Journal of the Association for Information Systems

JAIS

Special Issue

Sustainability of Free/Libre Open Source Projects: A Longitudinal Study

InduShobha Chengalur-Smith
University at Albany
shobha@albany.edu

Anna Sidorova
University of North Texas
Anna.Sidorova@unt.edul

Sherae Daniel
University of Pittsburgh
sldaniel@katz.pitt.edu

Abstract

This paper examines the factors that influence the long-term sustainability of FLOSS projects. A model of project sustainability based on organizational ecology is developed and tested empirically. Data about activity and contribution patterns over the course of five years for 2,772 projects registered with SourceForge is analyzed. Our results suggest that the size of the project's development base, project age and the size of niche occupied by the project are positively related to the project's ability to attract user and/or developer resources. The ability to attract resources is an indicator of the perceived project legitimacy, which in turn is a strong predictor of the project's future sustainability. Thus a project's ability to attract developer and user resources is shown to play a mediating role between the demographic (size and age) and ecological (niche) characteristics of the project and its future sustainability. Our results support the applicability of tenets of organizational ecology related to the liability of smallness, the liability of newness, and population characteristics (niche size) to the FLOSS development environment. The implications of the results for future research and practice are discussed.

Keywords: *FLOSS, sustainability, organizational ecology, legitimacy, developer activity*

* Michael Wade and Kevin Crowston were the accepting senior editors. This article was submitted on October 15, 2009 and went through two revisions.

1. Introduction

Over the past decade Free/Libre Open Source Software (FLOSS) has become a viable alternative to proprietary commercial computer programs. Fueled by the rise of Linux and open standards such as HTML and Java in the 1990s, the concept of Free/Libre Open Source Software development permeated the Information Technology world during the early and mid-2000s. While FLOSS development started as a movement by groups of "hackers," often with little or no monetary motivation, the movement has been embraced by many traditional software development companies, such as IBM, which are experimenting with FLOSS development styles. Even Microsoft Corporation has agreed to open up the source code of some of its products. Such breakthroughs have given further legitimacy to the FLOSS movement and currently many organizations rely, at least for some functionality, on FLOSS. Companies and government agencies with limited financial resources are turning to FLOSS products as they can provide cheaper solutions for organizational IT needs (Bulkeley, 2003). A Forrester Consulting Study concluded that over half the companies in Europe and North America are using FLOSS for mission critical applications, and over 80% are using FLOSS for application infrastructure (Gold, 2007). Additionally, in the US, over half of the government agencies currently have or are implementing FLOSS (Gross, 2007).

Interestingly, only a few FLOSS projects, such as Linux, Apache and PHP, are responsible for most of the FLOSS movement's success and organizational adoptions. Yet these constitute just a miniscule percentage of all FLOSS projects. For example, SourceForge, a popular FLOSS development platform, hosts over 168,000 projects, but most of them become inactive soon after registration on SourceForge (Madey and Christley, 2008). Specifically, while some projects manage to sustain their vitality over a long period of time, the large majority of them exhibit little or no activity, suggesting that the projects are abandoned by developers. In fact, Stewart et al. (2006b) found that most projects cease to have activity after the first year. High project abandonment rates make the FLOSS movement less attractive for developers and pose risks for adopters and users of FLOSS applications.

From the perspective of the FLOSS community, the high abandonment rate implies the inefficient allocation of effort on the part of FLOSS developers. From a project leader's perspective, there are the sunk costs of starting a project which is later abandoned. Therefore, FLOSS project leaders would benefit from a better understanding of the factors that lead to project sustainability. From the developers' perspective, joining an existing project is associated with considerable investment because understanding source code related to an application well enough to modify it requires significant effort (Fichman and Kemerer, 1997; von Krogh et al., 2003). Investing in a project that is abandoned shortly after can be disappointing. In the long run, repeatedly investing in projects that end up being abandoned could drive otherwise motivated developers away from the FLOSS movement. Therefore, the ability to predict FLOSS project sustainability is also useful for developers who are trying to choose projects in which to invest their time and effort. In the long run, better understanding of factors that drive project sustainability can inform the actions of FLOSS project leaders and potentially help reduce FLOSS project abandonment rates. Hence, the FLOSS community as a whole would benefit from higher levels of FLOSS project sustainability.

From the adopters' perspective, high FLOSS project abandonment rates make it difficult to accurately predict FLOSS application total cost of ownership and, therefore, affect software acquisition decisions. When making software sourcing decisions, organizations consider a variety of factors, most of which are relevant whether the application is developed using open source or traditional practices. These factors include the features and functionality of the software, its compatibility with existing hardware and systems (Sahay and Gupta, 2003), as well as the total cost of ownership. The last factor is usually categorized into acquisition and administrative costs (David et al., 2002). Although open source may have an advantage over commercial software in terms of acquisition costs (FLOSS is often free to use), the administrative costs may offset that advantage. Administrative costs include the costs associated with acquiring special hardware and software for standardizing, centralizing and auditing, as well as security and support services (David et al., 2002). Accurately assessing the

service and support cost component is critical, as maintenance is often the most expensive part of software use (Banker et al., 1998).

For FLOSS applications, when no intermediary distributor or integrator such as Red Hat Enterprise Linux is involved, organizations have to rely on ongoing support from the FLOSS community. If a FLOSS project is abandoned, users of corresponding applications face significant challenges in terms of the availability of support and service. Therefore, the ability to predict future sustainability (the likelihood that the project will not be abandoned), based on publicly available data, will help potential adopter organizations to more accurately forecast FLOSS application administrative costs of ownership and will facilitate better informed sourcing decisions.

Considering the benefits of being able to accurately predict future project sustainability based on publicly available data, there is surprisingly little research that addresses this issue. Whereas there is a wealth of research on FLOSS project success, it has the following limitations in trying to address the problem of sustainability prediction. First, the majority of extant research is cross-sectional in nature and seeks to identify the factors that are associated with project success in the present time (Grewal et al., 2006; Stewart and Gosain, 2006). A notable exception to this is the study by Subramaniam et al. (2009), which examines the dynamic nature of relationships among various demographic characteristics of a project, yet gives only limited consideration to the project's environment and lacks an integrated theoretical framework. Second, some of the existing studies examine the internal dynamics of open source projects as determinants of project success; yet information about such factors is not easily available to potential adopters and developers (Stewart and Gosain, 2006).

In this study we draw on organizational ecology, a theoretical framework specifically concerned with organizational survival and sustainability, in order to identify factors that can be used as predictors of future project sustainability. Specifically we focus on two research questions:

- What characteristics of the project influence its future sustainability?
- What characteristics of the project's ecological environment influence its future sustainability?

The paper is organized as follows. First, we discuss the nature of FLOSS projects and review extant research on FLOSS project success and sustainability. Next, we draw on the organizational ecology literature and develop a theoretical model of open-source project sustainability. Specifically, we focus on such factors as project development base size, project age and the population niche size in which the project operates as important predictors of project sustainability. We then outline our research method for testing the proposed model, which involved conducting a longitudinal study that tracked the activity of 2,772 FLOSS projects registered with SourceForge over five years (2004-2009). Finally, we discuss the results of our investigation and conclude with the limitations of this research and avenues for future research.

2. FLOSS projects: Background

FLOSS products are generally distinguished from commercial software by their licensing arrangements (Stewart et al., 2006a; Stewart and Gosain, 2006). FLOSS typically allows the downloading of readable source code along with the program and gives licenses to copy, distribute and/or modify the program as long as the enhancements are distributed under the same licensing terms as the original software. This concept existed long before Richard Stallman founded the Free Software Foundation in the 1980s. IBM used to provide software, along with its mainframe hardware, that customers could enhance, and the customers then contributed their products to IBM for maintenance and further development (O'Reilly, 1999).

FLOSS is often developed by independent, geographically distributed programmers, typically working on one or more projects on a volunteer basis. FLOSS development is frequently done by self-organized teams with members working in parallel, and usually involves computer-mediated virtual collaboration. FLOSS projects normally consist of a core development team as well as a large

number of contributors who test for bugs and add functionality (O'Reilly, 1999; Ye and Kishida, 2003). In addition, there is often a peripheral group of users who request additional features and also provide feedback to the developers by downloading and recommending the software (Lakhani and von Hippel, 2003). Thus, a FLOSS project involves a group of interacting and cooperating individuals who work towards a specific goal, namely development or enhancement of a software product. In particular, FLOSS projects share characteristics with virtual organizations, as their members are geographically distributed, but collaborate using e-mail, groupware and other information and communications technologies.

3. Existing research: FLOSS project success and sustainability

Because of the vast adoption of FLOSS and the challenges to its development, researchers have been intrigued by this phenomenon and this has spurred a large number of related publications (Grewal et al., 2006; Stewart et al., 2006a; Hahn et al., 2008; Feller et al., 2008; Subramaniam et al., 2009). While a complete review is beyond the scope of this paper, we briefly review the research that is most relevant to the study of FLOSS project sustainability.

Sustainability is generally defined as the ability of an organism or an ecosystem to maintain its activity and productivity over time. Definitions of organizational sustainability vary from narrowly organizational to those that include social, economic and natural ecosystems. For the purpose of this research we limit our examination to the narrow view of sustainability. Thus we adopt the definition of sustainability as the ability to "survive and profit over the long run in both economic and natural environments" (Jennings and Zandbeergen, 1995). Because the key outcome of FLOSS projects is software development and maintenance activity rather than profit, we define a sustainable project as one that exhibits software development and maintenance activity over the long run.

Because sustainability is related to project activity, it is conceptually similar to the concept of project success, which has been extensively examined in FLOSS research. The key distinction is that sustainability requires certain levels of activity to be maintained over a long period of time, whereas success can be measured at one particular point in time or over the entire life of the project. Many papers focus on developer activity as a way to assess the success of FLOSS projects because, to the extent that coding is voluntary, an active development community indicates success (Stewart and Gosain, 2006; Subramaniam et al., 2009). Indicators of the amount of activity or effort that is exerted by developers on a project's application development include the number of tasks completed (Herbsleb and Mockus, 2003; Stewart and Gosain, 2006), the number of releases (Subramaniam et al., 2009), the number of concurrent versions system (CVS) commits (Grewal et al., 2006) and the lines of code per contributor (Fershtman and Gandal, 2004). Although the duration of observation varies, in most studies these indicators are assessed cumulatively over the entire public life of the FLOSS project. While the total amount of effort attracted is an important outcome of success, many potential adopters are interested in a complementary question: What allows these projects to continually attract developer activity over time? Subramaniam et al. (2009) seek to answer this question and explore how the characteristics of projects, including license choice, programming language and operating system on which the application can be used, impact multiple future success measures. They find that user and developer activity success indicators are correlated and that a higher stage, using a Unix platform and a C programming language, leads to future success.

The emerging literature that examines the sustainability of virtual organizations may help us understand why these factors influence future FLOSS project success, as FLOSS projects share many common characteristics with other kinds of distributed and volunteer groups (Markus et al., 2000; Gallivan, 2001). The resource-based view is one dominant strategic view that has been adopted to understand the sustainability of virtual organizations (Butler, 2001). The resource-based view of the firm suggests that if an organization possesses valuable, rare, immobile, inimitable and non-substitutable resources, it will achieve a sustainable competitive advantage and will be able to earn supernormal profits (Wade and Hulland, 2004). Organizations that fail to acquire or create such resources lose market share and will eventually be driven out of business. Butler (2001) identifies factors such as the membership size and communication activity as key enablers of sustainability for

online social structures.

While the resource-based view has been used to explain virtual organization sustainability (Butler, 2001), there are some characteristics of FLOSS projects that make the resource-based view useful but not sufficient to fully explain their sustainability. The resource-based view argues that rare, immobile resources that are not easily imitated facilitate an organization's sustainability. As open source projects are usually volunteer based, their developer base is often considered to be the key resource. Developers are clearly a valuable resource, but they are mobile. There are over one million developers registered on SourceForge and they often join and contribute to globally distributed projects (Madey and Christley, 2008). Another key resource for FLOSS projects is the source code. The initial ideas that are implemented in the FLOSS applications are valuable. Yet because they are usually codified and made available to the wide community of users and developers, they are mobile and imitable. In fact, FLOSS developers work on projects in an environment where sharing code and ideas is required by the licenses. A final key resource is the coordination mechanisms and structures that are used to support FLOSS development. These resources are often provided by large community platforms, such as SourceForge or gforge. Although these platforms are valuable resources, they are available to all projects, so they are not rare. Nonetheless, of the thousands of projects registered every year, only a handful become sustainable.

We propose that in the case of FLOSS projects, ecological forces, in addition to competitive dynamics, can explain sustainability. An ecological perspective suggests that it is not just the characteristics of the project, like the ones studied by Subramaniam et al. (2009), but also the characteristics of the project's environment that impact its sustainability. By exploring the characteristics of the project's ecological environment, we seek to identify antecedents of opens source software (OSS) project sustainability that are distinct from and complement those studied by Subramaniam et al. (2009).

4. Ecological lens for studying sustainability

Rooted in natural science, the ecological view was originally adopted by Hannan and Freeman (1977) to study organizational dynamics and changes in organizational forms. Since then, the ecological view has been used to explain a variety of organizational and social phenomena. In political science, ecological frameworks have been used to study the survival of interest groups (Heider-Markel, 1997). In Information Systems research, they have been used to study the development of internet standards (Nickerson and zur Muehlen, 2006) and e-infrastructure (Hepso et al., 2009).

The distinctive characteristic of organizational ecology compared with the other perspectives that emphasize strategic choice is that it focuses on the natural selection process rather than on how an organization can adapt to its environment. According to the ecological view, organizations are limited in their ability to adapt to environmental changes due to inertial forces; therefore, changes in organizational forms are best explained by the natural selection process: certain types of organizations are more likely to survive than others (Hannan and Freeman, 1977).

Ecological theory focuses on three processes: demographic, ecological and environmental (Baum, 1996). The demographic process is related to the age and size of organizations, the ecological process deals with niche size (the number of organizations in a population), and the environmental process takes environmental uncertainty such as technology cycles and regulatory influences into account. All three processes are believed to influence the survival of organizations. The demographic process represents the interplay between the demographic characteristics of individual organizations and their chances of survival, and is therefore highly instrumental in helping predict future survival at the level of individual organizations. The ecological processes are concerned with the effect of population characteristics on the survival of individual members. Finally, environmental processes are likely to affect whole populations. Since our focus is on understanding a FLOSS project's sustainability and not a population's sustainability, environmental processes are outside of the scope of this study.

Organizational legitimacy plays an important role in both demographic and ecological processes

(Hannan and Freeman, 1988) and, therefore, deserves special attention. Organizational legitimacy is a complex construct which is defined differently by researchers belonging to different research paradigms (Suchman, 1995). Yet the common thread in various definitions of organizational legitimacy is that it denotes the extent to which various organizational audiences see valid reasons for the existence of an organization and are willing to lend their support to it (Suchman, 1995). Organizational ecology considers legitimacy a necessary condition for survival. Therefore, most demographic and ecological factors that are considered to impact organizational survival do so by contributing to, or detracting from, organizational legitimacy.

4.1. Liability of smallness

Organizational size is the first among the key demographic factors that have been shown to influence organizational legitimacy and, consequently, the ability to attract resources and support from others. Smaller organizations have several disadvantages compared with large organizations, primarily due to difficulties in raising capital and creating organizational routines when their resources are stretched thin (Baum, 1996). Second, in competing with large organizations for labor input, small organizations are at a major disadvantage, since they cannot offer the long-term stability and internal labor markets that large organizations are thought to have. Additionally, large organizations seem more legitimate because their size is taken as an indicator of prior success and future dependability (Baum, 1996). Together these disadvantages underlie a liability of smallness as described by Aldrich and Auster (1986).

4.2. Liability of newness

In the context of organizational ecology, the liability of newness proposition suggests that a young or new organization is likely to have lower legitimacy and a higher chance of failure. The explanation for this phenomenon is that young organizations expend significant amounts of time and effort on coordinating new roles for the individual actors and on their mutual socialization. Therefore, they may be less efficient than more established organizations with which they are forced to compete. Emerging organizations may face intense selection pressures as they could lack formal goals and clear boundaries (Amburgey and Rao, 1996). In addition, they are more vulnerable because they lack bases of influence and stable relationships with external constituencies, and hence lack legitimacy (Baum, 1996). Older organizations are likely to have established social networks, including supplier and client groups, which makes them more resilient to minor variations in the environment, and makes it easier for them to attract and retain resources.

4.3. Niche size

Organizational ecology further suggests that organizational legitimacy is influenced by the population size in the ecological niche in which an organization operates. As the population size increases, this provides legitimacy to member organizations (Hannan and Freeman, 1988; Baum, 1996). Hannan and Freeman (1988) explain the relationship between the niche population size and organizational ability to gather support as follows:

> *At low densities, the growth of a population of organizations is constrained by the novelty and rarity of the form (Stinchcombe, 1965; Hannan and Freeman, 1986). The fact that there are few organizations in the population presumably makes it difficult to convince key actors, such as banks and government agencies, to transfer material and symbolic resources to organizations in the population. (p. 28)*

Niche population size is expected to have a positive effect on organizational legitimacy and on an organization's ability to attract resources. In sum, according to the organizational ecology literature, demographic and ecological processes explain how demographic and ecological characteristics such as size, age and niche influence organizational survival through their impact on legitimacy. Thus an organization's ability to become institutionalized or legitimate is a function of its age and size, as well as the size of the niche in which it operates. Once an organization achieves legitimacy, it can exploit the available resources to continue to expand and flourish, thus making it more sustainable.

5. Ecological view of FLOSS sustainability

The organizational ecology perspective is particularly appropriate for addressing our research goal for the following reasons. First, the dependent variable of interest in this study is sustainability, i.e. the ability of a FLOSS project to maintain levels of activity and output over time. Sustainability, although more frequently used in virtual community research, is conceptually similar to survival, and project abandonment is conceptually similar to organizational death. However, because there are few formal mechanisms for determining the finality of FLOSS project abandonment and because in rare cases abandoned projects are revived after years of inactivity (Evangelopoulos et al., 2008), it is often difficult to draw a clear line between dead and surviving projects. Therefore we focus on sustainability, and the wealth of organizational ecology literature that examines factors contributing to organizational survival is useful for answering our research questions.

Second, as discussed below, because it assumes that organizational ability to adapt is limited, organizational ecology focuses on demographic and population-level determinants of organizational survival, with particular emphasis on the circumstances surrounding the birth and early stages of organizational development (Hannan and Freeman, 1988). Applied to FLOSS projects, this would imply that project activity and sustainability would be impacted by the demographic and ecological characteristics of the project early in its life. Consequently, one can make inferences about the future of a project by examining the information about its past. Thus the ecological perspective provides a useful foundation when trying to identify predictors of future sustainability.

Because legitimacy is not easily observable, organizational ability to attract resources can be viewed as a proxy for how legitimate it is in the eyes of those providing the resources. Therefore, for the purpose of this study we will consider a FLOSS project's ability to attract resources as a proxy for its legitimacy. Unlike traditional organizations, FLOSS projects are largely volunteer based; therefore, the critical resources for FLOSS projects include commitment on the part of developers and users, whereas financial capital plays a lesser role. Although the traditional distinction between users and developers is that developers create the application and users make use of the application, the roles are not always as distinct in FLOSS projects. While the importance of user involvement in traditional projects has long been recognized (Jiang et al., 2006), users play an even more critical role in FLOSS development. In FLOSS projects users provide feedback on bugs and desirability of features, and can even sometimes fix bugs (Ye and Kishida, 2003). Therefore, we consider participative users as a resource, potentially as important for project sustainability as developers.

In using the ability to attract resources as a proxy for legitimacy, we recognize that it is likely to be a lagging indicator of legitimacy. We reason that both developers and users need time to react to their perception of a FLOSS project as legitimate before rendering their support. Moreover, the perception of legitimacy is likely to persist for some time, and is likely to be reinforced by additional indications of legitimacy. Similarly, if a project starts being perceived as less legitimate, it will take some time for the developers and users to withdraw their support. Therefore, we believe if a project is perceived as legitimate in time period T, we will be able to observe the increase in resources in time period T+n. Next we will discuss individual demographic and ecological factors that are related to legitimacy and FLOSS projects' sustainability.

5.1. FLOSS projects and the liability of smallness

FLOSS projects are likely to be as susceptible to the liability of smallness as traditional organizations for the following reasons. Learning is important for FLOSS developers (von Hipple and von Krogh, 2003) and larger projects (those with more developers) may be able to provide greater learning opportunities for developers, as they learn from one another. Further, since FLOSS can be seen as a meritocracy, many developers may be seeking to gain a positive reputation based on their efforts (Raymond, 2001; Roberts et al., 2006), and it could be easier to build a reputation more quickly if a developer works on a project that has a larger number of developers. The size of the project may be perceived as an indicator of the importance and quality of the ideas underlying a FLOSS project. If a project is able to gather a larger development base, potential users are likely to view the underlying application as more valuable, and thus the project as more legitimate. Together, these factors

contribute to larger projects having higher pragmatic legitimacy, and to their ability to attract developer and user resources in future periods. Therefore we predict the following:

> ***H1a:*** *The size of the project's development base (at time T) will have a positive effect on the project's ability to attract and retain developer resources in future periods (at time T+n).*
> ***H1b:*** *The size of the project's development base (at time T) will have a positive effect on the project's ability to attract and retain active user resources in future periods (at time T+n).*

5.2. FLOSS projects and the liability of newness

Like other organizations, FLOSS projects rely on governance structures and coordination processes. A new project will be perceived as less legitimate because it will have had less time to establish clear governance procedures, such as explicit directions describing how a developer can join and contribute to it. In addition to recruitment strategies, a new project may not have had time to develop credibility (both for the developers and for the project) and appropriate governance structures, including operational rules for the peer review process as well as for conflict resolution (Schweik and Semenov, 2003).

Trust and control structures in FLOSS projects facilitate production (Gallivan, 2001), but a new project may have weaker trust and control structures because it has not had time to develop the appropriate mechanisms. Thus, a new FLOSS project faces the challenges of identifying market opportunities, developing individual competencies, and managing a cooperative effort simultaneously (Crowston and Scozzi, 2002). Additionally, since a new project's user community is relatively nascent, it may not have developed ideas for new features or identified and corrected bugs in the application. These problems may be compounded by the fact that when the project is new, users may suspect that the application is less reliable, and therefore less legitimate compared to other applications, and choose not to use it.

Like traditional organizations, FLOSS projects are embedded in broader social and professional networks and draw on such networks to establish their legitimacy (Grewal et al., 2006; Hahn et al., 2008). Relationships with other projects can enable a project to attract developer effort and user attention (Grewal et al., 2006; Hahn et al., 2008). But, it could take time to find other projects to work on or interact with in the OS community. A new project may not have enough relationships with other projects to draw on the broader FLOSS network. For these reasons, a new FLOSS project is likely to be susceptible to the liability of newness and will be perceived to be less legitimate. If a project is perceived as less legitimate, it will attract and retain fewer human resources. Therefore we propose:

> ***H2a:*** *Project age will have a positive effect on the project's ability to attract and retain developer resources.*
> ***H2b:*** *Project age will have a positive effect on the project's ability to attract and retain active user resources.*

5.3. FLOSS projects and niche size

As discussed earlier, organizational ecology suggests that an organization derives legitimacy from the legitimacy and size of the niche to which it belongs. An organization that belongs to a more populous niche is perceived as more legitimate, thus making it easier to attract resources. We contend that this is also true for a FLOSS project. A niche is a set of organizations that have something in common. For FLOSS projects a niche could be a set of projects that use the same programming language or operating system, or are aimed at the same audience. We discuss the implications of each of these niches below.

The bigger the size of the programming language and operating system niche in which a project operates, the more developers and users will be familiar with that programming language and operating system. A developer or user is likely to perceive a more familiar programming language and operating system as more legitimate than an unfamiliar one, and will perceive the project that uses it as more legitimate by association, and will therefore render support to the project. In addition, a

project operating in a larger programming language niche will have access to a larger pool of developers and users, which will make it easier to attract developer resources (Crowston and Scozzi, 2002 made a similar argument). In addition to human resource availability, a project that is in a popular programming language niche may have greater source code resources. Code reuse is extensive in FLOSS projects (Haefliger et al., 2008) and a project that uses the same programming language as many other FLOSS projects will be able to re-use code from those projects.

Conversely, the competition for developers operating in popular programming languages is going to be stronger, which makes it harder to attract and retain development resources. As these two conflicting influences interact, we expect that on balance the size of the programming language niche is going to have a positive effect on the project's ability to attract and retain development resources. The net benefit is positive because developers can work on multiple projects at once and over time they can return to a project after working on another one. Following similar logic, the size of the operating system niche is likely to have a positive influence on the project's ability to attract and retain development resources.

The size of the project's niche with regard to the audience for whom the application is intended indicates the strength of the FLOSS community's ability to deliver applications to this type of audience. This will give legitimacy to projects serving that audience. If there are very few FLOSS projects directed at satisfying the needs of a particular audience, users representing this type of audience are not likely to look to FLOSS for solutions. This is going to have a negative effect on developer motivation. Developers may not be motivated to create an application that few people are interested in using. Once a critical mass of projects targeted at a particular audience is in place, the projects are likely to attract more users, which will motivate developers to be more active in these projects. This corroborates the findings of earlier studies that projects developing applications targeted at developers or more technical users (the more established audiences for FLOSS projects) are more successful (Bezroukov, 1999). Therefore, the size of the audience niche is expected to have a positive influence on the project's legitimacy and its ability to attract and retain development resources.

Although the niche size is likely to change over time as new projects join and leave the niche, we believe that the effect of niche size on projects' ability to attract resources will be delayed because FLOSS users and developers need time to react to their changing perceptions of legitimacy and render or withdraw their support for the project. Thus we predict:

> ***H3a:*** *Project niche size (at time T) will have a positive effect on the project's ability to attract and retain developer resources in future periods (at time T+n).*
> ***H3b:*** *Project niche size (at time T) will have a positive effect on the project's ability to attract and retain active user resources in future periods (at time T+n).*

5.4. Project sustainability

Sustainability is defined in this paper as the ability of the project to exhibit software development and maintenance activity over the long term. Such maintenance and development activity can be produced either by active developers who are registered with the project or by active users. Therefore, in order to remain active, a project needs to retain its existing active developers and users and to be able to attract new ones. As developers and users are free to leave a project, move from one project to another or to simply cease their activity on a project, projects that are better at attracting and retaining active users and developers are more likely to remain active longer, i.e. are more sustainable.

Attracting new users and developers is particularly important. While the initial developers contribute initial ideas about the features and design of the application they were seeking to develop, innovation typically occurs when multiple perspectives are merged (Cohen and Levinthal, 1990). When a new user or developer is attracted to the project he or she can offer suggestions for features to add to the application, which may not have occurred to the original developers. These suggestions may be

drawn from different perspectives compared to those already in the FLOSS project because a new member may interpret, appropriate, and manipulate the software to fit his or her geographical, technological or cultural setting. In so doing, the new participant may broaden the purpose and function of the application, thus invigorating future development efforts. So the project is likely to garner activity for a longer period of time if it gets new ideas from new users or developers after the initial public development phase. Retaining original developers is also important as long as they remain active and continue contributing new ideas. So if developer and user efforts are attracted and retained, this should lead to more activity over the long run. For these reasons we predict:

> ***H4a:*** *A project's ability to attract and retain developer resources (at time T+n) will have a positive effect on its future sustainability (at time T+n+k).*
> ***H4b:*** *A project's ability to attract and retain active user resources (at time T+n) will have a positive effect on its future sustainability (at time T+n+k).*

5.5. Summary of the research model

Summarizing the aforementioned research hypotheses, we propose that the developer base and niche size, as well as the project's age influence sustainability, i.e. the project's ability to remain active in future periods (at time T+n+k). Consistent with the logic of organizational ecology, these relationships are mediated by the project's ability to attract developer and user resources (a proxy for the project's legitimacy). Because we believe that the ability to attract user and developer resources is a lagging indicator of project legitimacy, independent variables and mediator variables need to be observed at different time periods (T and T+n respectively). Sustainability is conceptualized as the ability to remain active for a longer period of time, and so it is observed at an even later time T+n+k (see Figure 1 for the summary of the research model).

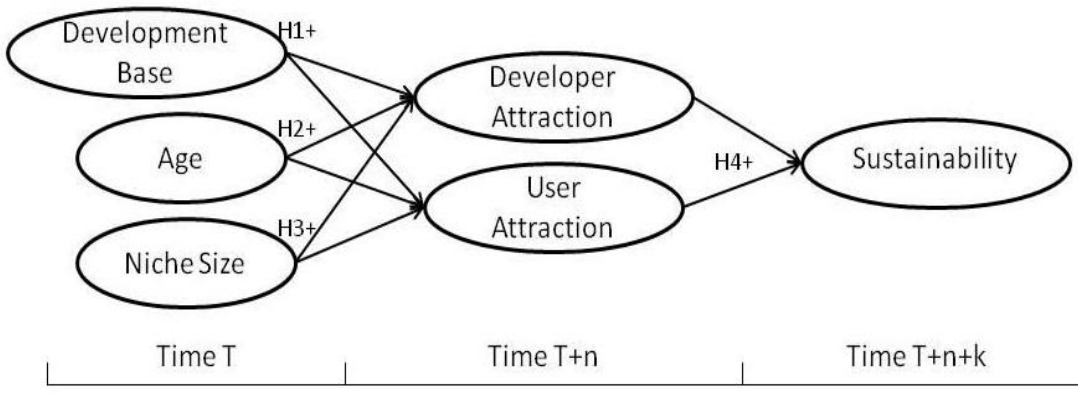

Figure 1: Research Model

In the next section we discuss the methodology we used to test the proposed model, including data collection and variable operationalizations.

6. Research method

In order to test the proposed model, we needed a large repository of data on open source projects so we turned to SourceForge. Given the temporal nature of our hypotheses, we collected annual data on both project and environmental characteristics starting with February 2004 to January 2005 and ending with the final observation period, February 2008 to January 2009. In order to align the causal reasoning in our hypotheses with the data available from SourceForge, we captured information on the antecedent variables (Age, Niche Size and Development Base) in 2004-2005 (corresponding to time T), recorded the legitimacy proxies (Developer and User attraction) in 2006-2007 (corresponding to time T+n) and observed the dependent variable (Sustainability) in 2008-2009 (corresponding to time T+n+k). Since we defined sustainability as the ability to remain active for a long period of time,

our intention was to maximize the time between observing the antecedent variables and measuring the project's future sustainability.

We used structural equation modeling (SEM) to explain the inter-temporal phenomena by capturing endogenous, mediator and exogenous variables at different points in time. Note that we did not measure the same indicators at different points in time, instead we used distinct indicators that uniquely captured our constructs. We had multiple indicators for each construct and we tested the model and the accompanying hypotheses using PLS. PLS is a regression-based SEM technique that is well suited to models containing multiple-item constructs and direct as well as indirect paths (Chin and Newsted, 1999). In addition, if the measures are not well established and the goal is to explain variance in the outcome variable, PLS is appropriate (Gefen et al., 2000). We began by testing a direct effects model and then introduced the mediator constructs to examine our hypotheses of complete mediation. We also performed a post hoc analysis using interaction effects to explore potential synergies among our antecedent variables. In the rest of this section we discuss our sample, indicators that we used for each of the constructs of interest, as well as the data collection and analysis procedures.

6.1. The sample

We used data available from the SourceForge Research Data Archive (SRDA) (Madey and Christley, 2008) and considered only those projects that showed signs of activity, i.e. had at least one closed artifact between Feb. 1, 2004 and Jan. 31, 2005 (time T). Of the approximately 5,000 projects that resulted, a critical threshold analysis showed that projects that had closed fewer than 5 artifacts (bugs, feature requests, patches, etc.) in 2004-2005 were not likely to remain active in 2008-2009 (time T+n+k). Specifically, projects with more than 5 closed artifacts had an overwhelmingly higher chance of showing some activity after 2 years, as compared to projects with fewer than 5 artifacts, suggesting that the latter set of projects might be distinct from projects that had more than 5 artifacts. FLOSS projects with fewer than 5 artifacts may have died before our observation period and were not the focus of our study. Alternatively, projects with fewer than 5 artifacts may have had little intention of using SourceForge for collaboration and may simply have wanted to make their source code public. These projects were not appropriate for our study because we wanted to study public collaboration. For these reasons we chose to focus on projects with more than 5 artifacts.

We defined a project as active if it had 5 or more artifacts (bug reports, feature requests, etc.) closed during the period Feb. 1, 2004 to Jan. 31, 2005. A total of 2,959 projects satisfied this "activity" criterion. We further queried the SourceForge archive to obtain the relevant data on these selected projects, using additional aggregation and grouping through MS Access, where necessary. Due to missing data, our final sample consisted of 2,772 projects.

6.2. The measurement model

The development base size was measured during time T using three indicators: (1) the number of developers registered with the project at the end of our first observation period, i.e. Feb. 2005, (2) the number of developers that contributed artifacts during the year of our first observation period (between Feb. 2004 and Jan. 2005), and (3) the number of developers that were assigned to address artifacts (i.e. bug reports or feature requests) between Feb. 2004 and Jan. 2005. Thus the development base size was more than just the total number of registered developers on the project team, it also captured information about the size of the active development team by considering the number of developers that opened and closed artifacts.

Project age was measured by two indicators: (1) the time elapsed between the date of project registration on SourceForge and March 1, 2009, and (2) the time elapsed between the date of the first release by the project and March 1, 2009. The niche size was measured on three dimensions, audiences, programming languages and operating systems. The indicators were: (1) the number of projects that served the same audiences as the focal project at the end of our first observation period (Feb. 2005), (2) the number of projects that used the same programming languages as the focal project as of Feb. 2005 and (3) the number of projects that used the same operating systems as the

focal project as of Feb. 2005.

The ability of the project to attract and retain developer resources was measured during time T+n by the following indicators: (1) the number of new registered developers between the ends of our first and second observation periods (Feb. 2005 and Feb. 2007), and (2) the number of artifacts contributed by developers during the year of our second observation period Feb. 2006-Jan. 2007. The ability of the project to attract and retain user resources was measured by the following indicators: (1) the number of new users who became active with the project between the ends of our first and second observations periods, i.e. Feb. 2005 and Feb. 2007, and (2) the number of artifacts contributed by users during the year of our second observation period (Feb. 2006-Jan. 2007).

Table 1: Variable Measurement

Construct	Indicator	Description	Dates
Development Base (Time T)	Number of Registered Developers 2005	Number of developers registered with the project as of Feb. 2005.	Feb. 2005
	Number of Active Developers 2005	Number of developers that contributed artifacts in the period Feb. 2004-Jan. 2005.	Period Feb. 2004-Jan. 2005
	Number of Assigned Developers 2005	Number of developers to whom artifacts were assigned in the period Feb. 2004-Jan. 2005.	Period Feb. 2004-Jan. 2005
Age (Time T)	Days Since Registration	The number of days between the registration date of the project and Mar. 1, 2009.	N/A
	Days Since First Release	The number of days between the first release date of the project and Mar. 1, 2009.	N/A
Niche Size (Time T)	Audience Niche	The number of projects that serve the same audiences as the focal project	Feb. 2005
	Programming Language Niche	The number of projects that use the same programming language as the focal project	Feb. 2005
	Operating System Niche	The number of projects that use the same operating system as the focal project	Feb. 2005
Developer Attraction / Retention (Time T+n)	New Active Developers 2007	The number of new developers registered with the project as of Feb. 2007 compared to Feb. 2005.	Feb. 2007
	Number of Developer Contributions 2007	The number of artifacts contributed by developers in the period Feb. 2006-Jan. 2007	Period Feb. 2006-Jan. 2007
User Attraction/ Retention (Time T+n)	New Active Users 2007	The number of new users who contributed artifacts as of Feb. 2007 compared to Feb. 2005.	Feb. 2007
	Number of User Contributions 2007	The number of artifacts contributed by users in the period Feb. 2006-Jan. 2007	Period Feb. 2006-Jan. 2007
Sustainability (Time T+n+k)	Number of Artifacts Closed 2009	Number of artifacts with the close date between Feb. 2008 and Jan. 2009.	Period Feb. 2008-Jan. 2009
	Number of Artifacts opened 2009	Number of artifacts with the open date between Feb. 2008 and Jan. 2009	Period Feb. 2008-Jan. 2009

Table 1 provides a summary of our constructs and corresponding indicators. (Note that the last column indicates which variables were measured over a period of a year as opposed to a point in time.)

Our sustainability measures captured during time T+n+k are similar to other outcome measures based on activity used in prior studies. Task completion has been used in the past to measure effectiveness of software development teams (Herbsleb and Mockus, 2003; Stewart and Gosain, 2006). It measures how responsive developers are to requests for new features and bug fixes. We measured project sustainability using two indicators: (1) the number of artifacts that were closed during the year of our third observation period Feb. 2008-Jan. 2009, and (2) the number of artifacts that were opened during the same time period. It is difficult to observe sustainability directly, we can only observe sustained activity at different periods in time; we therefore used sustained activity as an indicator of sustainability.

While one could argue that successful software projects may not need to add features, fix bugs or perform other activities, we believe that the number of open and closed artifacts is an appropriate measure of the sustainability of a FLOSS project. There are two main reasons for this. First, even successful software applications require continual modification because of the pace at which technology (i.e. computers, operating systems) change. Even if the functionality of software is deemed sufficient, continuous changes to the software are necessary to ensure its interoperability with the constantly changing technological environment. Without such upgrades, even the most functionally complete application will become obsolete in a couple of years. Another, perhaps even more important, reason to focus on the level of activity is our focus on FLOSS projects as information intensive virtual organizations, not on the specific software packages they produce. Without introducing continuous changes to their products, such organizations die. Not surprisingly, highly successful software development companies such as Microsoft and Apple continue to release new versions of their more “successful” applications.

6.3. Data analysis

The projects included used 46 different programming languages, supported 48 different operating systems, and addressed 18 different audiences. In addition, the projects had 40 different licenses and addressed 196 different topics. They also varied in age and size of the development base. (See Table 2 for descriptive characteristics.) There was substantial variation in the number of developers on a project, though the variation in the active (or assigned) developers was less than the variation in registered developers. Additionally, the number of registered developers was larger than the number of active or assigned developers, with each project averaging about 2 active developers. The projects in our sample were generally registered about 2,300 days before March 1, 2009 (i.e. the beginning of 2003), and also had their first releases at about the same time. The statistics show that projects shared audiences and operating systems more than they shared programming languages. This can be deduced by the fact that the average number of projects that share audiences and operating systems is about twice that of those that share the same programming language. The number of new developers and active users that joined the project during the observation mid-point period, 2006-2007, averaged one each, whereas the number of artifacts contributed by developers and users during this time period fluctuated wildly across the sample, averaging 8 and 25 respectively. Finally, the number of closed and opened artifacts during the last time period, 2008-2009, was high for some projects, but averaged 16 and 24, respectively, during the last year of observation.

We used PLS, specifically SmartPLS (Version 2.0 M3), to assess the validity of the model’s constructs and the relationships between those constructs. Although a normality check (Kolmogorov-Smirnov test) showed that none of our indicators was normally distributed, since PLS does not impose multivariate homogeneity and normality requirements, it was appropriate for our analysis. PLS performs a measurement (i.e., outer) model analysis to ascertain the overall psychometric properties of the indicators used to measure the model’s variables and a structural (i.e., inner) model analysis to ascertain the important relationships among the variables.

Table 2: Descriptive Statistics

Indicator	Mean	Std Dev	Min	Max
# of Registered Developers (2005)	6.8	11.8	1	281
# of Assigned Developers (04-05)	1.8	2.63	0	49
# of Active Developers (04-05)	1.7	3.17	0	58
Days Since Registration (2005)	2335.1	513.2	1514.9	3405.2
Days Since First Release (2005)	2307.4	602.5	1483.43	14304.3
Audience Niche size (2005)	45641.4	20776.9	39	101549
Op System Niche size (2005)	45646.4	26891.4	12	129535
Prog Language Niche size (2005)	15546.2	8684.0	4	61846
New Active Developers (2007)	0.3	1.03	0	20
# of Dev Contributions (06-07)	7.6	39.6	0	864
New Active Users (2007)	0.4	0.5	0	5
# of User Contributions (06-07)	24.8	110.5	0	3870
# of Artifacts Closed (08-09)	15.7	67.3	0	1506
# of Artifacts Opened (08-09)	23.4	185.3	0	8677

We began by evaluating the validity and reliability of the constructs in our model. Construct validity can be determined by assessing convergent and discriminant validities and they both measure how well the indicators relate to the constructs (Gefen and Straub, 2005). To demonstrate convergent validity, we used three tests: item reliability, composite reliability and the square root of the average variance extracted (Fornell, 1982). We determined item reliability by examining construct item loadings. In general, loadings at or above 0.500 demonstrate adequate item reliability (Chin, 1998). All our items had loadings above 0.600 (see bolded items in Table 3).

Table 3: Loadings and Cross-Loadings

Indicators	Dev. Base	Age	Niche Size	Developer Attraction	User Attraction	Sustain
# of Registered Developers (2005)	**0.834**	0.219	0.136	0.341	0.215	0.205
# of Assigned Developers (04-05)	**0.923**	0.129	0.123	0.480	0.348	0.319
# of Active Developers (04-05)	**0.945**	0.044	0.095	0.514	0.305	0.331
Days Since Registration (2005)	0.146	**0.927**	0.265	0.110	0.181	0.107
Days Since First Release (2005)	0.090	**0.899**	0.267	0.072	0.166	0.096
Audience Niche size (2005)	0.095	0.227	**0.664**	0.079	0.084	0.062
Op System Niche size (2005)	0.054	0.182	**0.604**	0.057	0.058	0.012
Prog Language Niche size (2005)	0.105	0.197	**0.786**	0.110	0.092	0.074
New Active Developers (2007)	0.509	0.118	0.112	**0.876**	0.472	0.408
# of Dev Contributions (06-07)	0.371	0.058	0.105	**0.873**	0.339	0.527
New Active Users (2007)	0.157	0.189	0.104	0.316	**0.686**	0.254
# of User Contributions (06-07)	0.332	0.127	0.084	0.408	**0.874**	0.367
# of Artifacts Closed (08-09)	0.341	0.120	0.081	0.562	0.414	**0.950**
# of Artifacts Opened (08-09)	0.139	0.041	0.038	0.227	0.195	**0.669**

Composite reliability is a measure of a construct's internal consistency, and all the composite reliabilities for our constructs were around 0.700 or higher (see Table 4). Finally, the square root of the average variance extracted (AVE) should be 0.70 or higher (Fornell and Larcker, 1981; Gefen and Straub, 2005), since it indicates that, on average, the construct accounts for at least 50% of the variance in its measures. These values are the diagonal elements of Table 4 (in bold font). Based on these three tests, we concluded that all our constructs have satisfactory convergent validity.

Discriminant validity can be measured by two tests. The first test compares the loadings of each item with its cross-loadings. Table 3 shows generally higher values for the loadings compared with the cross-loadings. The second test for discriminant validity requires that a construct's variance extracted, or shared variance between the construct and its items, should be greater than the shared variance between the construct and other constructs. This can be measured by comparing the square root of a construct's AVE to its correlations with other constructs (Fornell and Larcker, 1981). Our data exhibited discriminant validity on this measure as well (see Table 4). Based on these tests, we concluded that our items were valid and reliable and we proceeded to test the structural model.

Table 4: Construct Correlations and Composite Reliability

Construct	Dev Base	Age	Niche Size	Developer Attraction	User Attraction	Sustain	Composite Reliability
Dev Base	**0.902**						0.929
Age	0.132	**0.913**					0.909
Niche Size	0.128	0.291	**0.688**				0.728
Developer Attraction	0.503	0.101	0.124	**0.875**			0.867
User Attraction	0.328	0.191	0.116	0.465	**0.785**		0.761
Sustain	0.324	0.111	0.078	0.534	0.403	**0.822**	0.801

7. Results

A bootstrapping procedure (1,000 samples) in SmartPLS (v2.0 M3) was used to assess the significance of the hypothesized paths and the amount of variance in the dependent variables attributed to the explanatory variables (Chin, 1998). Figure 2 shows the results of our PLS model. Age, development base and niche size together explained about 13% of the growth in user attraction, and about 26% of the growth in developer attraction. This attraction among both users and developers explained about 32% of the variation in sustainability.

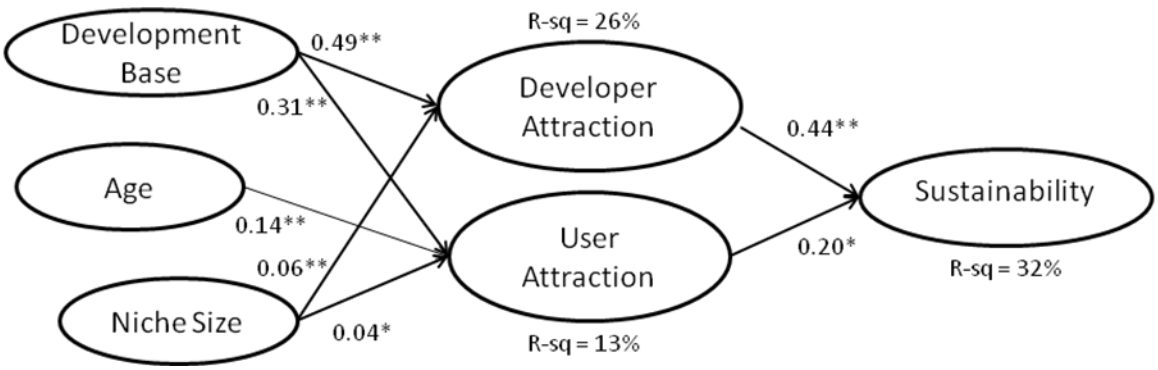

Figure 2: PLS Model Results

Table 5 summarizes our hypotheses and results. All our hypotheses, save one, were supported. The size of the development base in the early periods is a highly significant predictor of attracting future developers and users. The age of the project only seemed to be important to users, perhaps as an indicator of the maturity of the software. The size of the niche also played a small, but significant role, as the future ability of a project to attract developers and users, thus supporting hypothesis 3. Finally, the ability to attract users and developers had a significant impact on sustainability.

Table 5: Hypotheses				
Hypothesis	**From**	**To**	**Path**	**p-value**
H1a	Development Base	Developer Attraction	0.49	0.0000
H1b	Development Base	User Attraction	0.31	0.0000
H2a	Age	Developer Attraction	0.02	0.2852
H2b	Age	User Attraction	0.14	0.0000
H3a	Niche Size	Developer Attraction	0.06	0.0142
H3b	Niche Size	User Attraction	0.04	0.0481
H4a	Developer Attraction	Sustainability	0.44	0.0000
H4b	User Attraction	Sustainability	0.20	0.0449

We followed up this analysis with two post hoc analyses. First, in line with the organizational ecology theory, we tested the mediating effect of legitimacy in the model. Second, in order to further investigate the interplay among the constructs in our model, we tested for possible interaction effects among the significant explanatory variables.

In order to establish the mediation effect of legitimacy, as measured here by developer and user attraction and retention, we first tested a model with only direct paths from the antecedents to the dependent construct. The results of this model are shown in Figure 3.

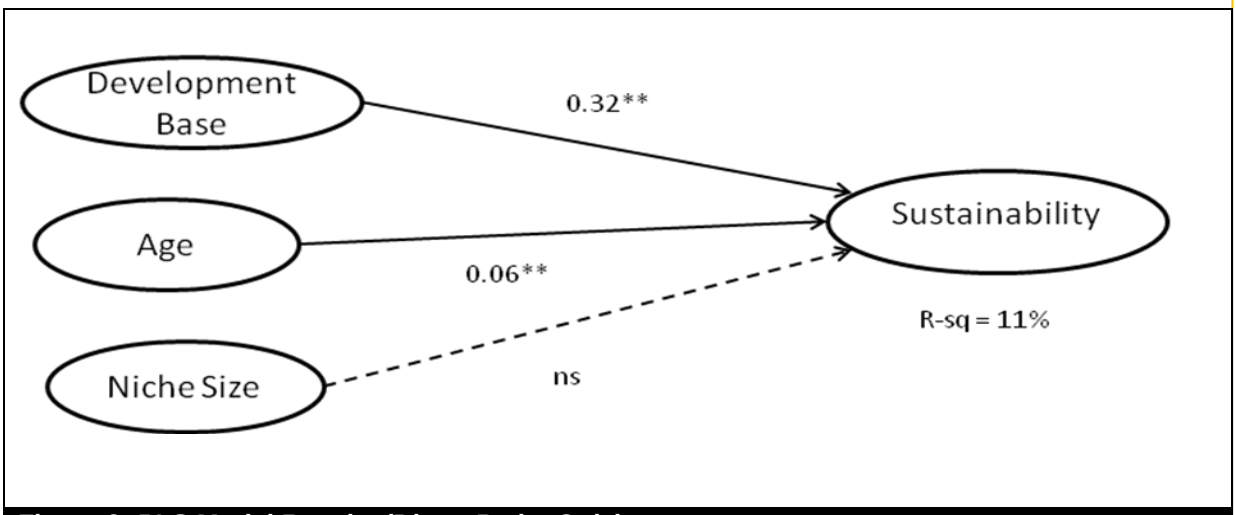

Figure 3: PLS Model Results (Direct Paths Only)

Table 6 summarizes our results for the model that includes only the direct paths from the antecedent constructs to the dependent construct. There is a significant direct effect of development base and age on sustainability, although the direct impact of niche size on sustainability is not significant.

Table 6: Path Coefficients and Significance			
From	**To**	**Path**	**p-value**
Development Base	Sustainability	0.32	0.0017
Age	Sustainability	0.06	0.0000
Niche Size	Sustainability	0.03	0.1242

After including the mediator variables in this model, we saw that in the case of both age and development base, their direct effects on sustainability vanished. The former acted on sustainability only through user attraction, whereas the latter acted through both developer and user attraction. (See Figure 4 and Table 7 for details.) Niche size appeared to have only a short-term effect, as it was

significantly related to both user and developer attraction, but the effect did not seem to persist to impact sustainability. This was manifested both in Figure 3, where we saw no significant impact of niche size on sustainability, and in Figure 4, where there was only an indirect effect of niche size on sustainability through both legitimacy mediators.

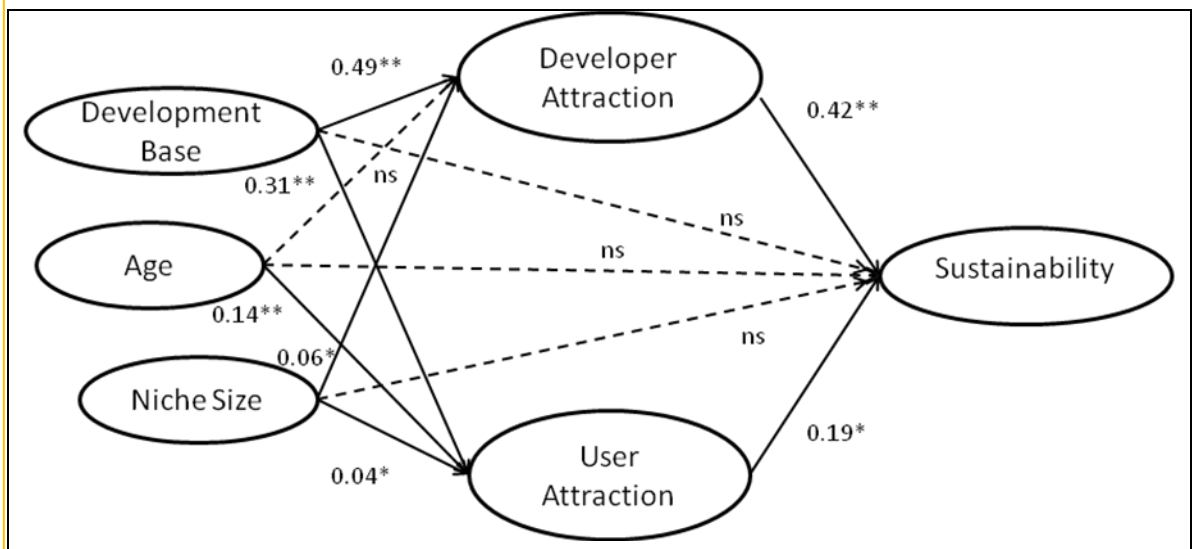

Figure 4: PLS Model Results (Direct and Indirect Effects)

Table 7 Direct and Indirect Paths

From	To	Path	p-value
Development Base	Developer Attraction	0.49	0.0000
Development Base	Sustainability	0.05	0.3129
Development Base	User Attraction	0.31	0.0000
Age	Developer Attraction	0.02	0.2590
Age	Sustainability	0.03	0.1477
Age	User Attraction	0.14	0.0000
Niche Size	Developer Attraction	0.06	0.0114
Niche Size	Sustainability	0.01	0.5554
Niche Size	User Attraction	0.04	0.0439
Developer Attraction	Sustainability	0.42	0.0000
User Attraction	Sustainability	0.19	0.0635

The second part of our post hoc analysis related to exploring potential synergies among the significant demographic and ecological characteristics of the projects. Our original model found that age was related to user attraction only, whereas development base and niche size related to both proxies of legitimacy, user and developer attraction (see Figure 2). Therefore we created an interaction term using development base and niche size and examined its relationship with legitimacy, i.e. developer and user attraction. Figure 5 shows the results of the model with this interaction term. We saw that there was a marginally significant, positive relationship between the interaction term and each of the legitimacy proxies, suggesting that these two project characteristics complement each other.

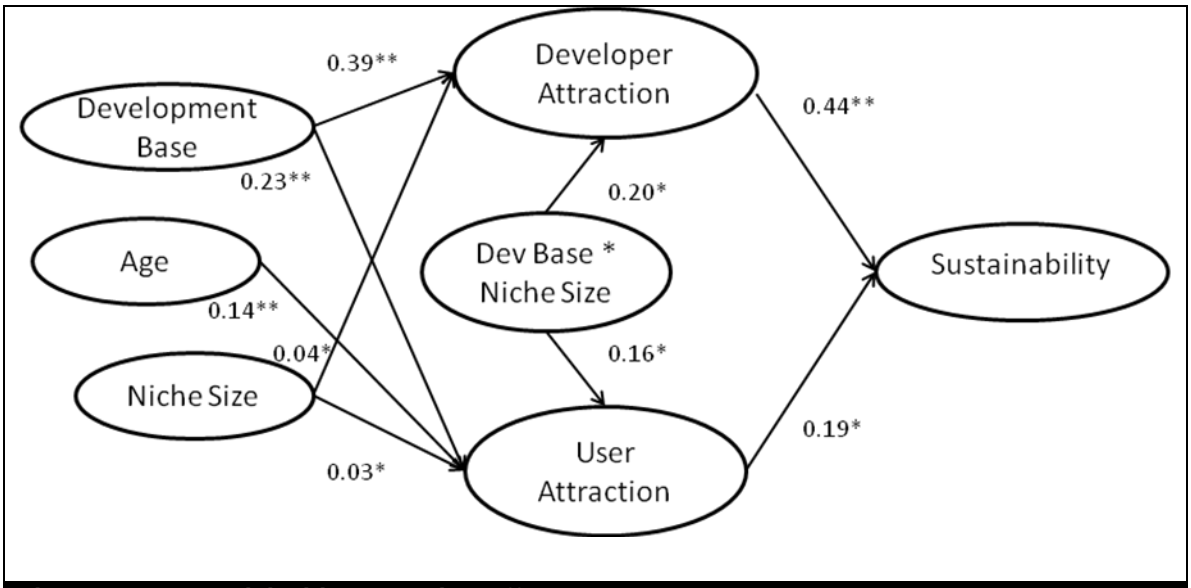

Figure 5: PLS Model with Interaction Effects

8. Discussion

The results of our study support the applicability of the ecological view to the study of FLOSS project sustainability. Specifically, our results suggest that consistent with the liability of smallness hypothesis, the size of a FLOSS project in terms of its development base predicts its legitimacy as indicated by the project's ability to attract resources in the future. Similarly, we found partial support for the liability of newness hypothesis. Our results showed that the age of the project played a marginally significant role in attracting active users, but not developers. We attribute this differential effect of age on users and developers to the fact that age may be seen as an indicator of application maturity by users, and hence taken as a positive signal, whereas it may convey more ambiguous signals to developers. For instance, a younger project may offer more interesting opportunities for development work as opposed to more maintenance-oriented work on relatively mature projects. The degree to which the work is interesting to developers is important because FLOSS developers are frequently volunteers. Additionally, an older project may have accumulated a more complex code base, making it challenging for a new developer to contribute to it (Ankolekar et al., 2003). While these arguments suggest why a FLOSS developer prefers a younger project, there are reasons that a FLOSS developer may select an older project. More established projects may be seen as better opportunities for developers to network and build reputations. Thus, the age of the project may send a conflicting message to a developer, resulting in a non-significant relationship between the age of a project and its ability to attract developer resources.

Further, our results are consistent with the ecological view which suggests that the size of the population in a niche positively contributes to the legitimacy of all organizations in that niche (Hannan and Freeman, 1988). Specifically, we found that the size of the niche to which a project belongs is positively associated with the project's ability to attract and retain developer and user resources. This finding is consistent with prior studies showing that the use of a particular programming language or focus on developing a certain type of application may impact the degree to which it is able to attract development activity (Fershtman and Gandal, 2004). Our study extends these findings by suggesting why some programming languages or operating systems are associated with more activity. This could be attributed to the synergies generated through code sharing and reuse among similar projects (Haefliger et al., 2008).

Additionally, our results show that there are some synergies between niche size and development base in terms of attracting developer and user resources. The positive relationship between the

development base size and the ability to attract human resources is stronger if the project belongs to a larger niche. This is consistent with the nature of FLOSS projects, whereby developers work in an environment in which sharing code and ideas is required by the licenses. However, the growth in certain areas could come at the expense of other programming language and operating systems niches, leading to dominance or emerging standards in these areas. Finally, consistent with the organizational ecology perspective, as well as the resource-based view, a project's ability to attract developer and active user resources was found to have a positive effect on project sustainability. The implications of these findings for research and practice are discussed in the following sections.

8.1. Implications for research

The results of this study have important implications for FLOSS research, as well as for research on virtual organizations and organizational research in general. We begin by examining the contributions of this work to FLOSS research and continue with broader implications.

Notably, our study was among the first efforts to use an ecological perspective in the study of FLOSS projects; it allowed us to illuminate certain interesting aspects of FLOSS development and to help explain the results of prior studies. We introduce the size of a project's niche as an indicator of its ecology, and our findings suggest that the project's niche size may be a significant indicator of its future sustainability. This finding contributes to a deeper interpretation of prior research. Researchers have observed differing levels of success for FLOSS projects that have various characteristics. For instance, extant research suggests that applications designed for Linux have less success than FLOSS projects that design applications for other operating systems (Fershtman and Gandal, 2004). Following the theoretical tenets presented here, the lack of success for a project developed for use on the Linux operating system could be explained by fewer projects working on applications developed for use on Linux (compared to Windows). Similarly Subramanium et al. (2009) find that the success of projects using more popular programming languages. such as C, is higher. Comino et al. (2005) find that 40% of projects on SourceForge are for use by developers, and other researchers suggest that FLOSS projects designed for the use of developers are the most successful (Fershtman and Gandal, 2004). Our results suggest that the fact that there are so many applications for developers is what allows these projects to be successful. In sum, this study shows that the organizational ecology perspective which emphasizes legitimacy and survival over economic profit and competitive advantage can offer unique insights into the dynamics of the FLOSS movement and can be used to inform future FLOSS research studies.

Several important contributions of our study are related to its longitudinal nature. While our results are consistent with previous research in that characteristics of the developer community are associated with the development group's level of activity (Stewart and Gosain, 2006; Grewal et al., 2006), our study is among the first to show the persistence of this relationship over multiple years. Subramaniam et al. (2009) explore the success of OSS projects using longitudinal data and our study complements theirs by taking a more theoretical approach and including ecological characteristics. By showing a temporally persistent relationship between demographic and niche characteristics of FLOSS projects and their future sustainability, our study helps bridge a gap in FLOSS research.

Similarly, the relationship between the characteristics of the developer base and project activity levels has been examined in extant research; however, such research has a few limitations that allow our study to complement it. For example, Stewart and Gosain (2006) and Grewal et al. (2006) measure their dependent and independent variables over the same period of time (over the entire life of the project). The use of longitudinal, rather than cross-sectional data, allows us to draw inferences about the causal relationships we hypothesize. Other studies that observe projects over time are limited to a relatively short time period. For example, Hahn et al. (2008) study projects for 2 months, Stewart et al. (2006a) observe projects for 8 months, and Fershtman and Gandal (2004) observe them for 18 months. While these studies shed light on the short-term dynamics of the project activity, they are less instrumental in understanding the project's long-term sustainability, which is of the utmost importance to FLOSS adopters. A few studies in which projects are observed for longer time periods include source code evolution studies, which observe how the source code changes over longer time

periods, but these mostly focus on the largest FLOSS projects such as Linux (Godfrey and Tu, 2002). Again, while providing valuable insights, such studies have limited implications for predicting the future sustainability of an average FLOSS project. By explicitly modeling the time lag between the demographic and ecological characteristics of a project and the project's legitimacy and future sustainability, this study opens avenues of research that examine projects over all stages in their life cycles.

Our study also highlights the importance of both developer and user resources for project sustainability, which has implications for future research on FLOSS development. With a few exceptions, FLOSS literature has focused primarily on the characteristics of the development group as an antecedent of development activity (Stewart and Gosain, 2006; Grewal et al., 2006). Because our results suggest that active users are as important a resource as the developers and that the ability of a project to attract active users and sustain their involvement is an important predictor of project sustainability, researchers should devote more attention to understanding the behaviors and motivations of active users.

From a methodological point of view, our study is distinct from prior FLOSS research. Rather than using the number of registered developers as a measure of the size of the development team, we offer a richer view of the project's development base by taking into account the number of active developers. Active developers are those who participated in the project by submitting bug reports or feature requests or those who contributed by fixing bugs or adding features during our observed time period. This more realistic portrayal of the development team provides clear evidence of the impact of the project's development base on its perceived legitimacy and hence the project's ability to attract developer and user resources in the future. Future researchers are thus encouraged to adopt richer operationalizations of the developer and user community, which more adequately reflect the involvement of developers and users with a FLOSS project.

Our study also has important implications for research on virtual organizations and organizational research in general. The use of organizational ecology and other perspectives that emphasize legitimacy and other institutional characteristics of organizations is consistent with the call by Orlikowski and Iacono (2001) to incorporate more institutional views of organizations in IS research. The applicability of the tenets of organizational ecology to the FLOSS movement suggests that organizational ecology may also be useful in explaining the dynamics of other virtual communities such as electronic networks of practice (Wasko et al., 2004) and social networks. Yet, one should exercise caution in extrapolating our results to other virtual communities and organizations. An ecological study of online communities (Wang et al., 2006) found that competition among communities in the same niche, e.g. newsgroups that shared content and membership, resulted in difficulty in retaining membership. This result is contrary to our findings and suggests that FLOSS communities differ from other online communities in important ways, and that they should be distinguished from other virtual organizations. It also indicates that the study of characteristics of virtual organizations that make them more susceptible to competition within the same niche is a fruitful direction for future research.

The results of the study are also important as they demonstrate support for the tenets of organizational ecology, which cannot easily be tested in traditional organizations. Testing is difficult due to the long time horizon necessary to observe organizational birth and decline. Further, there is a scarcity of publicly available data on sufficiently similar organizations to test the effect of demographic and ecological factors. There is a tension in the population ecology literature between the effects of legitimacy and competition, with increased legitimacy theorized to spawn a greater number of organizations initially, but eventually resulting in increased competition and mortality. Our findings of the positive effect of the niche size imply that legitimacy trumps competition during the five years of FLOSS projects we examined, but this may be an artifact of the particular time period. As FLOSS projects continue to flourish, if they are studied over a longer time period, the results may show the detrimental effects of competition. Additionally, our post hoc analysis of the interaction between demographic and ecological characteristics suggests that organizations are more susceptible to the liability of smallness in larger, more established niches. However, this finding needs further

investigation in other contexts before it can be generalized.

Our results also point to the existence of common ground between the organizational ecology and the resource-based views. The two perspectives may offer complementary insights into how the interplay of cooperation and competition in attracting resources impacts sustainability. Specifically, consistent with the resource-based view (Butler, 2001), we found that the ability to attract resources is key to project sustainability. We further found that both demographic and ecological characteristics contribute to the ability to attract resources, which suggests that competition for resources may exist among populations of organizations rather than among individual organizations. This implies that sharing resources within one population (one niche) may be beneficial for all organizations within that niche. Clearly our results only apply to FLOSS project populations and this finding should be corroborated by future organizational researchers and, if confirmed, can have implications on how resource sharing is viewed by organizations.

8.2. Implications for practice

This study has important implications for the FLOSS development community, as well as for the community of FLOSS users and potential adopters. Our results suggest the longitudinal persistence of the relationship between the ability to attract resources and sustainability, which makes it possible to use the ability to attract resources at a given time as a predictor of sustainability in the future. This is important for developers and adopters alike. Developers interested in joining a FLOSS project should not only focus on the current standing of the project, but should also consider its history (especially in terms of membership); such information is readily available at hosting sites. Information about other projects can also be culled from hosting sites and we submit that such information provides a valuable context for evaluating an individual project and its future sustainability. The wealth of public information about FLOSS projects provides the ability to track emerging trends or identify the growing dominance of certain kinds of projects, thus educating potential developers about the likely trajectory of the project of interest.

While developers could search for this information, hosting sites such as SourceForge may have opportunities to use the information about projects to facilitate their sustainability. A FLOSS project hosting website could recommend projects to developers based on the projects that they have worked on previously. Hosting sites might also consult with potential project initiators about the programming languages or operating systems that are associated with flourishing projects and encourage new projects to join those flourishing communities. In this way they could help to create more sustainable projects.

Our study has significant implications for FLOSS project leaders as well. While the results show the importance of both user and developer resources for project sustainability, they also suggest that the factors that influence the ability of a project to attract each of these resources are different. For example, in our study, age has a positive effect on attracting users, but not developers. This suggests that the legitimacy of the project is assessed differently by its users and by its developers, and project leaders may need to employ different methods to signal project legitimacy to each of these key stakeholder groups. Legitimacy can be communicated to developers by establishing the credentials of the team members or through media-based activities. Project leaders could explore other ways to increase the legitimacy of the project in the eyes of users, such as providing more documentation and expending more effort on improving the user interface.

While development base, age and niche were found to positively influence a project's ability to attract resources and the project's future sustainability, it is the positive effect of the niche size that is particularly noteworthy: it highlights that it is legitimacy and not competition that drives a project's sustainability. This suggests that FLOSS project leaders seeking to attract developers should make their projects more similar to others, as opposed to trying to differentiate on the basis of audience, operating system or programming language. They should also encourage more projects to join their niche as it is likely to increase the legitimacy of all projects occupying that niche. In more practical terms, the results imply the impact of code reuse among projects – as a greater wealth of code

becomes available to build on, the attractiveness of the niche increases, contributing to the legitimacy of projects in the niche.

This view also resonates with the emphasis of the FLOSS community on collaboration and knowledge sharing and is in line with the importance of FLOSS project networks shown in prior work (Hahn et al., 2008; Grewal et al., 2006). As projects share the same programming language or focus on the same operating system, there are likely to be developers that work simultaneously on many such projects. Thus a project occupying a larger niche is more likely to become related to another project through its developer network. This is apt to increase the legitimacy of the project and contribute to its sustainability. The relationship between network connectedness and niche is expected to be reciprocal. Projects that are connected by developers who work on multiple projects are likely to experience pressures to move towards the same niche in terms of programming language and operating system. Thus our results further support the importance of FLOSS development networks.

Our study also has important implications for the potential adopters of FLOSS applications. A number of practitioner texts such as (Golden, 2004) and (Guliani and Woods, 2005) provide advice on FLOSS selection, which includes paying attention to the maturity of the FLOSS and the support available. Based on our research, we can further inform potential adopters about the factors associated with sustainability in FLOSS projects. Organizations considering the use of FLOSS need to have some measure of confidence that the software project will not be abandoned in the future, as that would significantly impact the total cost of ownership of the software. Our results suggest that the current demographic and ecological characteristics associated with a project can help predict its future sustainability. In particular, our study indicates that a project that is similar to a large number of other FLOSS projects has advantages in terms of sustainability. This indicates that FLOSS as a method of development is more sustainable when developing applications for larger audiences using more popular operating systems and standard programming languages. Therefore, an organization considering adoption of a FLOSS application should consider how many FLOSS projects are developing similar applications using tools that are common among FLOSS projects. If a critical mass of similar FLOSS projects does not exist, specific projects may not be able to sustain themselves. In sum, the results of our study highlight important factors contributing to the sustainability of FLOSS projects and thus help inform the actions of FLOSS developers as well as FLOSS adopters.

9. Limitations and directions for future research

As this study sheds light on some important aspects of the dynamics of FLOSS projects, it is important to note its conceptual and methodological limitations. In terms of our methods, there are a couple of limitations. We used user IDs to indicate a unique developer or user, but it is possible that multiple individuals use the same identification. However, this technique is common in studies of online communities (e.g. Butler, 2001). We also focused on SourceForge projects, and since some of the largest projects, including Linux and Apache, do not use SourceForge, they are not part of our study. Because SourceForge represents a large portion of OSS projects, we do not consider this limitation to be critical. Finally, because of the nature of FLOSS projects, they can attract activity at any moment. So it is possible that a project that is deemed unsustainable could still have some activity in the future. However, this is unlikely.

The study also has some conceptual limitations. First, organizational ecology suggests that organizational accountability and reliability, along with demographic and niche factors, impact survival (Hannan and Freeman, 1977). While these may be relevant to FLOSS projects, it is difficult to assess a project's reliability and accountability based on public information; therefore, such predictors are less useful from the perspective of potential adopters. Yet, future researchers are encouraged to examine FLOSS project reliability and accountability characteristics and their relationship with project legitimacy and sustainability.

Second, in our study we considered the ecological environment to be static and, following the tenets of population ecology, assumed that organizations, and by extension FLOSS projects, do not change

in response to changes in the environment. Future research should more closely examine the applicability of such an assumption to FLOSS projects and explore how quickly a project can change if its environment requires it to change. A future study like this would build on this work as well as on the IT flexibility literature (Knoll and Jarvenpaa, 1994).

Further, although we conceptually relied on the legitimacy construct, we did not measure legitimacy directly, but only through the proxy of attracting resources. Direct assessment of legitimacy is not easy because the perceptions of various organizational stakeholders need to be captured. Therefore, a direct study of legitimacy may require alternate research methods, such as surveys or case studies. As legitimacy is central to the ecological view of sustainability and because our results lend indirect support to its importance, we encourage future researchers to more directly study FLOSS project legitimacy and its relationship with other project characteristics and outcomes.

In this study we focused on antecedents of legitimacy such as development base, age and niche size, the importance of which has been documented in organizational studies. However, it is possible to attain legitimacy in other ways, such as forming alliances with powerful organizations. For example, a FLOSS project could align itself with a large commercial software development firm or participate in FLOSS consortia. Such alliances have been documented and are considered by some to be the future of FLOSS development. Therefore, future studies should examine the effect of alliances on FLOSS project legitimacy and the consequences for project sustainability.

The results of our study also open a possibility for examining the dynamics of FLOSS projects through the lens of virtuous cycles. As a project with a larger development base attracts even more users and developers, it will be perceived as even more legitimate, leading larger projects to grow faster and smaller projects to decline at a higher rate. Similarly, as projects in a larger niche are perceived as more legitimate and remain active longer, the legitimacy associated with that niche is going to increase. Similarly, projects in a smaller niche may be perceived as less legitimate and decline, thus undermining the legitimacy of the niche itself. While the examination of the effect of feedback loops in the dynamics of FLOSS projects is beyond the scope of this paper, it may be an interesting direction for future research.

10. Conclusions

In this study we sought to identify project and ecological characteristics that can be used as predictors of future project sustainability. Guided by the organizational ecological view of FLOSS project sustainability, we found that project demographic characteristics such as age and size, as well as the niche size occupied by the project, influence the project's ability to attract and retain user and developer resources in the future. This, in turn, was found to be a good predictor of future project sustainability.

The key contributions of our study stem from its simultaneous attention to project-level and population-level factors and the longitudinal nature of the study. By showing how project demographic characteristics such as size, age and niche size influence a project's future ability to attract developer and user resources and, indirectly, project sustainability, we demonstrate the temporal persistence between the size of the development base, age and future activity. By highlighting the importance of niche size, our work strengthens the argument developed by studies that examine a project's network or relationship to other projects as important in understanding its success. A FLOSS project cannot be considered alone but should be considered as part of a larger ecology. Understanding how a project fits within this larger ecology will enable a stronger understanding of its sustainability. To best understand a project's future it is pertinent to understand both its internal characteristics and how it relates to the broader FLOSS community. Our findings have important theoretical and methodological implications for FLOSS research as well as for broader organizational studies, as they suggest future research directions. The results of our study also offer useful insights to FLOSS participants that can inform their decisions regarding starting a new FLOSS project, joining an existing project or adopting FLOSS applications. We hope that such insights can contribute to more efficient effort allocation and help improve the sustainability of future FLOSS projects.

References

Aldrich, H., and E. Auster. 1986. Even Dwarfs Started Small: Liabilities of Age and Size and Their Strategic Implications. In *Research in Organizational Behavior*, eds. L. L. Cummings and B. M. Staw. Greenwich, CT: JAI Press, pp. 165-198.

Amburgey, T. L., and H. Rao. 1996. Organizational Ecology: Past, Present and Future Directions. *Academy of Management Journal* 39(5):1265-1286.

Ankolekar, A., J. Herbsleb, and K. Sycara. 2003. Addressing Challenges to Open Source Collaboration with the Semantic Web. In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.) *Proceedings of Taking Stock of the Bazaar*: *3rd Workshop on Open Source Software Engineering*, held at the 25th International Conference on Software Engineering (ICSE), Washington, D.C. IEEE Computer Society.

Banker, R. D., G. B. Davis, and S. A. Slaughter. 1998. Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. *Management Science* 44(4):433-450.

Baum, J. A. C. 1996. Organizational Ecology. In *Handbook of Organizational Studies*, eds. S. R. Clegg and C. Hardy. Thousand Oaks, CA: Sage Publications.

Bezroukov, N. 1999. Open Source Software Development as a Special Type of Academic Research (A Critique of Vulgar Raymondism). *First Monday* 4(10).

Bulkeley, W. M. 2003. Free Software: Out of the Shadows. *The Wall Street Journal*, March 31, 2003, R6.

Butler, B. 2001. Membership Size, Communication Activity, and Sustainability: A Resource-Based Model of Online Social Structures. *Information Systems Research* 12(4):346-362.

Chin, W. 1998. The Partial Least Squares Approach for Structural Equation Modeling. In *Modern Methods for Business Research,* ed. G. Marcoulieds. Mahwah, NJ: Lawrence Erlbaum.

Chin, W. W., and P. R. Newsted. 1999. Structural Equation Modeling Analysis with Small Samples Using Partial Least Squares. In *Statistical Strategies for Small Sample Research*, ed. R. H. Hoyle. Thousand Oaks, CA: Sage.

Cohen, W. and Levinthal, D. 1990. Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly* (35):128-152.

Comino, S., F. M. Manenti, and M. L. Parisi. 2005. From Planning to Mature: On the Determinants of Open Source Take Off. Department of Economics working paper, no. 0517, University of Trento, Italy.

Crowston, K., and B. Scozzi. 2002. Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development. *IEE Proceedings Software* 149(1):3-17.

David, J. S., D. Schuff, and Louis, R. S. 2002. Managing Your Total IT Cost of Ownership. *Communications of the ACM*. 45(1):101–106.

Evangelopoulos, N., A. Sidorova, S. Fotopoulos, and S. Chengalur-Smith. 2008. Analysis of Survival of Open-Source Projects Based on Censored Activity Data. *Communications in Statistics: Simulation and Computation*. 37:1647-1662.

Feller, J., B. Fitzgerald, J. Hayes, and P. Finnegan. 2008. From Peer Production to Productization: A Study of Socially Enabled Business Exchanges in Open Source Service Networks. *Information Systems Research* 19(4):475-493.

Fershtman, C., and N. Gandal. 2004. The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination. *Centre for Economic Policy Research*.

Fichman, R. G. and C. F. Kemerer. 1997. Object Technology and Reuse: Lessons from Early Adopters. *IEEE Computer*, vol. 30, no. 10, pp. 47-59, Oct. 1997

Fornell, C. 1982. *A Second Generation of Multivariate Analysis: Methods*. New York, NY: Praeger.

Fornell, C., and Larcker, D. F. 1981. Evaluating Structural Equation Models with Unobservable Variables and Measurement Error. *Journal of Marketing Research* 18(1):39-50.

Gallivan, M. J. 2001. Striking a Balance between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies. *Information Systems Journal* 11(4):277-304.

Gefen, D., D. Straub, and M.-C. Boudreau. 2000. Structural Equation Modeling and Regression: Guidelines for Research Practice. *Communications of the Association for Information*

Systems 4:1-78.
Gefen, D., and D. W. Straub. 2005. A Practical Guide to Factorial Validity Using PLS-Graph: Tutorial and Annotated Example. *Communications of the Association for Information Systems* 16(5):91-109.
Godfrey, M. W., and Q. Tu. 2002. Growth, Evolution, and Structural Change in Open Source Software. Paper read at International Conference on Software Engineering, *Proceedings of the 4th International Workshop on Principles of Software Evolution*.
Gold, A. 2007. Open Source Solutions: Seek Value Beyond Cost. *Enterprise Open Source Magazine,* Sep 21, 2007, SYS-CON Media Inc.
Golden, B. 2004. *Succeeding with Open Source*. Washington, DC: Addison Wesley Professional.
Grewal, R., G. L. Lilien, and G. Mallapragada. 2006. Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. *Management Science* 52(7):1043.
Gross, G. 2007. Open Source Gaining Traction in US Government. *IDG News Service,* Nov, 2, 2007, Network World, Inc.
Guliani, G., and D. Woods. 2005. *Open Source for the Enterprise*. Sebastopol, CA: O'Reilly Media, Inc.
Haefliger, S., G. v. Krogh, and S. Spaeth. 2008. Code Reuse in Open Source Software. *Management Science* 54(1):180-193.
Hahn, J., J. Y. Moon, and C. Zhang. 2008. Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties. *Information Systems Research* 19(3):369-391.
Hannan, M., and J. Freeman. 1988. The Ecology of Organizational Mortality: American Labor Unions 1836-1985. *The American Journal of Sociology* 94:25-52.
Hannan, M., and J. Freeman. 1986. Where Do Organizational Forms Come From? *Sociological Forum* 1:50-72.
Hannan, M., and J. Freeman. 1977. The Population Ecology of Organizations. *American Journal of Sociology* 82(5):929-964.
Heider-Markel, D. P. 1997. Interest Group Survival: Shared Interests vs. Competition for Resources. *The Journal of Politics* 59(3):903-912.
Hepso, V., E. Monteiro, and K. H. Rolland K. H. 2009. Ecologies of e-Infrastructures. *Journal of the Association for Information Systems* 10(5):430-446.
Herbsleb, J. D., and A. Mockus. 2003. An Empirical Study of Speed and Communication in Globally-distributed Software Development. *IEEE Transactions on Software Engineering* 29(6):1-14.
Jennings, P. D., and P. Zandbeergen. 1995. Ecologically Sustainable Organizations: An Institutional Approach. *Academy of Management Review* 20:1015-1052
Jiang, J. J., G. Klein, and H.-G. Chen. 2006. The Effects of User Partnering and User Non-Support on Project Performance. *Journal of the Association for Information Systems* 7(2): 68-88.
Knoll, K. and Jarvenpaa, S. 1994. Information technology alignment or "fit" in highly turbulent environments: the concept of flexibility. In Ross, J. W., editor, *Computer Personnel Research Conference on Reinventing IS*, pages 1-14, Alexandria, Virginia. ACM Press.
Lakhani, K., and E. von Hippel. 2003. How Open Source Software Works: "Free" User-to-user Assistance. *Research Policy* 32:923-943.
Madey, G. and S. Christley. 2008. F/OSS Research Repositories & Research Infrastructures. *NSF Workshop on Free/Open Source Software Repositories and Research Infrastructures (FOSSRRI).* University of California, Irvine.
Markus, M. L., B. Manville, and C. E. Agres. 2000. What Makes a Virtual Organization Work? *Sloan Management Review* 41(1):13-26.
Nickerson, J. V., and M. zur Muehlen. 2006. The Ecology of Standards Processes: Insights from Internet Standard Making. *MIS Quarterly* 30 (Special issue on standards):467–488.
O'Reilly, T. 1999. Hardware, Software, and Infoware. In *Open Sources: Voices from the Open Source Revolution*, eds. C. DiBona, S. Ockman and M. Stone. Sebastpol, CA: O'Reilly & Associates, pp. 189-196.
Orlikowski, W., and S. C. Iacono. 2001. Research Commentary: Desperately Seeking the "IT" in IT research – A Call to Theorizing the IT Artifact. *Information Systems Research* 12(2):121-134.
Raymond, E. S. 2001. *The Cathedral and the Bazaar: Musing on Linux and Open Source by an*

Accidental Revolutionary. Sebastopol, CA: O'Reilly.
Roberts, J., I.-H. Hann, and S. Slaughter. 2006. Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science* 52(7):984-1000.
Sahay, B. S., and A. K. Gupta. 2003. Development of Software Selection Criteria for Supply Chain Solutions. *Industrial Management and Data Systems* 103(2):97-110.
Schweik, C., and A. Semenov. 2003. The Institutional Design of Open Source Programming: Implications for Addressing Complex Public Policy and Management Problems. *First Monday* 8(1).
Stewart, K. J., A. P. Ammeter, and L. M. Maruping. 2006a. Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. *Information Systems Research* 17(2):126-145.
Stewart, K. J., D. P. Darcy, and S. L. Daniel. 2006b. Opportunities and Challenges Applying Functional Data Analysis to the Study of Open Source Software Evolution. *Statistical Science* 21(2):167-178.
Stewart, K. J., and S. Gosain. 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Management Information Systems Quarterly* 30(2):291.
Stinchcombe, A. 1965. Social Structure and Organizations. In *Handbook of Organizations*, ed. J. March. Chicago, IL: Rand McNally, pp. 153-193.
Subramaniam, C., R. Sen, and M. Nelson, M. 2009. Determinants of Open Source Software Project Success: A Longitudinal Study. *Decision Support Systems* 46(2):576-585.
Suchman, M. 1995. Managing Legitimacy: Strategic and Institutional Approaches. *The Academy of Management Review* 20(3):571-610.
von Hipple, E., and G. von Krogh. 2003. Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science* 14(2):209-223.
von Krogh, G., S. Spaeth, and K. Lakhani. 2003. Community, Joining, and Specialization in Open Source Software Innovation: A Case Study., *Research Policy* 32(7):1217-1241.
Wade, M., and J. Hulland. 2004. The Resource-based View and Information Systems Research: Review, Extension, and Suggestions for Future Research. *Management Information Systems Quarterly* 28:107-142.
Wang, X., B. S. Butler, and L. Joyce, L. 2006. Competition: An Ecological Perspective of Online Communities. OCIS Division, *Academy of Management Conference Best Paper Proceedings*, AOM Annual Meeting, Atlanta, Georgia.
Wasko, M. M.; S. Faraj, and R. Teigland. 2004. Collective Action and Knowledge Contribution in Electronic Networks of Practice. *Journal of the Association for Information Systems* 5(11):493-513.
Ye, Y., and K. Kishida. 2003. Toward an Understanding of the Motivations of Open Source Software Developers. Paper read at *Twenty-Fifth International Conference on Software Engineering*, at Portland, Oregon.

About the Authors

InduShobha Chengalur-Smith is Associate Professor and Chair of the Information Technology Management department in the School of Business at the University at Albany – SUNY. She received her Ph.D. from Virginia Tech and her research interests are in the areas of Open Source Software, Technology Adoption and Implementation, Information Quality, and Security. She serves on the Editorial Boards of several journals including *Information & Management* and the *ACM Journal of Data and Information Quality*. Her research has been published in journals such as *Information Systems Research*, *Communications of the ACM*, and multiple *IEEE Transactions*.

Anna Sidorova is an Assistant Professor at the University of North Texas. She received her Ph.D. in MIS from Washington State University. Her research and professional interests include IT-enabled organizational transformation, business process management, and Open Source Software development. Her work appears in such journals as *MIS Quarterly*, *Journal of Management Information Systems* and *Communications in Statistics*.

Sherae Daniel is an Assistant Professor in the Katz Graduate School of Business at the University of Pittsburgh. She received her Ph.D. in MIS from the University of Maryland. Her research interests include Open Source Software development and other Internet mediated work processes. Her work appears in journals such as *Statistical Science*.