may achieve greater technical improvements. Members with multiple affiliations may also function as important channels for information dissemination. Their cross-membership may help increase the visibility of the project, leading to a higher acceptance rate.

Despite these potential advantages, a FOSS project's large internal and external networks could exert an unintended adverse effect on its performance, particularly at the growth stage. A project with a large internal network will have many developers. This could result in increased coordination and integration costs (Duysters & de Man, 2003), leading to a slower pace of code submission and software releases (slower technical improvements). Coordination and integration could be more challenging at the growth stage because projects at this stage typically have produced at least one publically released piece of source code and are open to non-member users/developers as well as member users/developers (Capiluppi et al., 2007; Schweik, 2005; Schweik & Semenov, 2003). Consequently, coordination and integration at the growth stage would involve a larger and more heterogeneous group of individuals. A large internal network, coupled with increased heterogeneity at the growth stage may lead to increased process loss, reduced productivity for a project because the project team would be less cohesive, and require more formal communication and coordination. Research on groups provides support for this argument as smaller teams have been found to be more cohesive and do not encounter the same coordination and communication problems that larger teams do (Smith et al., 1994). With a large external network size, a project will have developers working on too many projects simultaneously. This could divert the developers' focus and efforts and prevent them from engaging in intensive code submission, in-depth knowledge sharing, thorough feedback, code testing, and promoting for the focal project (Kuk, 2006), resulting in a low developer participation level, a low adoption rate, and slow technical improvements. Given these challenges, the net positive effect of network size on survival at the growth stage is expected to be negligible as the negative impacts may cancel out the positive influences.

Another reason that the net positive effects of network size on FOSS performance is expected to be negligible at the growth stage lies in the argument that the members rather than the size of the network matters. Researchers have found that only a small proportion of FOSS developers are heavy contributors (Kuk, 2006). Thus, a large internal network size does not guarantee that every member on the project team will contribute. Largely, it is the developers' devotion and participation level, rather than the size of a project's internal network that determines the level of code contribution and market campaigning, and ultimately the adoption rate and technical improvements. The conditioning effect of time on the performance impacts of social networks is also found in other contexts such as the internationalization of new ventures (Sasi & Arenius, 2008).

H_{4c}. FOSS projects' internal network size will not impact their survival at the growth stage.

H_{4d}. FOSS projects' external network size will not impact their survival at the growth stage.

Quality of external network

The survival of a FOSS project also depends on the quality of the network. When a project's developers are simultaneously affiliated with other successful projects, the project is said to have a high quality external network and is expected to have a higher likelihood of surviving compared to projects whose developers are affiliated with less successful projects. At the initial stage, because the track record of the FOSS project is rarely sufficient for the evaluation of its value and future prospect, the FOSS community will take into account the identities of its affiliates to resolve the uncertainties regarding its value. Lending support to this argument, social network theorists have long suggested that a social network tie implicitly transfers status between members of the tie and one's reputation is constructed in part by the identities of its associates (Blau, 1964). Similarly, a young FOSS project's status and future prospects may be greatly enhanced when its developers are affiliated with other successful or prestigious projects. If the project is connected to other successful projects, it will be perceived as more likely to survive because its ties are more likely to provide valuable and high-quality social capital compared to if it were connected to less successful projects. Ties to successful projects could also serve as an endorsement for the focal project. Such ties may facilitate young FOSS projects in their attempts to acquire legitimacy and other resources, which in turn helps the young projects improve their chance of surviving at the initial stage. In the business strategy and management literature, this reputation or legitimacy spillover effect has been observed to play a significant role in the survival of new firms (Stuart et al., 1999).

H_{5a} . The quality FOSS projects' external network will positively impact their survival at the initial stage.

The quality of a project's external network is also expected to exert a positive effect on the project's survival at the growth stage. Successful FOSS projects are more likely than the less successful ones to possess valuable and quality resources and capabilities including talented developers, a deep knowledge base, a large user base, and valuable FOSS project coordination experience and mechanisms etc. Hence, if a project's developers are affiliated with these successful projects, the magnitude and value of the resources and social capital available through the project's external network will be far greater than if this project's developers were affiliated with less successful projects. In the recent social network literature, there has been recognition of the significant impact of partners' resources and capabilities on one's performance (Zaheer & Bell, 2005). Studies have found that ties and alliances with innovative partners enhance a firm's performance (Baum, Calabrese, & Silverman, 2000). In the FOSS literature, researchers have also suggested that a FOSS project's performance is not only impacted by its network ties, but also by whom the project is connected to (Grewal et al., 2006). Clearly, the argument that FOSS projects' survival at the growth stage is a function of their external network quality is consistent with this line of research.

H_{5b} . The quality FOSS projects' external network will positively impact their survival at the growth stage.

Methodology

Data sample

The hypotheses are tested using data drawn from SF, the largest repository of open-source projects. SF provides FOSS developers with useful tools such as concurrent version systems, email lists, blogs, forums, source code repositories, and project activity statistics to control and manage their projects. SF has detailed data on over 300,000 projects and over 3 million registered users' activities, including data on project status, software development phase, project registration time, member developers, license, intended users, project category, number of bugs, patches and feature requested and solved, and project statistics. Using SF data for hypothesis testing has another advantage: all FOSS project managers have easy access to these data. Therefore, if the hypotheses are supported, ease of data access will enable FOSS project managers' to monitor and manage project risks and problems with relative ease. This is particularly valuable given the challenges FOSS project managers face including their lack of project management experience and the dearth of resources they are able to allocate for managing project.

At the time this research was conducted, only data from January 2005 to January 2010 was available. A stratified random sample of 2458 (the monthly average) was drawn from the start-up projects that were registered between January and March of 2005. Subsequent review indicated that 29 projects had missing data on development phase which was used as the proxy for technical improvements, 63 projects were listed on multiple websites and some websites did not keep track of the number of downloads for the projects (downloads was used as the proxy for adoption rate). In addition, 146 projects are not legitimate. For instance, based on the project description, it appears that some projects were listed by the project initiator(s) to test the account. Therefore, these 238 projects were excluded from the analyses and the final sample consisted of 2220 projects.

Dependent variables and their measures

Survival at the initial stage

Like Schweik and English (2007), the first year of a project's existence is defined as the initial stage because research suggests that projects that are not able to remain active or produce their first release of source code during the first year are generally abandoned. The ability to remain active on the hosting website and the ability to produce the first

Table 2 Classification of FOSS license types.

Strong copy-left	Weak copy-left	Non-copy-left
• Affero General Public License	• Common Development and Distribution License	• Academic Free License
• GNU General Public License	• eCos License	• Apache Software License
• License of Xinetd	• GNAT Modified GPL (GMGPL)	• Apple Public Source License (APSL)
• MITRE Collaborative Virtual Workspace License	• GNU Lesser General Public License	• BSD License
• Open Software License	• Interbase Public License	• Cryptix General License
• Ricoh Source Code Public License	• License of Guile	• Eclipse Public License
• Vita Nuova Liberal Source License	• License of Netscape Javascript	• EU DataGrid Software License
	• License of the run-time units of the GNU Ada compiler	• IBM Public License
	• License of Vim	• Intel Open Source License
	• Mozilla Public License (MPL)	• Jabber Open Source License
	• Nokia Open Source License	• LaTeX Project Public License
	• Open Publication License	• License of Perl
	• Public Domain	• License of ZLib
	• Sun Industry Standards Source License	• Lucent Public License
		• MIT/X Consortium License
		• Modified BSD License
		• OpenLDAP License
		• PHP License
		• Q Public License (QPL)
		• Standard ML of New Jersey Copyright License
		• University of Illinois/NCSA Open Source License
		• W3C License
		• Zope Public License (ZPL)

release of source code are used as two measures of survival at the initial stage. The first measure is operationalized as existence in the SF listing. Following Chengalur-Smith and Sidorova (2003), a project is considered as non-existing if it does not exist in SF activity ranking or if its project status is inactive or deleted. A search was also conducted to ensure that projects which moved out of SF and had their own websites created were not mistakenly identified as non-survivors. No such cases were found in the sample. Following Schweik and English (2007), the second measure is operationalized as the existence of publically released source code. Both measures are coded as dummy categorical variables with 1 for survivors and 0 for non-survivors.

Survival at the growth stage-adoption rate and technical improvement

SF provides data on the number of times a software project has been downloaded. This is used as the indicator for adoption rate. Projects listed on multiple websites were removed from the sample. This ensures that projects included in the analysis are distributed through a single channel and consequently the download measure can be used as a reasonable proxy for acceptance rate (Grewal et al., 2006). Because the mean and the standard deviation of the download measure are fairly large and its distribution is highly skewed, its natural logarithm (after adding unity to eliminate zeros) is used. The development of FOSS software typically proceeds through six phases: from planning phase to the release of intermediate pre-alpha, alpha, and beta versions and then up to the completion of production and mature versions. Thus, a FOSS project's evolution towards a stable and mature version is an indicator that the needs of the developers/users are effectively satisfied and can be used as a proxy for the technical improvements of the project (Comino et al. (2007). SF provides data on the current development phase a project has reached. This is used as the indicator for technical improvement (Comino et al., 2007).

Independent variables and their measures

User/developer participation

SF keeps data on the number of feature requests, new features, bug reporting, bug fixings, project tasks, and forum messages etc. submitted by users and developers. This data is used to approximate user/developer participation because it indicates the amount of time and effort users/developers devote to the project.

Service quality

SF provides data on the number of artifacts (bugs, patches, features, and services) and tasks created, reported, or requested by the developers/users and the number of artifacts and tasks solved by the team. Service quality is approximated by the percentage of requested artifacts and tasks that have been solved by the team as this percentage reflects the degree to which the team is able to satisfy developers/users' needs.

License restrictiveness and targeted users

These are approximated using SF data on a project's license type and target user segment(s). *License restrictiveness* is coded as 1 if the license terms have highly restrictive (e.g., strong copy-left) or restrictive provisions (e.g., weak copy-left) and 0 if otherwise (e.g., non-copy-left). Table 2 summarizes the classification scheme of the FOSS license

Table 3 Summary of the constructs, definitions, measures, and indicator variables used.

	Constructs	Definition	Measures	SourceForge.net Indicators
Dependent variables	Survival (initial stage)	A project's likelihood of survival during the first year of existence	Ability to remain active on hosting website	Existence in sourceforge.net activity ranking and project status set as active by the project administrator(s)
			Ability to produce the first release of source code	The existence of publically released source code
	Survival (growth stage)	A project's likelihood of survival after the first year of existence	Adoption rate	Number of downloads
			Technical improvements	Current development phase reached by a project
Independent variables	User/developer participation	The extent to which users/developers are involved in the development of the software	User/developer involvement in feature addition, bug reporting and fixing etc.	The number of feature requests, new features, bug reporting, bug fixings, project tasks, and forum messages etc. submitted by users and developers
	Service quality	The degree to which the project team satisfies the needs and requirements of the software users/developers through the addition of new features, patches, and fixing of bugs	The degree to which users' needs on bugs, patches, features, and services are satisfied	The total number of solved artifacts (bugs, patches, features, and services requested) and tasks divided by total number of artifacts and tasks
	License restrictiveness	The degree of restrictions imposed on the users/developers to re-distribute software derived or modified from FOSS	The classification of license types into strong copy left, weak copy left, and non copy left	Coded as 1 if the license terms have highly restrictive (e.g., strong copy-left) or restrictive provisions (e.g., weak copy-left) and 0 if otherwise (e.g., non-copy-left)
	Targeted users	Whether a project is targeted at the tech-savvy users	Whether a project is targeted at software developers and systems administrators	Coded as 1 if the project is geared towards software developers and systems administrators and 0 if otherwise
	Internal network size	The size of the network formed among developers within the project	The number of member developers for the project team	The number of member developers for a particular project
	External network size	The size of the network formed between the project's own members and members of other FOSS projects	The total number of projects the focal project's developers are affiliated with	The average number of projects the focal project's developers are affiliated with
	Quality of external network	The degree to which a project's developers are simultaneously affiliated with other successful projects	SourceForge Ranking of the affiliated projects	The average SourceForge Ranking of the affiliated projects

types. *Targeted user* is coded as 1 if the project is geared towards software developers and systems administrators and 0 if otherwise.

In analyzing the network size of the 2220 projects' social network, snowball sampling is used. This procedure starts with the initial set of 2220 projects (focal projects), then traces down all the connections (the second set) of the focal projects, and then further traces down all the connections of the second set of projects until no new connected projects could be traced.

Internal network size of a project

This is approximated by using SF data on the number of member developers a project has.

External network size of a project

This is approximated by using SF data on the total number of projects this focal project's developers are affiliated with. The average of this measure for all the member developers is used as the external network size measure at the project level.

Table 4 Correlations (initial stage: ability to remain active).

		Mean	SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	Ability to remain active	0.50	0.50	1																	
2	Developer experience	11.01	16.31	0.23**	1																
3	Project age	1763.34	8.27	0.14**	0.04	1															
4	Communication	0.04	0.19	0.16**	0.06	0.07*	1														
5	Games	0.04	0.18	0.16**	0.00	−0.01	0.01	1													
6	Internet	0.06	0.24	0.23**	0.05	0.06*	0.15**	−0.01	1												
7	Multimedia	0.04	0.18	0.18**	0.06*	0.04	0.01	−0.01	0.04	1											
8	Software development	0.06	0.24	0.18**	−0.01	0.08*	0.11**	0.10**	0.05	0.04	1										
9	Systems	0.04	0.19	0.15**	0.07*	0.07*	0.11**	−0.01	0.13**	0.01	0.11**	1									
10	Office	0.02	0.15	0.13**	0.03	0.01	0.15**	−0.03	0.11**	0.00	0.06*	0.3	1								
11	Science	0.03	0.18	0.14**	0.01	0.05	−0.01	−0.03	0.00	−0.03	0.11**	−0.01	0.00	1							
12	Participation	1.35	2.72	0.37**	0.09**	0.10**	0.10**	0.12**	0.10**	0.17**	0.09**	0.04	0.20**	0.06*	1						
13	Service quality	−0.96	0.23	0.13**	0.05	0.04	0.01	0.04	0.07*	0.09**	0.06*	−0.01	0.08**	0.07*	0.59**	1					
14	Restrictiveness	0.87	0.34	0.07**	0.00	0.07*	0.05	.0.00	0.09**	0.02	−0.01	0.02	0.04	0.02	0.00	−0.03	1				
15	Target users	0.16	0.37	0.37**	0.11**	0.10**	0.18**	0.13**	0.37**	0.18**	0.54**	0.33**	0.14**	0.19**	0.22**	0.11**	0.03	1			
16	Internal network size	1.47	1.38	0.32**	0.07*	−0.01	0.04	0.10**	0.07*	0.00	0.07*	0.03	0.13**	0.08**	0.36**	0.21**	−0.02	0.14	1		
17	External network size	2.08	1.90	0.25**	0.43**	0.02	0.07*	0.02	0.08**	0.04	0.08	0.07*	0.03	0.04	0.04	−0.03	0.04	0.14	0.02	1	
18	Quality of external network	779494899.42	267483712.25	0.46**	0.12**	0.11**	0.12**	0.13**	0.24**	0.20**	0.18**	0.15**	0.14**	0.08**	0.27**	0.20**	0.04	0.38	0.10	0.15	1

N = 1105.

** Correlation is significant at the 0.01 level (2-tailed).

* Correlation is significant at the 0.05 level (2-tailed).

Table 5 Correlations (initial stage: first release).

		Mean	SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	First release	0.50	0.50	1																	
2	Developer experience	11.65	16.49	0.10**	1																
3	Project age	1762.73	8.15	0.00	0.02	1															
4	Communication	0.05	0.23	0.11**	0.01	0.00	1														
5	Games	0.05	0.23	0.11**	0.01	0.02	−0.01	1													
6	Internet	0.09	0.29	0.15**	0.03	0.03	0.13**	0.02	1												
7	Multimedia	0.05	0.22	0.14**	0.06*	0.03	−0.01	0.01	0.00	1											
8	Software development	0.09	0.29	0.14**	0.01	0.04	0.01	0.01	0.04	0.05*	1										
9	Systems	0.07	0.26	0.16**	0.02	0.04	0.03	−0.05*	0.04	−0.02	0.07**	1									
10	Office	0.03	0.18	0.06*	0.02	−0.01	0.06*	−0.04	−0.07**	−0.03	0.02	−0.02	1								
11	Science	0.04	0.19	0.09**	0.00	0.02	−0.02	0.01	0.00	0.04	0.13**	0.02	−0.02	1							
12	Participation	1.94	3.09	0.20**	0.02	0.06**	0.06*	0.08**	0.09**	0.10**	0.15**	0.06*	0.06**	0.05*	1						
13	Service quality	−0.93	0.30	0.18**	0.04	0.07**	0.01	0.06*	0.05*	0.05*	0.09**	0.03	0.04	0.05*	0.50**	1					
14	Restrictiveness	0.86	0.34	0.04	−0.05*	0.02	0.05*	0.00	0.03	0.02	−0.03	0.02	0.02	0.03	−0.04	−0.02	1				
15	Target users	0.15	0.36	0.22**	0.02	0.05	0.01	−0.03	0.05*	0.02	0.75**	0.66**	0.01	0.07**	0.13**	0.08**	0.00	1			
16	Internal network size	1.61	1.60	0.14**	0.00	0.00	0.03	0.03	0.04	−0.01	0.05*	0.01	0.07**	0.04	0.29**	0.18**	−0.02	0.03	1		
17	External network size	2.19	1.99	0.14**	0.42**	0.02	0.02	0.00	0.05*	0.03	0.04	0.04	0.01	0.02	0.00	−0.03	−0.02	0.06**	0.00	1	
18	Quality of external network	873192651.37	317195130.25	0.66**	0.04	0.09**	0.14**	0.10**	0.15**	0.14**	0.13**	0.18**	0.10**	0.08**	0.20**	0.22**	0.04	0.24**	0.01	0.07**	1

N = 1878.

** Correlation is significant at the 0.01 level (2-tailed).

* Correlation is significant at the 0.05 level (2-tailed).

Table 6 Correlations (growth stage: adoption rate and technical improvement).

		Mean	SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	(Log)download	12399.29	280033.03	1																		
2	Development phase	3.20	1.49	0.44**	1																	
3	Development experience	12.64	16.81	0.07**	0.07	1																
4	Project age	1762.71	8.04	−0.01	0.01	0.02	1															
5	Communication	0.07	0.26	0.05	0.07**	0.00	0.02	1														
6	Games	0.06	0.25	0.07**	0.03	0.02	0.00	−0.03	1													
7	Internet	0.11	0.31	0.07**	0.09**	0.01	0.06*	0.10**	−0.01	1												
8	Multimedia	0.06	0.24	0.14**	0.09	0.06	0.02	−0.02	−0.01	−0.01	1											
9	Software development	0.10	0.30	0.09**	0.08**	−0.02	0.04	−0.03	−0.02	0.02	0.03	1										
10	Systems	0.09	0.28	0.12**	0.12**	0.01	0.04	0.03	−0.06**	0.04	−0.03	0.05*	1									
11	Office	0.04	0.20	0.07**	0.04	0.00	0.01	0.05*	−0.05*	0.04	−0.03	0.00	−0.01	1								
12	Science	0.05	0.21	0.04	0.06*	0.01	0.03	−0.03	0.00	−0.01	0.03	0.12**	0.02	−0.02	1							
13	Participation	21.10	157.74	0.23**	0.10**	0.03	−0.02	−0.01	0.00	0.03	−0.01	0.03	0.13**	0.04	−0.01	1						
14	Service quality	0.08	0.23	0.41**	0.22**	0.04	−0.01	0.03	−0.01	0.01	0.01	0.02	−0.01	0.00	0.05	0.19**	1					
15	Restrictiveness	0.88	0.33	0.00	−0.01	−0.04	0.02	0.03	0.00	0.03	0.01	−0.05	0.02	0.02	0.02	−0.02	−0.02	1				
16	Target users	0.29	0.46	0.21**	0.20**	0.06*	0.05	0.14**	0.07**	0.31**	0.11**	0.49**	0.30**	0.06*	0.11	0.03	0.08**	−0.02	1			
17	Internal network size	1.79	1.85	0.15**	0.05*	0.00	0.00	0.02	0.01	−0.01	−0.02	0.01	−0.02	0.05*	0.01	0.26**	0.20**	−0.01	0.02	1		
18	External network size	1.15	1.90	0.14**	0.11**	0.38**	0.03	0.01	−0.01	0.03	0.03	0.02	0.03	−0.01	0.03	0.00	0.03	−0.01	0.06*	−0.03	1	
19	Quality of external network	1789274.00	3439643.46	0.52**	0.23**	0.23**	0.00	0.01	0.04	0.04	0.06*	0.05*	0.04	0.05*	0.04	0.30**	0.35**	0.01	0.11**	0.028**	0.37**	1

N = 1667.

** Correlation is significant at the 0.01 level (2-tailed).

* Correlation is significant at the 0.05 level (2-tailed).

Table 7 Regression results.

	Initial stage						Growth stage					
	Ability to remain active			First release			Adoption rate			Technical improvement		
Variables	B	S.E.	Sig	B	S.E.	Sig	B	S.E.	Sig	B	S.E.	Sig
Developer experience	0.01	0.01	0.03^{**}	0.01	0.00	0.03^{**}	0.00	0.00	0.03^{**}	0.00	0.00	0.91
Project age	0.03	0.01	0.03^{**}	0.03	0.01	0.00^{***}	0.00	0.00	0.27	0.00	0.01	0.90
Games	0.73	0.62	0.24	0.65	0.31	0.04^{**}	0.37	0.13	0.00^{***}	0.21	0.18	0.25
Internet	0.80	0.61	0.19	0.56	0.26	0.03^{**}	0.13	0.11	0.24	0.26	0.15	0.09^{*}
Multimedia	1.52	0.86	0.08^{*}	0.88	0.33	0.01^{**}	0.71	0.13	0.00^{***}	0.48	0.19	0.01^{***}
Control variables												
Developer experience	0.01	0.01	0.03^{**}	0.01	0.00	0.03^{**}	0.00	0.00	0.03^{**}	0.00	0.00	0.91
Project age	0.03	0.01	0.03^{**}	0.03	0.01	0.00^{***}	0.00	0.00	0.27	0.00	0.01	0.90
Communication	0.46	0.58	0.42	−0.03	0.35	0.94	0.00	0.00	0.27	0.00	0.01	0.05^{*}
Games	0.73	0.62	0.24	0.65	0.31	0.04^{**}	0.37	0.13	0.00^{***}	0.21	0.18	0.25
Internet	0.80	0.61	0.19	0.56	0.26	0.03^{**}	0.13	0.11	0.24	0.26	0.15	0.09^{*}
Multimedia	1.52	0.86	0.08^{*}	0.88	0.33	0.01^{**}	0.71	0.13	0.00^{***}	0.48	0.19	0.01^{***}
Software development	−0.61	0.57	0.29	0.48	0.26	0.06^{*}	0.07	0.12	0.57	0.06	0.17	0.72
Systems	0.08	0.57	0.89	0.56	0.31	0.07^{*}	0.40	0.12	0.00^{***}	0.48	0.17	0.00^{***}
Office	−1.25	0.81	0.12	−0.71	0.48	0.14	0.32	0.16	0.05^{**}	0.32	0.23	0.15
Science	−0.08	0.58	0.89	0.16	0.39	0.68	−0.02	0.15	0.90	0.25	0.21	0.24
Developer characteristics												
Participation	1.12	0.10	0.00^{***}	0.12	0.06	0.03^{***}	0.00	0.00	0.01^{***}	0.00	0.00	0.10^{*}
Service quality	2.47	0.64	0.00^{***}	3.13	0.42	0.00^{***}	1.70	0.14	0.00^{***}	1.26	0.21	0.00^{***}
Software characteristics												
Restrictiveness	0.51	0.31	0.10^{*}	0.32	0.20	0.10^{*}	−0.01	0.09	0.92	−0.10	0.13	0.47
Targeted users	0.85	0.47	0.07^{*}				0.33	0.09	0.00^{***}	0.41	0.13	0.00^{***}
Community characteristics												
Internal network size	1.19	0.21	0.00^{***}	0.22	0.05	0.00^{***}	−0.02	0.02	0.21	−0.02	0.03	0.38
External network size	0.26	0.07	0.00^{***}	0.13	0.04	0.00^{***}	−0.02	0.02	0.53	0.03	0.03	0.20
Quality of external network	0.00	0.00	0.00^{***}	0.00	0.00	0.00^{***}	−0.01	0.00	0.00^{***}	0.00	0.00	0.00^{***}
Constant	−54.05	20.38	0.01^{***}	51.99	14358	0.00^{***}	8.47	6.81	0.21			
S_0								0		−5.82	9.63	0.55
S_1										−2.18	9.63	0.82
S_2										−1.46	9.63	0.88
S_3										−0.64	9.63	0.95
S_4										0.52	9.63	0.96
S_5										3.83	9.63	0.69
Model Summary	R^2 (Nagelkerke) = 68.2%			R^2 (Nagelkerke) = 64.4%			Adjusted R^2 = 37%			Pseudo-R^2 = 11.9%		
	Percentage correct = 89.1%			Percentage correct = 84%			F = 58.6		0.00^{***}	χ^2 = 203.38		0.00

N = 1105 for ability to remain active, N = 1878 for first release, & N = 1667 for adoption rate & technical improvement.
S_0—S_5 are the estimated thresholds delimiting the steps needed by a project to move to the next phase of development.
*** $p < 0.01$.
** $p < 0.05$.
* $p < 0.1$.

Quality of external network
SF uses a scoring system to rank all the projects listed on its website based on developers/users' interests in the project. This ranking system is used as a proxy for the quality of the external network because when a project's developers are affiliated with higher-ranking projects, this project is said to have higher quality external network. For a particular project, the quality of the external network is the average ranking of its affiliated projects.

Lagged independent variables are used in the analysis given the possible existence of simultaneity between dependent and independent variables. For example, developers' participation level may influence project outcome while project outcome is likely to feed back to encourage or discourage developers' participation level. Therefore, lagged rather than contemporaneous independent variables are specified in an attempt to alleviate the possibility of mutual dependency. Second, the specification of lagged

Table 8 Summary of the Hypotheses and the results.

Hypotheses	Supported?
H_{1a}. The level of FOSS projects' user/developer participation will positively impact their survival at the initial stage.	Yes
H_{1b}. The level of FOSS projects' service quality will positively impact their survival at the initial stage.	Yes
H_{1c}. The level of FOSS projects' user/developer participation will positively impact their survival at the growth stage.	Supported for Adoption Rate, Marginally supported for Technical Improvement
H_{1d}. The level of FOSS projects' service quality will positively impact their survival at the growth stage.	Yes
H_{2a}. The restrictiveness of a FOSS project's license will positively impact its survival at the initial stage.	Marginally supported
H_{2b}. The restrictiveness of a FOSS project's license will negatively impact its survival at the growth stage.	No
H_{3a}. FOSS projects targeting tech-savvy users will have a higher likelihood of surviving at the initial stage than those targeting end users.	Marginally supported for the ability to remain active, dropped from analysis for the ability to produce the first release
H_{3b}. FOSS projects targeting tech-savvy users will have a higher likelihood of surviving at the growth stage than those targeting end users.	Yes
H_{4a}. FOSS projects' internal network size will positively impact their survival at the initial stage.	Yes
H_{4b}. FOSS projects' external network size will positively impact their survival at the initial stage.	Yes
H_{4c}. FOSS projects' internal network size will not impact their survival at the growth stage.	Yes
H_{4d}. FOSS projects' external network size will not impact their survival at the growth stage.	Yes
H_{5a}. The quality FOSS projects' external network will positively impact their survival at the initial stage.	Yes
H_{5b}. The quality FOSS projects' external network will positively impact their survival at the growth stage.	Yes

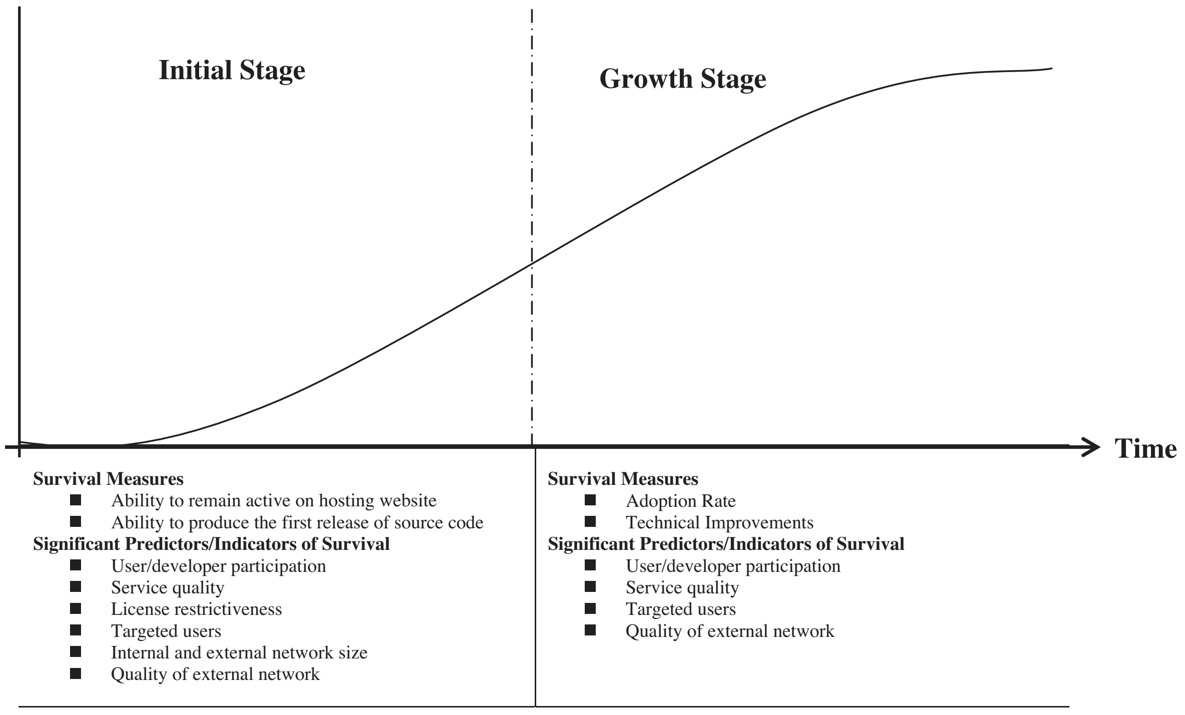

Initial Stage
Growth Stage
Time
Survival Measures
Ability to remain active on hosting website
Ability to produce the first release of source code
Significant Predictors/Indicators of Survival
User/developer participation
Service quality
License restrictiveness
Targeted users
Internal and external network size
Quality of external network
Survival Measures
Adoption Rate
Technical Improvements
Significant Predictors/Indicators of Survival
User/developer participation
Service quality
Targeted users
Quality of external network

Figure 3 The survival measures and significant indicators.

independent variables is also aligned with the rationale that the impacts of independent variables might require a certain time lag before they manifest themselves. Hence, a one-year lag is used for the independent variables.

Control variables and their measures

Control variables include developer experience, project age, and project category due to their potential influence on the survival of the software product. Developer experience is measured by the duration of time (in terms of month) a developer has been a registered developer on SF. The average of this measure for all member developers of a particular project is used as the developer experience measure at the project level. Project age is measured by the number of days a project is registered on SF. Project category is operationalized using the categorization scheme used by SF (e.g., communication, games, internet etc.).

Table 3 summarizes the constructs, their definitions, indicator variables, and measures used in this study.

Analyses and results

When *the ability to remain active on SF* is used as the survival measure at the initial stage, there are 553 non-survivors and 1667 survivors. To minimize the bias of unequal group size, 552 survivors are randomly selected from the 1667 survivor pool and 1105 projects remain in the final sample. Table 4 displays the means, standard deviations, and correlation coefficients of all the variables. Collinearity diagnostics indicate that multi-collinearity is not a concern. *The ability to remain active* is positively and significantly correlated with all the independent variables. When *the ability to produce the first release* is used as the survival measure at the initial stage, there are 939 survivors and 1281 non-survivors. To minimize the bias of unequal group size, 939 non-survivors are randomly selected from the total non-survivor pool and the final sample consists of 1878 projects. Table 5 reports the means, standard deviations, and correlation coefficients of all the variables. Collinearity diagnostics indicate that the multi-collinearity problem exists between *software development* and *targeted users*. Therefore, *target users* is dropped from the subsequent logistic analysis. *The ability to produce the first release* is positively and significantly correlated with all the independent variables.

In this research, adoption rate and technical improvements are used as survival measures at the growth stage. Therefore, only projects that survived the initial stage are included in the analyses. The final sample consists of 1667 projects that remain active on SF and the decision to include the projects that fail to produce the first release but remain active arises from the consideration that these projects may not be completely abandoned by their teams.[1] Table 6 displays the means, standard deviations, and correlation coefficients of all the variables. Except for license restrictiveness, all the independent variables are positively and significantly correlated with both dependent variables. Collinearity diagnostics indicate that multi-collinearity is not a concern.

The effect of the independent variables on the likelihood that a FOSS project will survive at the initial stage is tested using logistic regression. As is shown in Table 7, the overall models account for 68.2% (89.1% correctly predicted) of the variance in the ability to remain active and 64.4% of the variance in the ability to produce the first release (84% correctly predicted). With the exception *of license restrictiveness* and *targeted users*, all the independent variables positively and significantly impact the ability to remain active ($p < 0.01$) and the ability to produce the first release ($p < 0.05$ for participation and $p < 0.01$ for the other independent variables). *License restrictiveness* is only marginally and positively related to both dependent variables ($p < 0.1$). *Targeted users* is marginally and positively related to the ability to remain active ($p < 0.1$) but is dropped from the model for the ability to produce the first release due to multi- collinearity. The hypotheses and their results are summarized in Table 8. For the control variables, developer experience, project age, and project category are all found to impact survival at the initial age, particularly the ability to produce the first release.

The effect of the independent variables on the adoption rate is tested using ordinary least squares (OLS) regression. As shown in Table 7, the overall model is significant and accounted for approximately 37% of the variance in the adoption rate. When the effect of the independent variables on technical achievements is tested, because the dependent variable is discrete and ordinal, an univariate ordered probit model is used (Comino et al., 2007). As shown in Table 7 the goodness of fit χ^2 is 203.38 ($p < 0.01$), indicating that the predicated probabilities in the six development phases are equal to the true ones. The Pseudo-R^2 (11.9%) is higher than that of prior studies such as Comino et al.'s (2007) study where $R^2 = 2.41\%$.

Hypothesis 1c predicts that user/developer participation will continue to exert a positive influence on survival at the growth stage. This hypothesis is only marginally partially supported as the coefficient of user/developer participation is significant for adoption rate ($p < 0.01$), but only marginally significant for technical improvements ($p < 0.1$). Confirming H_{1d}, the positive and significant coefficients of service quality for both dependent variables ($p < 0.01$) suggest that FOSS projects with a higher level of service quality are more likely to be adopted and achieve a higher level of technical improvement. Hypothesis 2b which states that license restrictiveness will negatively impact survival at the growth stage is not supported because the coefficients of license restrictiveness for both adoption rate and technical improvements are not significant. The coefficients of targeted users are significant for both adoption rate and technical improvements, confirming H_{3b} that posits that projects targeting tech-savvy users are more likely to survive at the growth stage. Hypotheses 4c and 4d respectively predict that internal and external network sizes will not impact survival at the growth stage. As shown in Table 7, the coefficients of internal and external network sizes are not significant for both adoption rate and technical improve-

[1] To confirm the robustness of the results, additional analysis is performed by excluding the projects that are not able to produce a first release in the first year of their existence. The results are largely consistent with the one reported in Table 7 with the exception that internal network size is found to positively impact the adoption rate.

ment. Thus, H_{4c} and H_{4d} are supported. Supporting Hypothesis 5b, quality of external network was found to continue to exert a positive effect on the adoption rate and technical improvement ($p < 0.01$), suggesting that FOSS projects connected with high quality projects are more likely to survive at the growth stage.

Discussion, conclusion, and future research

The ability to identify key survival factors is of significant interest to the FOSS community as well as the firms that intend to learn from the FOSS model. Although several studies have investigated the determinants of FOSS project outcomes, they take a static view and assume that the effect of a factor on FOSS project outcomes remains stable throughout the life of the project. The purpose of this study, therefore, is to incorporate the concept of project life stages and theoretically and empirically explore the possibility that the criticality of a survival factor is contingent upon the life cycle stage (initial vs. growth) in which the project resides.

The results support the general argument of this study: as FOSS projects evolve so do the survival factors. The results indicate that user/developer participation and service quality are associated with survival at both stages. This finding highlights the importance of high quality developers who are devoted to participating in the project development activity and providing quality service to the users. License restrictiveness is found to be positively (although marginally) associated with survival at the initial stage, but it does not exert any impact on survival at the growth stage. This finding is intriguing, because it may reconcile the inconsistent results in prior studies on the direction of the effect that license restrictiveness exerts on FOSS performance (Comino et al., 2007; Subramaniam et al., 2009; West, 2003). It is possible that license restrictiveness plays a positive role only at certain stages of a project's life cycle and plays a minimal or negative role at other stages. Projects targeted at technical users have a higher likelihood of surviving at both stages, indicating that technically sophisticated users are still the niche market for FOSS applications. While internal and external network sizes appear to be significantly positively associated with FOSS survival at the initial stage, their net positive effects on survival at the growth stage diminish and become negligible. By contrast, the quality of an external network positively impacts survival at both the initial and growth stage. Although this finding confirms the importance of social networks, particularly the quality of the network, in shaping FOSS outcomes (Grewal et al., 2006; Wu et al., 2007), it also cautions that advantages stemmed from network sizes are not sustainable over long time periods. Figure 3 summarizes these findings.

This research adds to the existing literature in several respects. First, previous studies typically base their results on anecdotal evidence, case studies, or a few projects. This paper bases its analysis on a large dataset and notably reduces the risk of sample bias. Second, while the importance of identifying warning indicators for assessing project survival and success has been widely recognized in the project management literature, little work has been done in the FOSS domain. FOSS projects differ from traditional projects in significant ways that make many of the warning indicators commonly recommended for traditional projects inapplicable to the FOSS context. Many of the factors identified in this study reasonably predict survival and they can hence function as warning indications for FOSS project managers to assess the project's survival chances. Knowing and paying attention to such indicators helps project managers diagnose likely problems, take corrective actions, and consequently improve the probability of the project surviving. Third, unlike much of the prior work that mostly focuses on projects at the growth stage, this research investigates survival at both the initial and growth stages. Focusing on projects that are at the initial stage enables FOSS project managers to identify problems at the earliest possible stage, which is critical for project redirection and subsequent project survival. Finally, the importance of life cycle stage found in FOSS project survival is consistent with the findings of prior project management literature (Hoegl & Weinkauf, 2005; Pinto & Prescott, 1988). For the research community, this result demonstrates that a FOSS project outcome should be examined in its dynamic context. Specifically, time effects are important in FOSS survival and they should not be ignored in future studies.

The results also suggest several practical implications. First, the results highlight the importance of both human and social capital. While possessing developers/users who invest a substantial amount of time participating in the project's development tasks and providing high quality services is critical to the survival of FOSS projects, nurturing social networks is equally important. Research has suggested that the existence, strength, and quality of prior collaborative relations positively impact the probability that a FOSS project will attract more developers (Hahn, Moon, & Zhang, 2008). Thus, in order to rally developers to contribute to a FOSS project, its initiator(s) will need to develop high-quality relationships with other FOSS developers, even before a project is registered on the hosting website. Among other means, actively contributing to existing FOSS projects (Hahn et al., 2008; Kuk, 2006), strategically engaging in knowledge sharing with other developers (Kuk, 2006), abiding by the norms and the ideologies of the FOSS community (Bergquist & Ljungberg, 2001; Stewart & Gosain, 2006), and building social trust with other developers (Stewart & Gosain, 2006) have been documented as effective approaches that lead to high-quality and positive collaborative relationships. Second, building a large social network is not as important as building a quality network. Third, this research demonstrates that the strategic advantages offered by network size and license restrictiveness in the initial stage can dissipate over time. Consequently, FOSS projects should constantly reinvent their strategies and operations in search of sustainable competitive advantages. Finally, as user/developer participation, service quality, internal and external network size, and quality of external network have been found to reliably predict survival at the initial stage, a weak showing in these measures alerts early signs of trouble. Management should take heed to these signs, assess the probability of the project surviving, and decide whether to terminate the project or to attempt to redirect it and turn it into success. The value of early problem detection and assessment is clear when the project can be turned around. Nevertheless, the ability to recognize such early trouble signs will still be of substantial value for the participating developers even when they decide to terminate the project. This is because the developers can shift their focus to projects that have greater potential and such projects can provide the developers with better visibility in the FOSS community, leading to enhanced reputation, status, and career opportunities.

However, this study has some limitations that should be recognized. The importance of life cycle stage suggests that if we are ever to understand what leads to FOSS project survival, we must consider the context as a critical contingency factor for determining what critical factors work under a particular situation. Only life cycle stage was considered here. Future research may wish to explore other important contextual contingencies. Further, the sample was categorized into those targeting technically advanced users and those targeting ordinary users. Given the significant impact of targeted audience, future research may consider exploring this variable in more detail and perhaps categorize the projects based on programming language or operating systems. Other interesting results might emerge based on different classification schemes. Finally, while this research uses a large and well-recognized data source, it may suffer from the common method variance bias like all studies that use a single data source. Common method variance bias may occur when data representing the dependent variables and independent variables is collected using similar methodologies. For instance, individual developers tend to behave in specific ways (users/developers that devote a substantial amount of time to develop the software and provide quality services tend to download and use the software more frequently). Hence, linear correlation may be inflated. While the use of lagged independent variables in this research may help reduce common method variance, perhaps future research can mitigate this issue by collecting independent and dependent variables from separate sources. For instance, future researcher could use SF data measuring independent constructs and compare it with subjective questionnaire items measuring perceptions on the survival of a FOSS project.

References

Baum, J. A. C., Calabrese, T., & Silverman, B. S. (2000). Don't go it alone: Alliance network composition and startups' performance in Canadian biotechnology. *Strategic Management Journal, 21*, 267–294.

Bergquist, M., & Ljungberg, J. (2001). The power of gifts: Organizing social relationships in open source communities. *Information Systems Journal, 11*, 305–320.

Blau, P. (1964). *Exchange and Power in Social Life*. New York: Wiley.

Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed. *Research Policy, 32*, 1243–1258.

Capiluppi, A., González-Barahona, J. M., Herraiz, I., & Robles, G. (2007). Adapting the ''staged model for software evolution'' to free/libre/open source software. *IWPSE '07 ninth international workshop on principles of software evolution: In conjunction with the 6th ESEC/FSE joint meeting.*

Chengalur-Smith, S., & Sidorova, A. (2003). Survival of open-source projects: A population ecology perspective. *ICIS 2003 proceedings*. Paper 66.

Comino, S., Manenti, F. M., & Parisi, M. L. (2007). From planning to mature: On the determinants of open source take off. *Research Policy, 36*, 1575–1586.

Duysters, G., & de Man, A. (2003). Transitory alliances: an instrument for surviving turbulent industries? *R&D Management, 33*, 49–58.

Goode, S. (2005). Something for nothing: Management rejection of open source software in Australia's top firms. *Information & Management*, 669–681.

Grewal, R., Lilien, G. L., & Mallapragada, G. (2006). Location, location, location: How network embeddedness affects project success in open source systems. *Management Science, 52*, 1043–1056.

Hahn, J., Moon, J. Y., & 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.

Hoegl, M., & Weinkauf, K. (2005). Managing task interdependencies in multi-team projects: A longitudinal study. *Journal of Management Studies, 42*, 1287–1308.

Kappelman, L. A., McKeeman, R., & Zhang, L. (2006). Early warning signs of IT project failure: The dominant dozen. *Information Systems Management, 23*(4), 31–36.

Kuk, G. (2006). Strategic interaction and knowledge sharing in the KDE developer mailing list. *Management Science, 52*, 1031–1042.

Lee, S.-Y. T., Kim, H. W., & Gupta, S. (2009). Measuring open source software success. *Omega, 37*, 426–438.

Lerner, J., & Tirole, J. (2005). The scope of open source licensing. *Journal of Law, Economics, and Organization, 21*, 20–56.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*, 309–346.

Peppard, J. (2003). Managing IT as a Portfolio of Services. *European Management Journal, 21*, 467–483.

Pinto, J. K., & Prescott, J. E. (1988). Variations in critical success factors over the stages in the project life cycle. *Journal of Management, 14*, 5–18.

Rajlich, T. V., & Bennett, H. K. (2000). A staged model for the software life cycle. *Computer, 33*, 66–71.

Raymond, E. S. (2001). *Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastapol, CA.: O'Reilly and Associates.

Sasi, V., & Arenius, P. (2008). International new ventures and social networks: Advantage or liability? *European Management Journal, 26*, 400–411.

Schweik, C. M. (2005). An institutional analysis approach to studying Libre software 'commons'. *European Journal for the Informatics Professional, IV*, 17–27.

Schweik, C. M., & English, R. (2007). *Tragedy of the FOSS commons? Investigating the institutional designs of free/libre and open source software projects*. National Center for Digital Government Working Paper Series 7.

Schweik, C. M., & Semenov, A. (2003). *The institutional design of open source programming: Implications for addressing complex public policy and management problems*. First Monday 8.

Sen, R., Subramaniam, C., & Nelson, M. L. (2009). Determinants of the choice of open source software license. *Journal of Management Information Systems, 25*, 207–239.

Singh, P.V., Tan, Y., & Mookerjee, V.S. (2007). Social capital, structural holes, and team composition: Collaborative networks of the open source software community. In *twenty eighth international conference on information systems*.

Smith, K. G., Smith, K. A., Sims, H. P., Jr., O'Bannon, D. P., Scully, J. A., & Olian, J. D. (1994). Top management team demography and process: the role of social integration and communication. *Administrative Science Quarterly, 39*, 412—438.

Sohn, S. Y., & Mok, M. S. (2007). A strategic analysis for successful open source software utilization based on a structural equation model. *Journal of Systems and Software, 81*, 1014—1024.

Stewart, K. J., Ammeter, A. P., & 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.

Stewart, K. J., & Gosain, S. (2006). The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly, 30*, 291—314.

Stuart, T. E., Hoang, H., & Hybels, R. C. (1999). Interorganizational endorsements and the performance of entrepreneurial ventures. *Administrative Science Quarterly, 44*, 315—349.

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

von Hippel, E., & von Krogh, G. (2003). Open source software and the ''private-collective'' innovation model: Issues for organization science. *Organization Science, 14*, 209—223.

West, J. (2003). How open is open enough? Melding proprietary and open source platform strategies. *Research Policy, 32*, 1259—1285.

Wu, J., Goh, K.-Y., & Tang, Q. (2007). Investigating success of open source software projects: A social network perspective. *Twenty eighth international conference on information systems*.

Zaheer, A., & Bell, G. G. (2005). Benefiting from network position: Firm capabilities, structural holes, and performance. *Strategic Management Journal, 26*, 809—825.

Zimmerman, M. A., & Zeitz, G. I. (2002). Beyond survival: Achieving new venture growth by building legitimacy. *Academy of Management Review, 27*, 414—431.

JING WANG is an assistant professor of Decision Sciences at the Whittemore School of Business and Economics, University of New Hampshire. Her research focuses on the area of IT Outsourcing, Open Source Software, and Agent-based Decision Support Systems. Her work has been published in *Decision Support Systems, Journal of Business Research, Journal of Information Systems, Journal of Strategic Information Systems*, and *Journal of Systems and Software*.