Dissertation Proposal

Early Dynamics of Open Source Software Projects: Exploratory Studies of Team Formation and User Adoption

Chen Zhang

Ph.D. Candidate
Krannert Graduate School of Management
Purdue University
West Lafayette, IN 47907
zhang153@mgmt.purdue.edu

October 29, 2006

CHAPTER 1. INTRODUCTION AND OVERVIEW

In this section we first explain the background of this dissertation research. Specifically we introduce the concept of open source software development (OSSD), and provide a brief survey of the current status of OSSD research. Next we describe the research problems and questions. We conclude the section by presenting the overall research model and an overview of the essays.

1. Background

1.1. Open Source Software Development

Software development has traditionally been regarded as an activity that can only be effectively conducted and coordinated within a firm setting. However, since the early 1990s an alternative model of software development, the open source software development model, has gained popularity as a promising approach to developing high-quality software (Raymond 2001). Open source software refers to software whose source code is available to users so that they can read, modify, and redistribute new versions of the software. A number of open source software (OSS) products, ranging from end-user applications (e.g., Emacs and OpenOffice), programming languages (e.g., Perl and PHP), web servers (e.g., Apache), to operating systems (e.g., Linux) have been widely adopted, demonstrating the attractiveness and viability of OSSD as an alternative to the conventional proprietary model of software production (O'Reilly 1999, Raymond 2001).

Raymond (2001) characterizes OSSD as a bazaar-style mode as apposed to the centralized cathedral-style mode adopted by traditional software development. In OSSD software developers, usually from different organizations and dispersed geographic locations, in Internet-based communities voluntarily collaborate to develop software (von Hippel and von Krogh 2003). From a broader perspective, OSSD is an example of the mergence of a new mode of production, "common-based peer-production" (Benkler 2002). Lerner and Tirole (2002) suggest that OSSD process is similar to "user-driven innovation" whereas von Hippel and von Krogh (2003) view OSSD as a "private-collective" model of innovation.

1.2. Overview of Research in Open Source Software Development

According to von Hippel and von Krogh (2006), researchers have been focusing on three main aspects of open source software development: developers' motivation to contribute, innovation

process, and competitive dynamics. Below we present a brief overview of the two areas that are most relevant to this dissertation: developers' motivations, and organization of the OSSD process.

OSS Developers' Motivations

From an economic perspective, developers' active involvement in the non-contractual software production activity without monetary remuneration seems like a behavioral anomaly that deserves a closer examination (Lerner and Tirole 2002). Therefore, the question related to developers' incentives and motivations has received a significant amount of attention from researchers. Individual motivations for participating in OSSD can be grouped into two categories, namely intrinsic motivation and extrinsic motivation (Hars and Qu 2001; Lakhani and Wolf 2005). Extrinsic motivation is associated with the immediate or delayed benefits that are typically awarded in the form of monetary compensation. External incentives identified in empirical studies include reputation effects, user software needs, learning benefits, and performance improvements (Hars and Qu 2001; Hertel et al. 2003; Lakhani and Wolf 2005). Intrinsic motivation relates to performing an action for the sake of the action itself rather than the consequences of the action. Intrinsic drivers of individual participation include enjoyment of programming, altruism, reciprocity, and community identification (Hars and Qu 2001; Hertel et al. 2003; Lakhani and Wolf 2005). Roberts et al. (2006) study the interactions among different motivations of OSSD contributors, how motivations influence developers' contribution, and how past project performance impacts individuals' future motivations. Shah (2006) identifies two types of participants, need-driven individuals and hobbyists, in the open source community.

Governance and Organization of OSSD Process

Researchers have investigated not only why individuals participate but also how they participate and coordinate in OSSD.

A number of researchers have examined how developers and users organize around an OSSD project. An OSS community can be characterized as a network of individuals performing various roles and making heterogeneous amount of contributions (Koch and Schnider 2002, Mockus et al. 2002). Nakakoji et al. (2002) classify the eight roles that a member of the OSSD community can take, namely passive user, reader, bug reporter, bug fixer, peripheral developer, active developer, core member, and project leader. Xu et al. (2005) categorize OSSD community members into four overlapping groups - project leaders, core developers, co-

developers, and active users. Other authors characterize the community as a series of concentric circles representing people playing a different role in the development process, which include core developers, maintainers, patchers, and users (Maas 2004; Moon and Sproull 2002). Most studies not only point out the existence of various roles participants can play within the OSSD community, but also stress the importance of keeping a balanced structure containing the different roles in the community. Recognizing that the evolution of OSSD communities is a rather dynamic process, von Krogh et al. (2003) develop an inductive theory of the strategies and processes by which newcomers join an existing OSSD community and how they start contributing. In addition, Robles et al. (2005) quantitatively analyze the evolution of the volunteer maintainer participation in the Debian project and investigate how maintainer involvement affects the software releases and the project community.

Besides division of labor in OSSD, scholars have also looked at other aspects of OSSD, such as collaboration between users and development teams (Zhao and Deek 2003), conflict management (van Wendel de Joode 2003), determinants of open source licenses (Lerner and Tirole 2005), coordination mechanisms (Crowston et al. 2005), structural and dynamics of OSSD social networks (Crowston and Howison 2005, Howison et al. 2006, Madey et al. 2002), and knowledge sharing (Kuk 2006).

A number of researchers have tried to identify the success factors of OSS projects (Colazo et al. 2005, Crowston and Scozzi 2002, Grewal et al. 2006, Stewart and Ammeter 2002, Stewart and Gosain 2006b). Stewart et al. (2005) examine the impact of licensing choice and organizational sponsorship on OSS success. Stewart and Gosain (2006a) propose a framework of the OSS community ideology and tests how adherence to the ideological beliefs impacts team effectiveness.

2. Research Questions

2.1. How do developers self-select into OSSD projects?

While traditional software development teams are formed by managers based on developer skills and experiences, in most cases the formation of OSSD teams is a dynamic, self-organizing process in which developers self-select into projects. Usually an OSSD project starts out with a developer or a small group of developers who want to create a software product that meets their personal needs. Other developers may join the project at a later time. Although the research on

OSS developer motivations has shed light on why individuals contribute time and effort to the development of open source software, it remains unclear how developers evaluate projects at the early stage and choose which project to become a member of.

Understanding the formation process of OSSD project teams is important for several reasons. Research on small groups has shown that group composition affects group performance through its impact on group cohesion and coordination (Beal et al. 2003, Gruenfeld et al. 1996, Levine and Moreland 1990), and administrative and expertise coordination effectiveness (Faraj and Sproull 2000b). In addition, team formation determines the programming skills and application domain experiences of team members, which also affect software development team performance (Boehm 1987, Curtis et al. 1988). Hence, in order to understand and solve the key problems related to staffing and project performance, it is important to understand the dynamics of OSS team formation.

In the first essay we attempt to explore the factors that impact individual choice of which newly-initiated open source software project to contribute to.

2.2. What factors impact users' adoption of open source software?

A number of researchers have identified OSSD as a fundamentally different way of organizing innovation production that depends on the collective involvement of community members (Benkler 2002, Franke and Shah 2003, von Hippel and von Krogh 2003). User adoption plays an essential role in the success of OSSD, an example of decentralized innovation process, for several reasons. First, user adoption can provide confirmation of developers' efforts and contribution, offering a source of motivation and helping retain developers in the project. Second, the individuals who actually try the software are more likely to contribute code and provide feedback to the project in the future. Thirdly, OSS success measures such as project popularity, project vitality, number of downloads, and size of the development community (Crowston et al. 2003, Grewal et al. 2006, Stewart et al. 2005), are all closely related to user adoption of OSS. Therefore, it is important to identify the factors that help an open source project attract users to adopt its software.

In the second essay we focus on the early user adoption behavior of OSS projects' first software releases and try to answer the following research questions: when the first version of open source software becomes available for downloads, what factors impact user adoption behavior and how this behavior affects the project's future performance.

3. Dissertation Overview

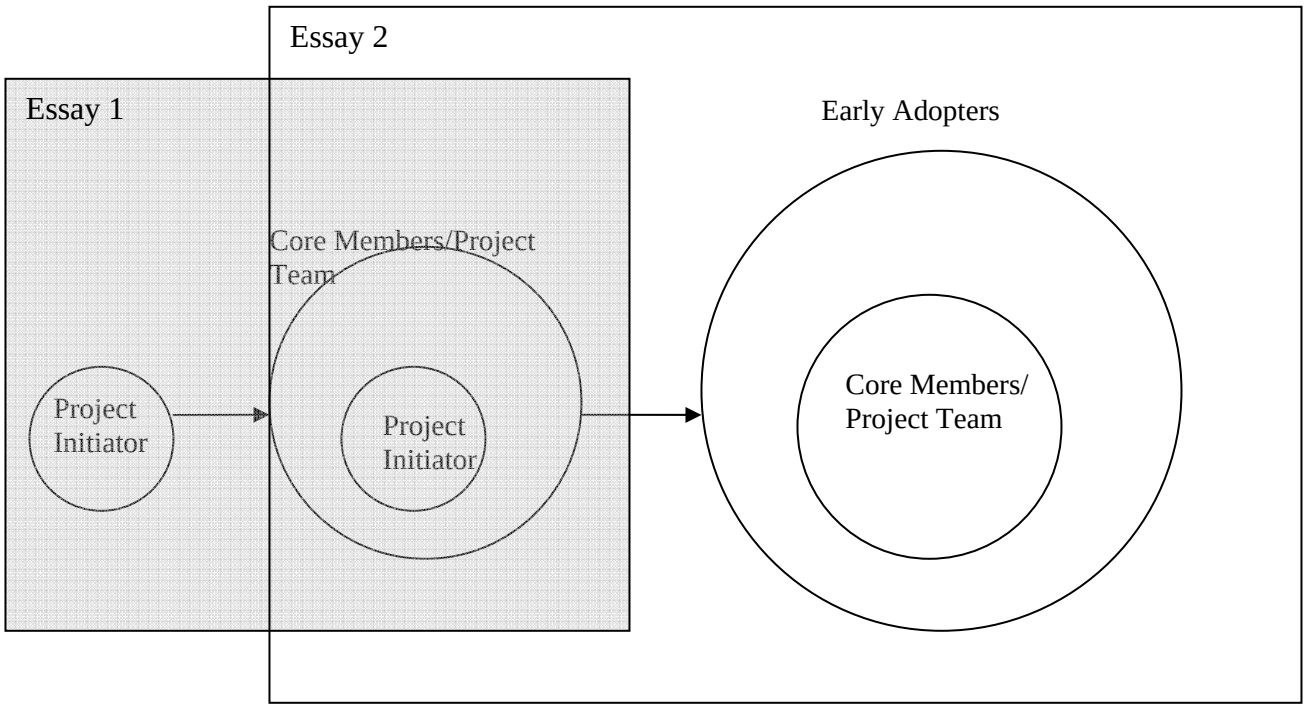
Figure 1 presents an overall research model that illustrates the scope of this dissertation. The first essay explores how a project team consisting of core developers emerges after the initiator registers the project. The second essay investigates how users become early adopters after the project team releases the first version of software. The dissertation focuses on two stages of OSSD that are partially overlapping in temporal sequence. Although initiation of a project precedes its code releases, formation of teams may overlap with code releases because developers can choose to join a team at any point in time as long as the project is still under development, which usually takes several years or even longer.

One of the main objectives of OSS research is to identify how to make open source projects successful, which has been the topic of many studies (Colazo et al. 2005, Crowston and Scozzi 2002, Grewal et al. 2006, Stewart and Ammeter 2002, Stewart and Gosain 2006a, Stewart and Gosain 2006b, Stewart et al. 2005). The success of OSS projects is path-dependent in that it not only depends on their current state but also depends on the steps they take to reach the current state. A project that is considered successful (i.e. it has many developers as well as a large number of downloads) based on its current status may not continue to be successful in the future and vice versa. Therefore, an ideal approach is to perform a longitudinal study of OSS projects to trace their major changes over their entire life span and identify the events that have a significant impact on the evolution of the software and the project (von Hippel and von Krogh 2003). However, the feasibility of this approach is constrained by the unavailability of OSS project data covering such a long period of time. In addition, the sample size is likely to decrease as time passes because some projects may become either abandoned, inactive for a long period of time, or forked, which makes it problematic to maintain a manageable initial sample size and have a large enough sample after several years of data collection.

Hence, instead of conducting a longitudinal study spanning over a number of years, we break it down into a series of cross sectional studies focusing on different events of the OSSD process. This dissertation consists of two cross sectional studies concentrating on the early stage of team formation and user adoption because projects often start to show differences in team size and user base size right from the beginning. A closer look at the early team emergence and

software adoption can yield a better explanation for the successes of some projects and the failures of other projects.

Figure 1. Overall Research Model



Essay 1 empirically examines the impacts of project goal definition, initiator experience, developer experience, initiator-developer prior collaborative ties, and task-skill fit on how OSSD project teams are formed. Using software project data from real world OSSD projects, we find that both the existence and the process-related strength of collaborative ties between the developer and the project originator positively impact the likelihood of the developer joining the project. Although initiators' experience in terms of duration and project is not a significant predictor, their network embeddedness does have a positive and important influence on the project's probability of attracting more developers. We also found that developers' experience has a negative impact on their joining behavior. Overall, the results suggest that the emergence of OSS project teams embedded in a collaborative network of developers is influenced by the relations formed by developers' past collaborations. We also explore the performance implications of early team formation behaviors.

Essay 2 focuses on early user adoption behavior of the first version of code released by OSSD projects. We propose that project visibility factors influence the number of potential

adopters seeking project-related information and that project and software properties help attract potential adopters to download and try open source software. We also plan to explore whether early user adoption leads to differences in OSS projects' subsequent performance.

CHAPTER 2. EMERGENCE OF OPEN SOURCE SOFTWARE DEVELOPMENT TEAMS

1. Introduction

The creation of industrial-strength software code (or software development) has traditionally been regarded as an activity that can only be effectively conducted and managed within a firm setting. An alternative model of software development, the open source software development (OSSD) model in which programmers in Internet-based communities collaborate to voluntarily contribute programming code has emerged as a promising approach to developing high-quality software (Raymond 2001). During the past few years, a number of open source software (OSS) products, ranging from end-user applications (e.g., Emacs and OpenOffice), programming languages (e.g., Perl and PHP) to applications supporting the Internet infrastructure (e.g., sendmail), have been widely adopted, demonstrating the attractiveness and viability of OSSD as an alternative to the conventional proprietary model of producing software (O'Reilly 1999, Raymond 2001).

Open source software project success depends on successfully attracting and sustaining volunteer developer interest and effectively coordinating their contributions. A growing body of literature has examined factors motivating individuals to participate in OSSD despite the lack of monetary compensation (e.g., Hars and Qu 2002, Lerner and Tirole 2002, Roberts et al. 2006). The OSS literature points to different types of benefits that motivate OSS developers. Hars and Qu (2002) identify both intrinsic motivations such as altruism and extrinsic motivations such as direct compensation. Another study surveys the motivations of the contributors to a large OSS project and finds that participation is mainly driven by developers' group identification, by the possibility of improving their own software, and by their tolerance of the required time investments for contributing to the project (Hertel et al. 2003). Lakhani and Wolf (2005)

identify enjoyment-based intrinsic motivation, user need, and learning as the most pervasive drivers of developer participation. Other possible explanations for developers' participation in OSS projects include motivations related to career concerns and ego gratification (Lerner and Tirole 2002). Overall, the prior literature suggests that developers participate in OSSD mainly because of intrinsic factors such as enjoyment and extrinsic factors such as career advancement.

The number of open source software projects launched by both commercial and non-commercial actors has increased rapidly during the past few years. However, despite the impressive success of some OSSD projects, many projects fail to take off and become abandoned. One of the main reasons cited for the failure of OSS projects is the lack of developers in the project teams, or the inability of the project to bring together a critical mass of developers (Lerner and Tirole 2001, O'Reilly 1999, von Krogh et al. 2003). Since OSSD projects typically do not provide monetary rewards for developers' contributions, many of them are under-staffed and consequently not well-equipped to deal with the complexity in software development (von Krogh et al. 2003). In most cases the formation of an OSSD project team is a dynamic, self-organizing process in which developers self-select into the project. The collective results from the research on OSS developer motivations suggest that developers will most likely join those projects that they perceive will provide the greatest opportunity for realizing private benefits such as learning, reputation or enjoyment. We try to identify the factors that developers consider when choosing which OSSD project to contribute to.

Understanding how OSS project teams are formed is important for several reasons. Such an understanding can enable us to examine the impact that the driving forces behind team formation process have on the possible composition of emergent teams. Research on small groups has shown that group composition is an important determinant of group performance through its impact on group cohesion and coordination among other factors (Beal et al. 2003, Gruenfeld et al. 1996, Levine and Moreland 1990). Member composition can also affect software development team performance through its impact on administrative and expertise coordination effectiveness (Faraj and Sproull 2000b). Software development team performance is also affected by the programming skills and application domain experiences of team members (Boehm 1987, Curtis et al. 1988). Hence, in order to understand and solve the key problems related to staffing and project performance, it is important to understand the dynamics of OSS

team formation – how OSS projects attract developers to join the development team and how developers choose software project teams to contribute to.

In this paper, we undertake an empirical examination of the formation process of OSSD project teams by taking into consideration characteristics related to project, project initiator, developer, developer-initiator past collaboration, and task-skill fit between project and developer. We attempt to identify the driving forces that bring developers to an emergent project team.

2. Theoretical Development and Hypotheses

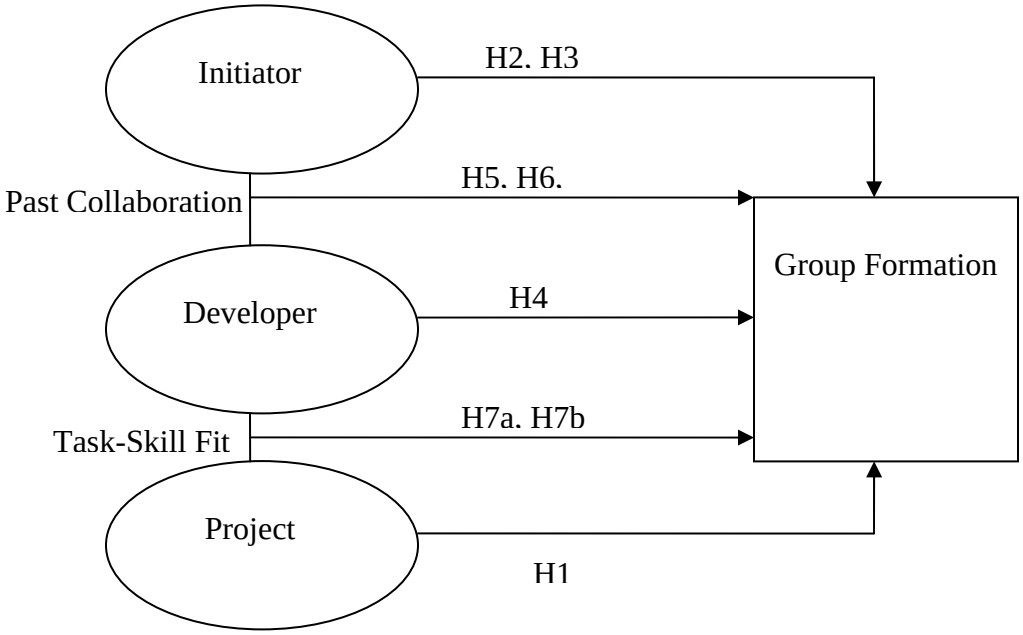
Some researchers in social psychology have conducted studies investigating the formation of natural or social groups (Levine and Moreland 1991, Moreland & Levine, 1992, 1996, Moreland, 1987). Their proposed group formation model consists of three stages – evaluation, commitment, and role transition. During the evaluation phase the individual looks for a group that can satisfy his or her personal needs whereas the group looks for an individual who can help achieve the group goal. Unlike the group examined by these work, an OSSD team is formed by developers self-selecting into projects[1] with few projects engaged in active recruiting (von Hippel and von Krogh 2003). In other words, during the evaluation stage a developer seeks a project that he or she wants to join. Furthermore, research on group formation indicates that group formation is a result of deliberate, strategic decisions of individuals to satisfy individual and group objectives (Owens et al. 1998). In the OSSD context, to strategically evaluate and decide which project to join developers rely on as much information as possible.

A newly-initiated open source software project consists of two key components – project and project initiator. Thus, there are two types of information, project-related and initiator-related, available to developers. Developers rely on such information not only to make inferences and form opinions about a project but also to make strategic decisions whether or not to join the project and collaborate with the initiator. We propose that there are five possible aspects influencing the self-selection process of OSS team formation, namely project characteristics, initiator characteristics, developer characteristics, developer-initiator past

[1] Although project administrators may decline a developer's request to join a project, they usually do not actively recruit developers. Thus, the selection is typically one-way from developers to projects.

collaboration, and task-skill fit between project and developer. Figure 2 presents our research framework and hypotheses.

Figure 2. Research Framework and Hypotheses



2.1. Project Characteristics

Traditional collective behavior theory maintains that the specification of project goals is important for successfully recruiting individuals to perform a collective task (McPhail and Miller 1973, Snow et al. 1980, Snow and Benford 1992, Benford 1993). Although most OSSD projects do not have a code base when they are first initiated, they do usually contain the information informing developers of the scope, intended functionalities, and technical properties of the software. From a developer's perspective, a project with a clear definition of its goals and relevant properties is likely to be better-planned and have a higher chance of succeeding than other vaguely-defined projects.

H1: (Project Level): *The number of developers who join a project is greater when the project has a clear definition of goals.*

2.2. Initiator Characteristics

During the early phase of a project the initiator usually plays the role of a project leader although other developers may take over the leadership role from his/her as the project

progresses. The experience level of the initiator may influence developers' perceived competence of the project leader, which impacts their perceived likelihood of project success. Therefore, we hypothesize

H2: (Developer Level: Developer-Project Dyad): *The probability that a developer joins a project is greater when the project initiator has more experience.*

Moreover, OSS developers may use indirect experiences of other developers in the collaboration network to judge the likelihood of project success. Projects initiated by people who are perceived to have higher status on average are more attractive to developers, possibly due to the belief that high status individuals are more competent and hence have a greater likelihood of initiating a successful project (Stewart 2005, Thye 2000). In addition, because of the reputation-conferring benefits of association with high-status participants in the network (Stewart 2005), OSS developers motivated by reputation will be more likely to join projects that are initiated by members with high perceived status in the community. The status of the initiator in the collaboration network can be inferred from the degree of his or her network embeddedness, or the relative perceived centrality, within the OSS developer network. Thus, we hypothesize

H3: (Developer Level: Developer-Project Dyad): *The probability that a developer joins a project is greater when the project initiator has a greater amount of preexisting collaborative ties with the open source software developer network.*

2.3. Developer Characteristics

As summarized earlier, developers participate in OSSD due to extrinsic motivations such as reputation, user needs, and learning benefits as well as intrinsic motivations such as enjoyment, altruism, and reciprocity. We propose that such motivations vary from individual to individual depending on a number of factors such as personality, belief, and skills levels although in this study we only consider the experience level of developers. The rationale behind this argument is that developers who are relatively new to the OSSD community are more likely to join newly-initiated projects in order to keep up their momentum to participate in OSSD and to confirm their value to the community. Hence, we propose the following hypothesis:

H4: (Developer Level: Developer-Project Dyad): *The probability that a developer joins a project is greater when the developer has less experience in the open source software developer network.*

2.4. Developer-Initiator Past Collaboration

OSSD has become a significant social phenomenon in which OSS developers and users form a complex social network via various electronic communication channels on the Internet. Therefore, we adopt a social network perspective to examine the OSSD team formation process.

Social network analysis aims to understand the relationships between people, groups, organizations, and other types of social entities (Granovetter 1973, Wasserman and Galaskiewicz 1994, Wellman and Berkowitz 1998), and has been used extensively in fields such as sociology (Cook and Whitmeyer 1992, Wasserman and Galaskiewicz 1994) and management (Borgatti and Foster 2003, Tsai 2001) among others (Huang and DeSanctis 2005, Singh 2005).

A social network is modeled as a graph with nodes representing the individual actors in the network and ties representing the relationships between the actors. The strength of a tie varies depending on a number of factors. Granovetter (1973) asserts that tie strength depends on the amount of time, the emotional intensity, the intimacy, and the reciprocal services associated with the relationship. Strong ties are characterized by a sense of special relationship, an interest in frequent interactions, and a sense of mutuality of the relationship (Walker et al. 1994). Strong ties maintain and promote trust and collaboration whereas weak ties enable actors to access resources and information that are unavailable in their immediate social circles (Burt 1992, Granovetter 1973).

In the context of OSSD, when developers decide whether to join a project, in addition to the intrinsic and extrinsic motivational factors that have been identified in prior research, factors related to the collaborative relationships between developers and initiators may become important.

In general, when forming teams people prefer to work with those with whom they have worked in the past because of the reduced uncertainty stemming from familiarity based on previous collaborative experiences (Hinds et al. 2000). Prior research also suggests that people are more likely to work together when they have prior social ties (McClelland et al. 1953, Schachter 1959). Software development teams composed of members with prior joint project experience may be more effective in coordinating programmers' distributed expertise because

they have developed knowledge of 'who knows what' (Moreland 1999). In the open source software development context in particular, due to the lack of face-to-face contacts, developers face greater barriers to effective communication and coordination and are thus more likely to be concerned about these issues. Previous collaborative relations with existing members of a project can mitigate these concerns due to the shared context and understanding accrued from prior interactions (Hinds et al. 2000, Moreland 1999). Therefore, OSS developers will be more likely to be attracted to projects initiated by developers with whom have collaborated in the past. Hence, we hypothesize

H5: (Developer Level: Developer-Project Dyad) *The probability that a developer joins a project is positively related to the existence of a prior collaborative tie with the initiator.*

In addition, not all past collaborative ties have the same strength. OSS developers are more likely to choose the same collaborator only if the past collaborations have been successful both in terms of final project outcome as well as in terms of the quality of the process. Research on group formation in laboratory and field study contexts have found that people are attracted to groups when their prior experiences with key group members have been positive and successful (Hinds et al. 2000, Zander and Havelin 1960). Thus, we hypothesize:

H6: (Developer Level: Developer-Project Dyad): *The quality and strength of the collaborative tie will moderate the relationship between the existence of a collaborative tie and the likelihood that the developer will join the project. More specifically, the likelihood that a developer will join a project will be greater when they share a positive, strong collaborative tie with the project leader than when the tie is weak.*

2.5. Task-Skill Fit Between Project and Developer

When faced with a newly-registered project and motivated by factors other than learning benefits, a developer may also consider whether he or she has the technical skills required by the project and how likely he or she can contribute to the development. However, the task-skill fit between the project and the developer is unlikely to influence the developer's joining decision is he or she intends to participate in the project mainly to learn new programming languages or new topic domains. Hence, we come up with the following hypotheses:

H7a: (Developer Level: Developer-Project Dyad): *The task-skill fit between a project and a developer has a positive influence on the probability that the developer will join the project.*

H7b: (Developer Level: Developer-Project Dyad): *The task-skill fit between a project and a developer has a negative influence on the probability that the developer will join the project.*

3. Research Methods

3.1. Data and Measures

Project and developer data were obtained from the dump of SourceForge.net's project databases currently hosted at the University of Notre Dame (http://www.nd.edu/~oss/). As the largest repository of open source applications on the Internet, SourceForge.net currently provides free hosting to more than 100,000 projects and more than 1,100,000 subscribers. It also offers a variety of services to hosted projects, including site hosting, mailing lists, bug tracking, message boards, file archiving, and other project management tools. SourceForge.net has been an attractive source of data for many researchers studying open source software mainly due to the abundance of publicly accessible data (Howison and Crowston 2004).

We selected all public OSS projects newly registered on Sourceforge.net between September 13, 2005 and October 14, 2005 (N = 1780). This was the sample for hypothesis testing at the project level. We revisited these projects on November 21, 2005 to capture the developers who had subsequently joined the project in the first 1 to 2 months. This process enables us to distinguish between the initiator and the developers who subsequently joined. Further, in order to identify the previous collaborative ties of the developers, we collected data on other projects that each developer had participated in prior to his/her joining the focal project to identify his/her past collaborators. Based on this data, we constructed affiliation matrices of developers and projects that depict the existence of the relationship ties between developers.

At the developer-project dyad level, hypothesis testing was conducted using a subset of the sample described above. Sample selection was based on availability of project and developer information required to operationalize measures of fit between developers' technical skills and project technical requirements. (The specific variables that we constructed for hypothesis testing are described in more detail below.) First, we included only those projects that explicitly defined technical details such as programming language, domain of software and operating system platform. Second, we restricted the sample of SourceForge.net developers to those who had

participated in at least one project by October 2005. This is because since only 2.3% of developers make their technical skill profiles accessible on SourceForge.net, we inferred the developers' technical skills from their past project experience. The final dataset used for hypothesis testing at the developer-project dyad level consists of 938 projects and 173,523 developers.

Dependent Variables

The final outcome of interest at the project level is to what extent a project has successfully attracted developers to join, which is operationalized as the number of developers joining a project within the target time frame (*NumJoiningDev*). At the developer-project dyad level the dependent variable is a binary indicator that captures whether a developer joins a particular project within the first 2 months of project initiation (*Join*).

Independent Variables

To test the influence of projects' goal definitions on the team formation process (H1), we included attributes of the project that indicate how clearly it is defined. These attributes include whether the project has defined its technical properties included in the OSSD community SourceForge software map *(TroveDefined),* and the level of details available in the project description that would facilitate information gathering required for making a joining decision (*DescDetail*).

To examine the impact of initiators' experience on developers' joining decisions (H2), we measured the experience level of project initiators using absolute participation duration at SourceForge.net (*InitiatorExperience_t*) as well as the number of projects in which the initiator has participated in the past (*InitiatorExperience_p).*

To test hypothesis H3 with regard to the number of previous collaborative ties that an initiator has, we constructed a measure of a project initiator's social capital and status based on the number of developers in the open source software development network with whom he/she has had previous collaborative ties with (*InitiatorTieAmount*) prior to project inception.

In order to investigate the role of developers' experience in the team emergence process (H4), we computed the experience of developers in terms of number of projects (*DeveloperExperience_p*) they participated in and total participation duration (*DeveloperExperience_t*) at SourceForge.net.

To test the effect of prior collaborative ties between a project initiator and a developer (H5 and H6), we captured the existence of such a tie using a binary indicator variable (*HasTie*). Because the quality and strength of the tie between the developer and project initiator may vary depending on the nature of the past collaborative experiences, that is, depending on the outcome quality of the project as well as the coordination process quality. In order to operationalize tie strength and quality, we characterized the nature of the past collaborations based on Granovetter's (1973) distinction between strong and weak ties and on measures of OSS project success (Crowston et al. 2003). For each developer-project initiator pair we constructed seven measures of joint project activity. The extent to which past collaborative experiences had been successful was measured by the average amount of code released (*AverageBytes*), the number of code downloads by users (*AverageDownloads*) and whether or not the project resulted in a successful release of the working program code (*AverageHasRelease*). The intimacy and amount of time spent in cultivating the tie was measured through the number and duration of the projects they collaborated on (*NumCollaborations, AverageDuration*) as well as the development status of the project (*AverageDevelopmentStatus*). Because we expect project administrators to interact more frequently in order to coordinate the project we also tracked whether the developer and project initiator had shared project administration responsibilities in prior projects (*BothAdminRole*).

We conducted exploratory factor analysis to determine the underlying latent constructs to measure the strength of a past collaborative tie. Two factors emerged from the analysis. The first factor, which we call *TieStrengthProduct*, represents tie strength based on whether or not the *outcome* of the past collaboration between the developer and the project initiator was positive, that is whether the project was successful as measured through number of download among other factors. The second factor, *TieStrengthProcess*, represents tie strength that is dependent on the *process* of coordination such as whether or not the developer collaborated as administrator with the project initiator in past collaborations. The measures were constructed using factor scores based on the factor loadings shown in Table 1.

Table 1. Composite Measures of Strength of Prior Collaborative Ties with Project Initiator

Variable	Factor 1 *TieStrengthProduct*	Factor 2 *TieStrengthProcess*
AverageDownloads	0.539	-0.232
AverageDuration	0.732	0.070
AverageHasRelease	0.703	-0.003
AverageDevelopment Status	0.376	-0.177
AverageBytes	0.283	0.053
BothAdminRole	-0.160	0.651
NumCollaborations	0.133	0.795

To test the influence of the task-skill fit between a project and a developer (H7a, H7b), we captured the fit in three variables that reflected whether the technical details in terms of topic (*MatchTopic*), programming language (*MatchProgLang),* and application platform (*MatchOS*) of any of the projects in which the developer had participated previously matched the details of the new project.

Control Variables

We also controlled for other project attributes that would influence developer joining decisions. These included the lifetime of the project (*Duration*), the popularity of the project application domain measured as a proportion of developers for the top 1000 projects who work on this domain (*TopicPopularity*), and whether the project is set up to accept donations from users (*AcceptDonation*). The summary of the measures computed for empirical analysis is shown in Table 2.

Table 2. Summary of Measures

Variable	Operational Definition
Project Level	
NumJoiningDev (DV)	The number of developers who joined the project within the first one/two months of project initiation (i.e., before Nov 21, 2005).
InitiatorTieAmount	The amount of collaborative ties that the project initiator has prior to project inception, It is calculated as the natural log of the number of distinct developers (+1) who have collaborated with the project initiator on OSS projects at Sourceforge.net.
InitiatorExperience_t	The project initiator's experience in terms of the natural log of the number of days since he/she registered on Sourceforge.net.
InitiatorExperience_p	The project initiator's experience in terms of the natural log of the number of prior projects (+1) that he/she has participated in on Sourceforge.net.
DescDetail	The level of details in project description as measured by the natural log of the number of characters in the project description.
TroveDefined	Indicator variable that is 1 if the project has been defined in the software map of Sourceforge.net, 0 otherwise.
AcceptDonation	Indicator variable that is 1 if the project has been set up to accept donations from users, 0 otherwise.
TopicPopularity	Popularity of the project's topic as measured by proportion of developers working on the top 1000 projects within the topic category.
Duration	Duration of the project's life (log of days)
Developer-Project Dyad Level	
Join (DV)	Binary variable, which equals 1 if the developer joined the project within the first one/two months of project initiation, 0 otherwise.
HasTie	Indicator variable that is equal to 1 if the developer has past collaborative tie with the project initiator, 0 otherwise.
TieStrengthProduct	The strength of a collaborative tie approximated by the product of the collaboration. See Table 1.
TieStrengthProcess	The strength of a collaborative tie approximated by the process of the collaboration. See Table 1.
InitiatorTieAmount	Same as project level.
InitiatorExperience_t	Same as project level
InitiatorExperience_p	Same as project level
DeveloperExperience_t	The developer's experience in terms of the natural log of the number of days since he/she registered on Sourceforge.net.
DeveloperExperience_p	The developer's experience in terms of the natural log of 1 plus the number of prior projects that he/she has participated in on Sourceforge.net.
MatchTopic	Indicator variable that is 1 if the developer's prior OSS projects' topics match the project's topic, 0 otherwise.
MatchProgLang	Indicator variable that is 1 if the developer's prior OSS projects' programming languages match the project's programming language, 0 otherwise.
MatchOS	Indicator variable that is 1 if the developer's prior OSS projects' operating systems match the project's operating system, 0 otherwise.
Duration	Same as project level
DescDetail	Same as project level
AcceptDonation	Same as project level.
TopicPopularity	Same as project level

3.2. Analytical Procedures

Project Level Analyses.

We used negative binomial regression to test the hypothesized effects of projects' definitions on *how* successful they are in attracting developers (i.e., *NumJoiningDev*).[2]

Developer-Project Dyad Level Analyses.

To test hypotheses at the developer-project dyad level, we adopted the technique of choice-based sampling or endogenous stratified sampling (King and Zeng 2001, Manski and Lerman 1977).[3] The strategy is to choose a fraction of the developer-project dyads representing the joining event and to choose a much smaller fraction of the non-event pairs. We used our sample of projects (N = 938) and selected all developers who have joined these projects as the event sample. In addition, we matched each dyad in the event sample with six control dyads as the control sample while ensuring that control sample has similar (and/or dissimilar) characteristics as the event dyads. In particular we controlled for the match between project requirement and developer skills (e.g., software topic, programming language, operating system) as well as the existence of prior collaborative social ties. In addition, two random dyads are selected for each event dyad. The choice-based sampling procedure produced a sample of approximately 3,800 dyads.

Corresponding to the choice-based sampling technique, we adopted the weighted exogenous sampling maximum-likelihood (WESML) estimator (Manski and Lerman 1977) as a validated approach adopted in prior literature (e.g., Singh 2005). The WESML estimator is calculated by maximizing the weighted pseudo-likelihood function that weighs each observation in the sample with the number of population observations that it represents. For example, the weight of a sample dyad that represents 100 potential dyads in the entire population is 10 times

[2] Although Poisson regression is a common analysis technique when the dependent variable has only non-negative integer values, our data suffers from problems of overdispersion (Deviance/DF = 1.914, Pearson χ^2/DF = 4.246; LR statistic = 1316.479, $p < 0.01$). Because many observations in our dataset have a value of zero for the count variable, a negative binomial regression is more appropriate (Allison 1999).

[3] For our sample, conventional logistic regression approach with random sampling is impractical due to the rarity of a developer's project joining event. For instance, with approximately 1,000 sample projects and 170,000 sample developers, there would be over 170 million (i.e., 1,000 × 170,000) developer-project dyads in total. However, of those possible dyads, there are only a very small percentage of dyads representing the event that a developer joined a project. Thus, pure random sampling from all possible dyads would make the sample size impractically large and lead to biased statistical estimation.

the weight of a sample dyad that represents 10 population dyads[4]. In addition, because the same developer may be included in multiple project developer dyads, we calculated the standard errors without assuming independent errors among observations.

In order to investigate the robustness of the estimation with respect to the choice-based sampling procedure, we drew 1000 bootstrap choice-based samples to derive the bootstrap mean and the confidence intervals for each parameter estimate.

4. Results

4.1. Project Level Analysis

The project sample descriptive statistics are shown in Table 3. Table 4 presents the pairwise correlations of the measures. The correlation between *InitiatorTieAmount* and *InitiatorExperience_p* is moderately high ($\rho = 0.642$, $p < 0.001$). However, further collinearity diagnostics show that the tolerances of all predictor variables are above 0.4 and that the highest variance inflation factor value is 2.123, indicating that multicollinearity is not a major concern in the analysis.

Table 3. Descriptive Statistics (Project Level)

	Descriptive Statistics			
Variable	**Mean**	**St. Dev**	**Min**	**Max**
NumJoiningDev	0.55	1.963	0.00	40.00
InitiatorTieAmount	0.44	0.930	0.00	6.01
InitiatorExperience_t	4.76	1.962	0.40	7.68
InitiatorExperience_p	0.45	0.606	0.00	3.04
DescDetail	5.00	0.600	2.30	5.92
TroveDefined	0.53	0.500	0.00	1.00
AcceptDonation	0.05	0.218	0.00	1.00
TopicPopularity	0.12	0.074	0.00	0.62
Duration	3.96	0.169	3.62	4.22

[4] For more technical details on WESML refer to King and Zeng (2001).

Table 4. Correlations (Project Level)

Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) NumJoiningDev								
(2) InitiatorTieAmount	0.200***							
(3) InitiatorExperience_t	0.026	0.423***						
(4) InitiatorExperience_p	0.082***	0.642***	0.577***					
(5) DescDetail	0.010	0.009	0.025	0.018				
(6) TroveDefined	0.020	0.025	0.130***	0.079***	0.091***			
(7) AcceptDonation	-0.023	-0.018	0.050**	0.009	0.053**	0.135***		
(8) TopicPopularity	-0.010	0.032	0.108***	0.067***	0.093***	0.224***	0.055**	
(9) Duration	0.003	-0.007	0.144***	0.001	0.026	0.137***	0.051**	0.027
Significance levels: *** 0.01, ** 0.05, * 0.1								

We hypothesized that the better-defined a project the more additional developers it would attract to join the team (H1). We tested this hypothesis by estimating the parameters for the following negative binomial regression model:

$$\log E(y) = \ln(Duration) + \alpha + \beta_1 InitiatorTieAmount + \beta_2 InitiatorExperience\_t + \beta_3 InitiatorExperience\_p + \beta_4 DescLength + \beta_5 TroveDefined + \beta_6 AcceptDonation + \beta_7 TopicPopularity + \sigma\varepsilon$$

The results are summarized in Table 5. The parameter estimate for *TroveDefined* is positive and significant, indicating that whether the project has defined its technical properties positively influences the number of additional developers who join the team. The parameter estimate for *DescDetail* is positive but not significant. Overall, the results show partial support for hypothesis H1.

Table 5. Project-Level Regression Results

Variable	Parameter Estimate
Constant	-4.285***
InitatorTieAmount	0.848***
InitiatorExperience_t	-0.134***
InitiatorExperience_p	-0.500***
DescDetail	0.004
TroveDefined	0.262**
AcceptDonation	-0.307
TopicPopularity	-1.016
Duration	
Dispersion	4.008
Model Statistics	
Sample Size (*N*)	1780
Deviance	0.579

Significance levels: *** 0.01, ** 0.05, * 0.1

4.2. Developer-Project Dyad Level Analysis

Table 6 summarizes the descriptive statistics and Table 7 presents the pairwise correlations of the measures for the developer-project dyad sample. The highest correlation among the independent variables is between *InitiatorTieAmount* and *InitiatorExperience_p* ($\rho = 0.652$, $p < 0.001$).

Table 6. Descriptive Statistics (Developer-Project Dyad Level)

Variable	Mean	St. Dev	Min	Max
Join	0.14	0.351	0.00	1.00
HasTie	0.02	0.142	0.00	1.00
TieStrengthProduct	-0.01[a]	0.164	-2.56	3.32
TieStrengthProcess	0.01[a]	0.226	-1.06	8.57
InitiatorTieAmount	0.56	1.041	0.00	5.64
InitiatorExperience_t	1.96	1.179	0.09	4.12
InitiatorExperience_p	0.52	0.632	0.00	2.77
DeveloperExperience_t	6.48	1.163	1.25	7.69
DeveloperExperience_p	0.83	0.337	0.00	3.09
MatchTopic	0.14	0.351	0.00	1.00
MatchProgLang	0.15	0.354	0.00	1.00
MatchOS	0.19	0.395	0.00	1.00
Duration	3.97	0.164	3.62	4.22
DescDetail	5.05	0.547	2.94	5.92
AcceptDonation	0.08	0.273	0.00	1.00
TopicPopularity	0.13	0.083	0.00	0.62

Notes: [a] For those developer-project dyads without ties *TieStrengthProduct* and *TieStrengthProcess* are coded as 0.

Table 7. Correlations (Developer-Project Dyad Level)

Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) Join								
(2) HasTie	0.159***							
(3) TieStrengthProduct	-0.124***	-0.574***						
(4) TieStrengthProcess	0.122***	0.417***	-0.070***					
(5) InitiatorTieAmount	0.189***	0.184***	-0.079***	0.061***				
(6) InitiatorExperience_t	0.010***	0.117***	-0.055***	0.047***	0.387***			
(7) InitiatorExperience_p	0.055***	0.142***	-0.075***	0.063***	0.652***	0.537***		
(8) DeveloperExperience_t	-0.602***	-0.002	0.041***	0.001	-0.135***	-0.000	-0.038	
(9) DeveloperExperience_p	-0.131***	0.141***	-0.059***	0.091***	-0.052	0.003	0.001	0.352***
(10) MatchTopic	-0.015***	0.198***	-0.100***	0.106***	0.020	0.029***	0.019	0.117***
(11) MatchProgLang	-0.005***	0.206***	-0.097***	0.107***	0.025***	0.040***	0.024*	0.106***
(12) MatchOS	-0.067***	0.119***	-0.022*	0.072***	0.007	0.021	0.022	0.172***
(13) Duration	-0.032***	-0.010***	0.008	-0.005	-0.137***	-0.047***	-0.096***	0.050
(14) DescDetail	0.007***	0.002	0.000	-0.011*	-0.043***	-0.037***	-0.033	-0.001***
(15) AcceptDonation	-0.038***	-0.005**	-0.008	0.004	-0.049***	0.024***	-0.012	0.025
(16) TopicPopularity	0.002	0.024	0.003	0.032**	0.021	0.082***	0.021	0.006

Variable	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)
(1) Join								
(2) HasTie								
(3) TieStrengthProduct								
(4) TieStrengthProcess								
(5) InitiatorTieAmount								
(6) InitiatorExperience_t								
(7) InitiatorExperience_p								
(8) DeveloperExperience_t								
(9) DeveloperExperience_p								
(10) MatchTopic	0.256***							
(11) MatchProgLang	0.220***	0.216***						
(12) MatchOS	0.248***	0.212***	0.305***					
(13) Duration	0.039***	0.006***	-0.002***	-0.004***				
(14) DescDetail	0.005	0.037*	0.013	0.019	0.047*			
(15) AcceptDonation	0.018*	0.013	0.002	0.012	0.030*	0.071*		
(16) TopicPopularity	0.007	0.186***	0.019	0.054**	-0.011**	0.121***	0.018**	

Significance levels: *** 0.01, ** 0.05, * 0.1

At the developer-project dyad level, we hypothesized that the experience level of the initiator (H2, β_2, β_3), the number of project initiator's collaborative ties (H3, β_4), collaborative tie between the developer and the project initiator (H5, β_1), and the strength and quality of the collaborative tie (H6, β_5, β_6) would have a positive impact on developer decisions to join a project. We also hypothesized that the experience level of the developer (H4, β_7, β_8) would have a negative influence on developers' joining decisions. In addition, we proposed hypotheses regarding the impact of task-skill fit (H7a, H7b, $\beta_9, \beta_{10}, \beta_{11}$). We used WESML to estimate the parameters for the following logistic regression model:

$$\begin{aligned}\text{logit}\left(\Pr(y=1)\right) = {} & \alpha + \beta_1 HasTie + \beta_2 \ln(InitiatorExperience\_t) + \beta_3 \ln(InitiatorExperience\_p) \\ & + \beta_4 \ln(InitiatorTieAmount) + \beta_5 HasTie * TieStrengthProduct \\ & + \beta_6 HasTie * TieStrengthProcess + \beta_7 DeveloperExperience\_t \\ & + \beta_8 DeveloperExperience\_p + \beta_9 MatchTopic \\ & + \beta_{10} MatchProgLang + \beta_{11} MatchOS + \beta_{12} Duration + \beta_{13} DescDetail \\ & + \beta_{14} AcceptDonation + \beta_{15} TopicPopularity + \varepsilon\end{aligned}$$

The results of the logistic regression are presented in Table 8 (Model 4). The variables capturing the experience level of the project initiator have negative and significant coefficient estimates, indicating that OSS developers prefer to join projects initiated by newcomers. Hence, H2 is not supported. The parameter estimate for *InitiatorTieAmount* is positive and significant, showing support for H3 and indicating that developers are more likely to join a project whose initiator is more embedded in the network. The variables *DeveloperExperience_t* and *DeveloperExperience_p* have negative and significant parameter estimates, suggesting that new developers are more likely to join newly-initiated projects. Therefore, hypothesis H4 is supported. Moreover, the variable *HasTie* has a significantly positive impact on the likelihood of developer joining ($\beta_1 = 7.244$, $p < 0.01$), suggesting that a developer is far more likely to join a project that has been initiated by a past collaborator whom they are familiar with (H5 is supported). Furthermore, the interaction term between *HasTie* and *TieStrengthProcess* has a positive and significant estimate whereas the interaction term between *HasTie* and *TieStrengthProduct* is not significant, indicating that the past collaboration process itself has a greater moderating influence on a developer's future joining decisions than the successful production of software (H6 is partially supported). With respect to the technical fit between the project and the developer, the parameter estimates for all three relevant variables are not significant, not supporting our hypothesis H7a or H7b.

In summary, the results indicate that both the existence and the process-related quality and strength of collaborative ties between the developer and the project originator positively impact the likelihood of the developer joining the project. They also suggest that developers' experience has a negative impact on their joining behavior. Although initiators' experience in terms of duration and project is not a significant predictor, their network embeddedness does have a positive and important influence on the project's probability of attracting more developers.

Combining the project-level and developer-level results, we conclude that the OSSD team emergence process is driven by the project's goal definition, its initiator's network embeddedness, developers' experience level, and the past collaborative ties between the initiator and developers.

4.3. Post-Hoc Analyses of Project Joining Decisions

We performed additional exploratory analyses to investigate whether the same pattern of results can be observed in terms of the joining decision regardless of the experience level of developers. Experienced developers may have a different set of motivations for OSSD project participation and thus may employ a different decision calculus when deliberating the choice of project to join. We thus divided the sample based on developers' past project experience and performed the logistic regression analysis omitting the developer project experience variable. Model 5 in Table 8 shows the results for the developer-project dyads in which the developer has participated in one project in the past (i.e., less experienced developers) whereas Model 6 is associated with the dyads in which the developer has worked on more than one projects (i.e., more experienced developers). Results are presented in Table 8.

The parameter estimates for *HasTie* and *HasTie×TieStrengthProcess* are still significantly positive for experienced developers (Model 6). However, they are no longer significant predictors for inexperienced developers (Model 5). In addition, there are some significant differences in the parameter estimates between the two models. For example, estimates for *MatchTopic* and *MatchOS* are both negatively significant in Model 5, suggesting the presence of possible learning effect whereas those for *MatchTopic* and *MatchProgLang* are both significantly positive in Model 6, suggesting that the experienced developers tend to join projects related to the topic domains and programming languages that they are familiar with. Overall the results indicate that the factors influencing experienced developers' joining decisions are quite

different from those influencing inexperienced developers' decisions. Inexperienced developers seem to choose which project to join mainly based on potential learning benefits whereas experienced developers tend to make the joining decision based on their past relations with project initiators as well as their personal interests in software domains and programming languages.

Table 8. Logistic Regression Results (Developer-Project Dyad Level Analysis)

	Model 4		Model 5		Model 6	
Variable	**Parameter Estimate**	**Odds Ratio**	**Parameter Estimate**	**Odds Ratio**	**Parameter Estimate**	**Odds Ratio**
Constant	-2.521		6.509		-6.515*	
HasTie	7.244***	1399.211	5.922	373.031	7.545***	1890.40
HasTie×TieStrengthProduct	-0.122	0.885	-1.135	0.321	-0.491	0.612
HasTie×TieStrengthProcess	1.370***	3.934	2.366	10.653	0.880**	2.411
InitiatorTieAmount	0.124***	1.131	0.162***	1.176	0.141***	1.151
InitiatorExperience_t	-0.092***	0.912	-0.123	0.884	0.022	1.022
InitiatorExperience_p	-0.462**	0.630	-0.528	0.590	-0.978***	0.376
DeveloperExperience_t	-1.611***	0.200	-2.537***	0.079	-1.416***	0.242
DeveloperExperience_p	-1.156***	0.315			0.458	1.580
MatchTopic	0.360	1.434	-2.134***	0.118	0.969***	2.636
MatchProgLang	0.224	1.251	-0.885	0.413	0.794***	2.212
MatchOS	0.039	1.039	-1.466***	0.231	0.375	1.455
Duration	0.105	1.111	-0.881	0.414	0.571	1.770
DescDetail	0.092	1.096	0.005	1.005	0.292	1.339
AcceptDonation	-0.435	0.647	-0.411	0.663	-0.497	0.608
TopicPopularity	-0.297	0.743	0.121	1.129	-0.571	0.565
Significance levels: *** 0.01, ** 0.05, * 0.1						

5. Exploratory Analyses of Subsequent Project Performance

Next we conducted additional exploratory analyses to examine the project performance implications of OSSD project team formation process. Specifically, we tried to answer two follow-up questions. First, do projects started by a developer with many collaborative ties in the OSS developer network perform better? Second, does the developer joining behavior in a project's early stage impact its performance?

We identified three measures of project performance – *HasRelease*, *ProjectScore*, and *DevelopmentStatus* that respectively measured project output, level of development activity including communication and project administrator's self-reported project development phase,[5] which we assessed 7-8 months after project initiation with the June 2006 dump of the SourceForge.net project database. After dropping projects that were no longer active, we

[5] See Crowston et al. (2003) for other measures of OSS success.

analyzed the effect of group composition as well as the number of developers who had joined the project in the first 2 months of project inception (*NumJoiningDev*) on success in 1775 projects. Initial exploratory results presented are in Table 9.[6]

Table 9. Project Performance Regression Results

Dependent Variable	***HasRelease***		***ActivityScore***	***DevelopmentStatus***	
Regression Framework	**Logistic**		**Linear**	**Cumulative Logit**	
Variables	**Parameter Estimate**	**Odds Ratio**	**Parameter Estimate**	**Parameter Estimate**	**Odds Ratio**
Constant	-3.667		50.154		
NumJoiningDev	-0.001	0.999	0.381***	0.053*	1.054
InitatorTieAmount	-0.131*	0.877	0.091	0.123	1.131
InitiatorExperience_t	-0.003	0.997	0.334**	0.024	1.025
InitiatorExperience_p	0.276**	1.318	2.083***	-0.268**	0.765
DescDetail	0.291***	1.337	1.160**	-0.277***	0.758
TroveDefined	1.127***	3.086	4.388***	0.438***	1.550
AcceptDonation	0.515**	1.674	1.443	-0.107	0.898
TopicPopularity	-0.807	0.446	-2.657	0.093	1.097
Duration	0.250	1.284	-9.443	-2.497	0.082
Intercept (Planning)				13.238	
Intercept (Pre-Alpha)				14.094*	
Intercept (Alpha)				14.977*	
Intercept (Beta)				16.251*	
Intercept (Stable)				19.218**	
Model Statistics					
Sample Size (*N*)	1775		1775	1023[a]	
Likelihood Ratio (χ^2)	165.554***			26.218***	
F Statistic			24.990***		
Adj. R^2			0.109		

Significance levels: *** 0.01, ** 0.05, * 0.1

Notes: [a] The sample size is 1023 because 752 sample projects had not defined their development status by the time of our collection of project performance data.

A comparison of these results shows that the role of a project initiator's social ties does not seem to be significantly important. However, *NumJoiningDev* is a significant predictor of two performance measures. Given our earlier results that *InitiatorTieAmount* does favorably influence the developer joining behavior with a project's first 1 to 2 month, we propose that although *InitiatorTieAmount* does not directly impact the project performance it may does so

[6] Given the short period of time since the inception of the projects and the assessment of those projects' performance (i.e., Sept-Oct 2005 through June 2006), we present the initial exploratory analysis for discussion purposes. These results are not meant to be used for drawing any definitive conclusions.

indirectly. Another interesting finding is that the project performance in its first 7 months seems to depend on the initiator's experience as well as how well the project is defined.

However, because of the relatively short period between a project's start and our performance data collection, the above results should be interpreted with caution. The performance data are taken from the latest snapshot of the projects. A higher performance in our dataset is unlikely to guarantee the definitive future success of the project. Nonetheless, our exploratory study in this section can shed some light on the factors that may have performance implications during the early phase of the project.

6. Conclusion

In this study we examined the driving forces behind the self-organizing process of OSSD team formation. Specifically, we investigated the influence of project properties, project initiator characteristics, developer characteristics, past collaboration between the initiator and the developer, and the task-skill fit between the project and the developer on developers' joining behavior. Our results indicate that both the existence and the process-related strength of collaborative ties between the developer and the project originator positively impact the likelihood of the developer joining the project. Although initiators' experience in terms of duration and project is not a significant predictor, their network embeddedness does have a positive and important influence on the project's probability of attracting more developers. We also found that developers' experience has a negative impact on their joining behavior. Overall, the results suggest that the emergence of OSS project teams embedded in a collaborative network of developers is influenced by the relations formed by developers' past collaborations.

One practical implication of our study is that commercial actors interested in involving OSS developers in their projects may benefit from actively cultivating relationships with existing developers by among other means actively contributing in existing OSSD projects. In addition, this paper finds OSS developers choose to continue collaborating with people with whom they have had positive project experiences, providing empirical evidence for one mechanism that may explain the formation of scale-free networks prevalent in digital and social networks. This research also contributes to our understanding of self-organizing groups by identifying the factors influencing individuals' self-selection into groups, which has received limited amount of attention in the existing literature.

Next we identify some limitations of the study and additional directions for future research. One limitation may be that we only examine joining behavior within the first two months after project registration. The joining behavior may differ during different stages of project development. It would be interesting to look at whether the factors influencing developer joining decision change over time as a project goes through different stages. Another important extension of this paper would be to study the effect of developer joining behavior on the coordination mechanisms adopted by the OSSD team. Furthermore, our data only includes information available from SourceForge.net. Even though SourceForge.net is currently the largest repository, constraining the data collection to one site only may introduce measurement error. For example, even if a developer may have had extensive OSSD experience outside of SourceForge.net, if she joins her first project hosted herein, she would be considered inexperienced. We plan to extend our data collection to other OSS project hosting sites to reduce measurement error and make the results more generalizable. In addition, further research needs to examine the variations in group composition of OSSD teams in terms of attitudes, ideological beliefs regarding free and open source software, gender and occupational background among others as a result of the preferential attachment of OSS developers.

CHAPTER 3. AN EMPIRICAL INVESTIGATION OF EARLY ADOPTION OF OPEN SOURCE SOFTWARE INNOVATIONS

1. Introduction

The success of some well-known open source software (OSS) such as the Apache web server and the Linux operating system has demonstrated the attractiveness of open source software development (OSSD) as an alternative model of software production. A few researchers have identified OSSD as a fundamentally different way of organizing innovation production that depends on the collective involvement of community members (Benkler 2002, Franke and Shah 2003, von Hippel and von Krogh 2003). OSSD can be viewed as an example of decentralized innovation diffusion system, in which innovations come from users and peer users decide whether to adopt and diffuse innovations.

Some researchers have investigated the organization of the OSSD community based on the roles played by its members (Maas 2004, Moon and Sproull 2002, Nakakoji et al. 2002, Xu et al. 2005). Drawing from these prior work, in this paper we group the members of a OSS project community into three categories, namely core members, peripheral members, and passive users. Core members are the developers formally listed as the members of the project team. They develop code, maintain code, and interact with peripheral members. A project is likely to be abandoned when its core members either lose their interest in participating or are no longer available to contribute their time and efforts. Peripheral members are the individuals who are not listed as formal members of the project team but who participate frequently or infrequently by submitting bug reports, posting messages, and contributing code. They can be either software developers or end users. Passive users are the individuals who are interested in the project but do not actively participate in the communication activities. They download and use the software for various reasons, among which are personal software need and learning effects. Usually when a project is first initiated, there is only one core member - the initiator. Subsequently other developers join the team and become core members. After code becomes available, enabling people to take a closer look at the project, passive users begin downloading the software, of whom some may choose to contribute to the development at a later time.

Attracting user adoption plays an important role in a successful decentralized software innovation process. First and foremost, because developers volunteer to participate in a project without monetary incentives, many of them need to confirm the relevance and importance of their work based on the feedback from users. Few developers want to spend time and effort on a project that no one is interested in. Secondly, those individuals who actually try the software are more likely to become peripheral and even core members of the project. Raymond (2001) stresses the importance of having users because users not only demonstrate the relevance of developers' efforts but also accelerate the software debugging and improving process by becoming co-developers. Thirdly, because many success measures for proprietary software such as being within-budget and being delivered on-time are not applicable to open source software due to developers' volunteer participation and lack of delivery deadlines requested by users in OSSD, a new set of performance measures have been proposed and used in the existing open source literature. Some often used measures for OSS success are project popularity among users and developers, project vitality (Stewart et al. 2005), number of downloads (Grewal et al. 2006),

and size of the development community (Crowston et al. 2003), all of which are directly related to user interest and user adoption.

However, OSS projects exhibit a significant amount of heterogeneity in terms of user interest and adoption. Some popular projects have hundreds of thousands project page views in a week whereas between 20% and 30% of all projects hosted at SourceForge.net have zero weekly page views. The number of mailing list subscribers ranges from zero to thousands. And the total number of cumulative software downloads ranges from fewer than 100 to hundreds of millions. Krishnamurphy (2002) also points out that a large number of OSS projects have very small development communities and only a small number of projects have large communities.

So what determines the huge heterogeneity in user interest and adoption? Quite a few studies have attempted to examine the performance differences of OSS projects in these aspects by analyzing data aggregated over projects' entire life span and have found that project network embeddedness, software license choice, and organizational sponsorship have significant impacts on project success (Grewal et al. 2006, Stewart et al. 2005). An OSS project usually takes several years to develop a mature software product, during which the development goes through various stages that lead to fluctuating user interest and adoption of software. Such a fluctuating pattern is difficult to be revealed from an aggregate perspective because a surge in user adoption during one period of time may be cancelled out by a drop in user adoption during another time period. Therefore, we argue that analyzing data collected during a specific time period rather than cumulative data can yield deeper insights into the dynamic OSSD process. Specifically we undertake a study of early user adoption behavior of OSS projects' first software releases for the following reasons.

First, we limit the scope of this paper to the one-month period following the event of projects' first software release because first release is a critical event for many OSS projects. Although it may have many defects and missing features, the first version of the code often outlines the functionalities envisioned by the project initiator, demonstrating the potential merits as well as the feasibility of a project (Lerner and Tirole 2002). Project statistics often show a steep increase in the amount of attention received by the project shortly after its first release. Our preliminary analysis shows that the magnitude of project traffic shortly after the project's initial release is highly correlated with the magnitude of project traffic in the subsequent phases of the project.

Second, projects often start showing enormous differences along several dimensions such as the amount of traffic they receive, discussion activities, and code development activities in their early stages. We argue that such differences can be traced back to the number of downloads of their initial releases, which can be viewed as an approximate indicator of the amount of user interest or the number of potential users. von Krogh et al. (2003) also suggest that the download statistics may indicate the size of the potential developer base that can be mobilized for future development. From the innovation diffusion perspective, open source software innovations typically begins diffusing among users after the first version of code becomes available to the public. It is important to understand how users react to these innovations and why they are attracted to some projects but not the others. Therefore, we choose to examine the user adoption behavior after projects have their initial releases.

In this research we try to answer the following questions: when the first version of open source software becomes available for downloads what factors impact users' adoption behavior? Does users' early adoption behavior impact the future performance of a project? The answers can help OSS project managers attract initial user interest in the project, which may enable the project to gain a critical mass of users and possibly acquire a self-reinforcing impetus for growth. This study can also further our understanding of the diffusion process of decentralized innovations by identifying its differences and similarities with centralized innovation diffusion.

2. Theoretical Development

In this section we first introduce and compare centralized innovation and decentralized innovation. Then we discuss why innovation visibility is especially important for OSS projects and how they can promote their visibility. Next we identify the innovation properties that impact users' early adoption. We conclude the section with our research framework.

2.1. Centralized Innovation vs. Decentralized Innovation

There are two types of innovation-development processes: centralized and decentralized. A centralized innovation-development process follows a top-down approach from technical experts to consumers. The driving force behind this process is technological advances that make an innovation available. An innovation starts with a technical expert or an organization recognizing the need for a new product. After formal research and development activities,

technical experts and administrators decide whether to diffuse the innovation. Then a change agent pushes the commercialized innovation to potential adopters and acts as the communication channel between adopters and inventors. One major role of the change agent is to ensure the flow of innovations from the innovator to the clients. In addition, the change agent acts as a filter by selectively transmitting to the clients only the relevant information to deal with possible information overload. The diffusion system mainly depends on the one-way communication from the change agent to users. Given an innovation, an individual consumer makes the decision whether or not to adopt it. As the innovation diffuses, it goes through relatively low level of adaptation based on consumer needs.

In contrast, a decentralized innovation-development system is not run by a small number of technical experts or by government administrators. The process starts with an innovator who is also a user and diffuses among peer users. After recognizing their needs and problems, local innovators invent and develop an innovation to solve their problems. Then the innovation is diffused to other peer users through horizontal networks consisting of users and other local innovators. The diffusion process relies on the two-way communication among peer users. Users decide which innovations to adopt based on their individual opinions and preferences. During the diffusion process, the innovation undergoes a high level of adaptation and reinvention to tailor to various user needs.

With respect to software innovation process, traditional software innovation-development is a centralized process. Major decisions, such as innovation creation, R&D, commercialization, and marketing, are made by commercial software companies. Open source software development emerged as an alternative and viable decentralized innovation-development approach. A software product is initiated and developed by local users to solve their own problems. At the time of initial development of the software, the inventors are unaware of the needs and problems of their peer users. As the software circulates among users, its developers and other developers make modifications based on feedback from users.

2.2. Innovation Awareness

Rogers (1983) identifies that the innovation-decision process consists of five stages, namely knowledge, persuasion, decision, implementation, and confirmation. The first stage starts when an individual becomes aware of an innovation and how it works. Generally speaking, there are three types of knowledge about an innovation – awareness knowledge, how-

to knowledge, and principles knowledge. Awareness knowledge is associated with the information that an innovation exists. It usually motivates individuals to seek how-to knowledge, information about how to use the innovation, and principles knowledge, information related to the underlying principles of how the innovation works (Rogers 1983).

During the diffusion process of centralized innovations, a change agent often provides the information related to new ideas to consumers in order to create adequate awareness of innovations rapidly, which then leads to a faster adoption process. In the decentralized innovation process, the change agent who filters and passes relevant information to consumers in a centralized setting does not exist any more. Consequently, users have to process a large amount of information related to many innovations that try to diffuse in the same network of peer users, which may lead to users' information overload. As a result, they may not be aware of all existing innovations. We propose that the first challenge a decentralized innovation developed by a local user has to face in its diffusion process is to gain initial awareness among peer users instead of becoming unnoticed and invisible among other innovations.

In the OSSD context, the difficulty of obtaining project visibility among users is still present in spite of some popular project hosting sites with millions of subscribers. Currently there exist a number of hosting sites such as SourceForge.net and FreshMeat.net for open source software projects. These sites not only provide project managers with a platform and tools needed to create and manage innovative software products but also create an online collaborative community of developers and users from diverse background and locations. Utilizing the hosting services is beneficial to OSS projects in that projects may be able to tap into the large resource pool of potential users and developers in the entire community. However, negative network externality may occur as the number of projects hosted at a site increases. A project, especially a relatively new project, tends to become less visible to potential users as more projects compete for user attention and users become overwhelmed with the large selection of possible interesting projects.

Next, we identify the possible ways that an OSSD project can overcome the difficulty of attracting initial user awareness.

First, a project manager can play a similar role as that of a change agent in centralized innovation process by pushing project information to users through mass communication channel, which allows the project to spread messages to a large audience quickly.

Second, some scholars in traditional innovation diffusion maintain that individuals gain awareness of an innovation through *active* information seeking behavior after they feel a need for the innovation (Hassinger 1959). Users in a decentralized network are more likely to engage in active information seeking behavior to cope with the possible information overload. They tend to pull information about projects until they find a project that he or she may be interested in. Thus, the easier a project is to search for, the more users may find it and become aware of it.

Third, a project is not isolated from the other projects. Instead, it is embedded in a network of projects. Projects are related to each other in various aspects. A project may inherit code from an earlier project that is no longer under active development. Consequently, users of the earlier project may become aware of the new project. Another possibility is that a project forks from an existing and active project and attracts some users from it as well. In addition, projects are connected by having common developers in the development team. If project A shares a common developer with project B, B's users may become aware of A because of that developer. Grewal et al. (2006) find that network embeddedness of projects has significant effects on their technical and commercial success.

Last, a project may rely on interpersonal communication channels to gain user awareness. In other words, its core developers may spread the information to other developers through their direct or indirect contacts in the network. Developers become collaborators who are connected to each other through participating in projects. They may exchange information related to the various projects they work on. A project whose developers have many collaborators in the entire network of developers is more likely to obtain awareness than a project whose developers have few collaborators.

After users become aware of an OSS innovation by obtaining relevant information, they make the decision whether to try and adopt the software or to ignore it. In the next subsection, we discuss the factors influencing users' adoption decisions.

2.3. Innovation Adoption

In this subsection, after identifying and discussing the differences between our study and prior work, we develop our theoretical framework of OSS early adoption by modifying the traditional innovation theory of innovation properties and adoption process.

Researchers in traditional, centralized innovation diffusion have identified some innovation attributes that influence the S-shaped adoption curve of the innovation. Rogers (1983) suggests

that individuals' perception of an innovation's relative advantage, compatibility, complexity, trialability, and observability affects the adoption rate. However, the scope of this paper differs from prior work examining the influence of product attributes on innovation adoption in that we mainly investigate users' adoption of the first software release of OSSD projects. In other words, our focus is not the entire adoption process but early adoption, a short segment of the adoption curve. Consequently, the type of adopters in this study also differs from that in prior literature.

We argue that the users who download and try the first version of OSS tend to be software developers rather than end users. End users tend to adopt OSS that is relatively stable and mature to fulfill their software needs. However, the first OSS release is only a preliminary product that has bugs and flaws. Moreover, the innovator may not know about end users' needs because he or she usually starts a project to scratch "a personal itch" (Raymond 2001). Thus the first release may not be able to satisfy end users' needs.

Software developers are interested in the early version of OSS for several possible reasons. First, they are venturesome and keen on exploring new ideas of other developers. They may also take pleasure in reading other people's code and learning different coding styles. Second, they try and possibly explore the code in order to become a contributor to the project later. Shah (2006) identifies two types of contributors in the open source community, need-driven individuals and hobbyists. Need-driven individuals contribute code due to reciprocity, possibility of code improvement, desire to add their own code to the existing software, and career concerns. In contrast, hobbyists contribute code to gain feedback from others and confirm the importance of their participation. Given the motivations of the early adopters of OSS, we examine whether the product characteristics identified in traditional diffusion theory may still be significant predictors of the adoption rate.

- *Relative advantage.* It refers to the extent to which an innovation is perceived to be better than the ideas preceding it. It can be viewed as the ratio of the expected benefits and the costs of adopting an innovation. The cost of downloading and installing an OSS product is low given the availability of the software and the low cost of required hardware and infrastructure. However, due to the nature of software as an experience and intangible product, it is difficult for users to perceive the expected benefits of using the

software prior to the actual trial. Therefore, we exclude this predictor from our theoretical model.

- *Compatibility*. It refers to the degree to which an innovation is perceived to be consistent with potential adopters' values, beliefs, past experiences, and needs. Prior literature suggests that the perceived compatibility of an innovation is positively associated with its rate of adoption. We hypothesize that the same argument holds for early adoption of OSS provided that compatibility is more technical- and legal-oriented. The subdimensions of compatibility in existing theory such as values, beliefs, past experiences, and needs may not be a concern for early OSS adopters. Instead, these adopters tend to consider technical and legal factors such as whether the software will work with their existing hardware and software as well as whether they have the legal rights to modify and distribute the code (Stewart et al. 2005).

H1a: Technical compatibility is positively related to the early adoption rate of OSS.

H1b: Legal compatibility is positively related to the early adoption rate of OSS.

- *Complexity*. *Complexity* is how difficult an innovation is perceived to be understood and used. According to traditional diffusion theory, the perceived complexity of an innovation is negatively related to its rate of adoption because the majority of adopters do not have a high level of technical expertise. However, the early adopters of a software product tend to be tech savvy developers who have more experience and expertise than the average users. They try the software partly to satisfy their intellectual curiosity and to learn about other developers' work. We propose that software complexity is positively related to its early adoption because complex software seems more interesting and challenging to pioneer OSS adopters.

H2: Complexity is positively related to the early adoption rate of OSS.

- *Trialability*. *Trialability* is the extent to which an innovation can be tried without significant investment. According to the existing literature, if a traditional innovation can be designed to be tried more easily, it will have a more rapid rate of adoption. Because open source software can be downloaded free of charge and installed with few limitations, trialability to some extent depends on the easiness of downloading and installing the software. Although the same argument that higher trialability leads to faster adoption may still hold for the entire adoption process of open source software, easiness

of downloading and installing may not be a major concern for pioneer adopters who are usually software developers with technical knowledge. Therefore, we develop the following hypothesis regarding *trialability* of OSS:

H3: Trialability does not have a significant impact on the early adoption rate of OSS.

- *Observability*. It refers to the degree to which the results of an innovation can be observed and communicated to others. The use of software by adopters is not widely observable to other potential adopters. Potential adopters cannot easily observe the benefits obtained by those using the software. Hence, observability is not applicable in the OSSD context.

Given the characteristics and motivations of early OSS adopters, we identify two additional dimensions of a project that may be associated with its early adoption.

- *Anticipated influence of project*. One of the reasons that users choose to download the first open source software release is for future participation in the project. They want to contribute to the project partly due to reciprocity, desire to add their own code to the existing software, career concerns, and to gain feedback confirming the importance of their contribution, all of which are positively associated with the perceived influence of the project. A project that is perceived to have a greater influence among users in the future is more likely to attract early adopters than other projects.

H4: Perceived influence of a project has a significant and positive impact on the early adoption rate of OSS.

- *Information availability*. It refers to the amount of project-related information available to users. More information can not only better inform potential adopters but also indicate that the project is well planned and has a higher chance of sustaining its development in the future.

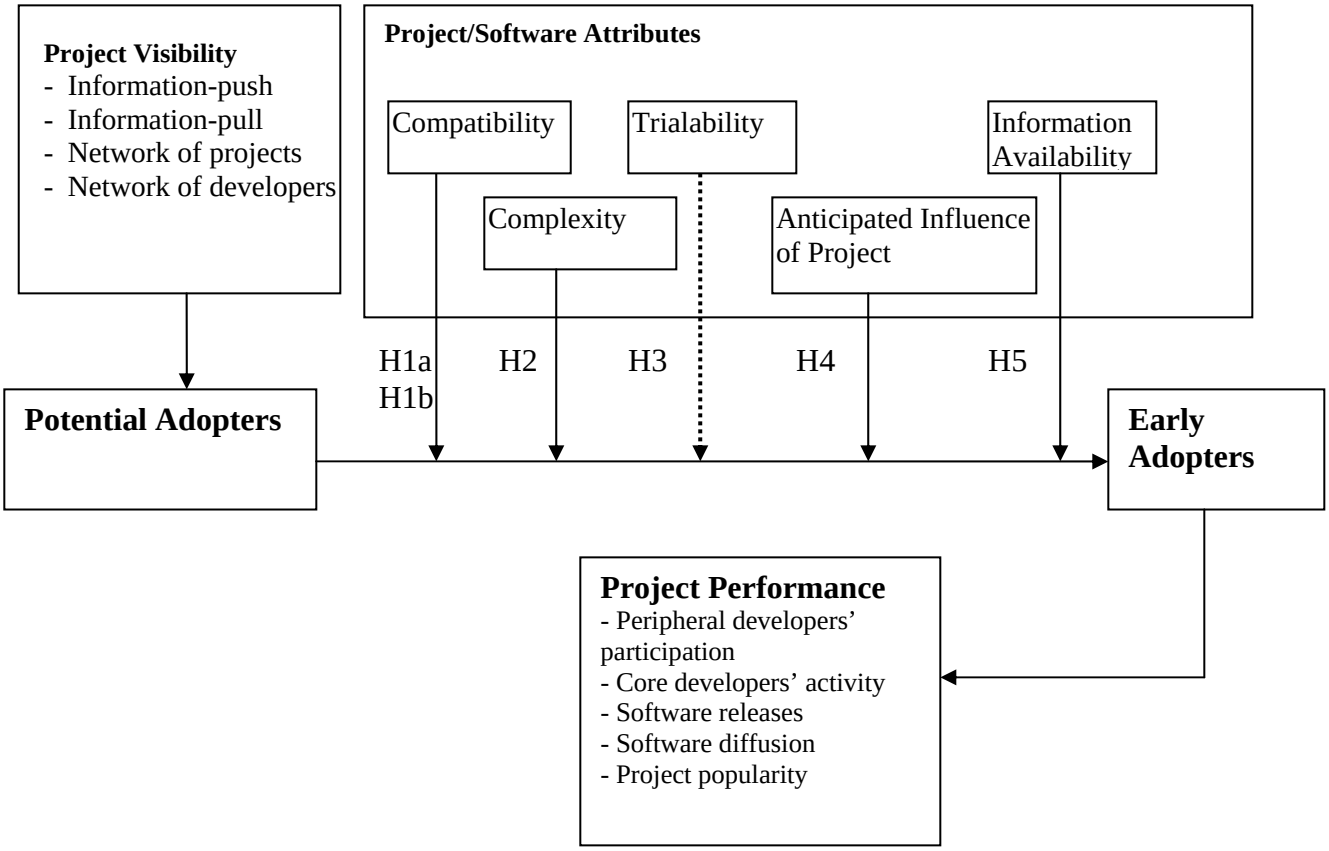
H5: Information availability of a project has a significant and positive impact on the early adoption rate of OSS.

2.4. Research Framework

Figure 3 presents our research framework that focuses on users' adoption of the first OSS releases. The process consists of two steps. First, a project's visibility obtained through information-push, information-pull, network of projects, and network of developers is positively associated with the number of potential adopters who are aware of the project. Second, the

conversion from project-aware potential users to actual adopters depends on certain characteristics of the project and the software along several dimensions, such as complexity, compatibility, anticipated influence of project, and information availability. An inherently less attractive project that is hard to search for is less likely to obtain many early adopters who are willing to download and try the software. In addition, recognizing that the OSS system is a self-organizing system where popular projects tend to attract more users and less popular projects tend to become even less popular, we expect that the early adoption behavior of users will have an impact on the project's subsequent performance in terms of peripheral developers' participation, core developers' activity level, software releases, diffusion of the software, and project popularity. Hence, we also include several metrics of project success in the model in order to explore the performance implication of having early adopters in OSSD.

Figure 3. Theoretical Framework



3. Research Methodology

3.1. Data

We collected data from the monthly dump of SourceForge.net's databases currently hosted at the University of Notre Dame (http://www.nd.edu/~oss/). As the largest repository of open source applications on the Internet, SourceForge.net currently provides free hosting to more than 100,000 projects and more than 1,100,000 subscribers. It also offers services and development tools such as web site hosting, mailing lists, and discussion forums to hosted projects. SourceForge.net has been one of the main data sources for OSS researchers due to the abundance of publicly accessible data (Howison and Crowston 2004).

Our sample consists of all public OSS projects registered on Sourceforge.net between October 30, 2004 and December 3, 2004 that had at least one software release by July 2006. We excluded 64 projects that released software code before their registration at SourceForge, a phenomena that may be caused by project take-over or project forking. The final sample has 933 projects. They are the oldest projects in the monthly SourceForge data dump that have complete recorded historical data. We choose them to ensure that they have sufficient time for code development. Because we are also interested in examining the subsequent project performance implications of early user adoption, these projects have activity and performance data over a relatively long period of time.

3.2. Variables

Dependent Variables (Analysis of Potential Adopters)

ProjectTraffic measures the amount of traffic to the pages related to the project. It approximates the number of potential users looking for more project-related information. It is computed as the natural log of the sum of *WebHit* and *PageView*, which are explained in details below.

SourceForge.net collects several traffic-related statistics that can be used as approximate measures of the number of potential users seeking project information.

- *WebHit* measures the amount of traffic to the project pages at SourceForge.net. It represents the natural log of the total number of files served from the SourceForge project web server (i.e. URL is http://sourceforge.net/projects/projectname) within

30 days of the project's first release. It includes a count of how many times the project summary page and project tool pages are accessed.

- *PageView* measures the amount of traffic to the project web pages. It represents the natural log of the number of hits to the pages of each project hosted in SourceForge's project web space (ie. URL is http://projectname.sourceforge.net) within 30 days of the project's first release. Each web page of a project usually displays a SourceForge logo on the top. *PageView* is calculated by counting the number of times the logo is loaded.

Because it is likely that a potential user may access multiple pages related to one project, both *WebHit* and *PageView* may overestimate the number of potential users. However, since the scope of this study is limited to user interest and adoption behavior in the first month after a project's first software release, during which the project may not have set up project tool pages either at SourceForge.net or at its own web space, *WebHit* and *PageView* can serve as reasonable measures of user interest.

Independent Variables (Analysis of Potential Adopters)

As described in Section 3, we identified four channels for obtaining innovation visibility: information-push, information-pull, network of projects, and network of developers. We explain the variables that represent each channel below.

- Information-push. A project can push information to users through mass communication channels. The most direct way is to use the news announcement tool provided by SourceForge.net. We include the following variables related to information-pushing:

  NumNews represents the degree of publicity a project creates before its software release by announcing project-related information to the community. It is computed as the number of news announcements that the project makes before its first release.

  AnnounceRelease indicates whether a project uses the news announcement feature to publicly announce its software releases to the entire SourceForge community.
- Information-pull. A project tends to have higher visibility if it makes itself easier for an individual to search for. For example, if projects hosted at SourceForge.net specify technical details about their software, users can use these details as filter criteria and

search for a particular project in the software map. In addition, users can search for a project based on some keywords in its general description.

TopicDefined, *AudienceDefined, PLanguageDefined*, and *OSDefined* indicate whether a project defines its topic, intended audience, programming language, and operating system in the software map of SourceForge.net.

DescLength represents how much information is contained in a project's description. It is computed as the natural log of the number of characters contained in a project's description.

- Network of projects. *PjtEmbeddedness* measures the extent to which a project is embedded in a network of projects. It is computed as the number of projects that share at least one common developer with the focal project during the month of the focal project's first software release.
- Network of developers. *DevEmbeddedness* captures the extent to which a project's core developers are embedded in a network of developers. It is calculated as the number of distinct developers who are collaborating with the focal project's core developers at the time of the project's first release. In order to identify the existing collaborative ties among developers, we collected data on all the other projects that the focal project's developers are working on during the month of the first release and these projects' core members. *NumDeveloper* is the number of core developers participating in a project during the month of its first release.

Control Variable (Analysis of Potential Adopters)

FirstReleaseTime measures the number of days a project takes have its first software release. It is the number of days between a project's registration and its first release.

Table 10 gives a summary of the variables used in our analyses of potential adopters.

Table 10. Summary of Variables (Analysis of Potential Adopters)

Variable Name	Description
Dependent Variables	
ProjectTraffic	The amount of traffic to the pages of the project. It is calculated as the natural log of the total number of files served from the SourceForge project web server plus the number of hits to the project pages hosted in the SourceForge's project web space.
Independent Variables	
NumNews	Number of news announcements that the project has made prior to its first release.
AnnounceRelease	Whether the project publicly announced its first software release.
DescLength	Natural log of the number of characters in the project description.
TopicDefined	Whether the project has defined its topic by the month of its first release.
AudienceDefined	Whether the project has defined its target audience by the month of its first release.
PLanguageDefined	Whether the project has defined its programming language by the month of its first release.
OSDefined	Whether the project has defined its operating system by the month of its first release.
PjtEmbeddedness	Number of projects that share at least one common developer with the focal project.
DevEmbeddedness	Number of external developers who are currently collaborating with the focal project's developers on other projects.
NumDevelopers	Number of core developers participating in the project during the month of the first release.
Control Variable	
FirstReleaseTime	Number of days between a project's registration and its first software release.

Dependent Variables (Analysis of Early Adopters)

We use the following dependent variable to study the actual adoption behavior of users:

- *AdoptionRatio* measures the conversion rate from potential users seeking information to actual users who downloaded the software. It is computed as the ratio of the number of downloads of the project's first software release at SourceForge.net to *ProjectTraffic*, the amount of traffic to the project pages, within 30 days of the project's first release.

Independent Variables (Analysis of Early Adopters)

In our theoretical framework, we identify the properties of an OSS project or its software that may influence users' early adoption behavior. Next we explain the variables representing these properties in detail.

- *Complexity*. We identify three measures that represent software complexity. *SoftwareSize* refers to the natural log of total number of bytes in the first software release.

NumFiles refers to the number of files and *NumPackages* is the number of packages in the software.

- *Compatibility*. We use *PlatformIndep*, which indicates whether the software is platform independent, to capture the technical compatibility. Legal compatibility is mainly related to the license adopted by the software. Lerner and Tirole (2005) distinguish the restrictiveness of these licenses based on two criteria: whether the license requires modified versions of the software to remain open, and whether the license requires modified versions of the software to be combined with the code under the same license. Licenses satisfying the first criterion are restrictive whereas those meeting the second criterion are highly restrictive. For example, the BSD license is unrestrictive, the GNU Lesser General Public License (LGPL) is restrictive, and the GNU General Public License (GPL) is most restrictive. Stewart et al. (2005) also use similar criteria when coding restrictiveness of an open source license. We follow the same coding scheme of license restrictiveness. *RestrictiveLicense* indicates whether the software license requires that modified versions of the software remain open and *HighlyRestrictiveLicense* indicates whether the license requires combining code under the same license.
- *Trialability*. Release notes can have an influence on the easiness of downloading, installing, and understanding the software. *NoteLength* is computed as the natural log of the number of characters in the release notes. Usually the longer a release note, the more information it contains and the easier it is to install and try the software.
- *Anticipated Influence of Project*. *NumDeveloper* represents the number of core developers participating in the project during the month of its first release. The more developers working on the project, the greater the anticipated impact of the project. *TopicPopularity* refers to whether the project's topic belongs to the most popular topics at SourceForge.net, such as Internet and communications. *PjtExperience* measures the number of other projects that the focal project's core developers are participating in.
- *Information Availability*. *DescLength*, the natural log of the number of characters in the project description, captures the amount of information provided in it. *AudienceDefined* and *TopicDefined* indicate whether a project has defined its intended audience and topic by the month of its first release.

Control Variable (Analysis of Early Adopters)

FirstReleaseTime measures the number of days a project takes to have its first release. It is the number of days between a project's registration and its first release.

AcceptDonation refers to whether the project has opted to accept donations from users.

Table 11 gives a summary of the variables used in our analyses of early adopters.

Table 11. Summary of Variables (Analysis of Early Adopters)

Variable Name	Description
Dependent Variables	
AdoptionRatio	The ratio of the number of downloads of the project's first software release at SourceForge.net to *ProjectTraffic*, the amount of traffic to the project pages at SourceForge.net, to within 30 days of the project's first release.
Independent Variables	
SoftwareSize	The natural log of total number of bytes in the first software release.
NumFiles	The number of files in the software.
NumPackages	The number of packages contained in the software.
PlatformIndep	Whether the software is platform independent.
RestrictiveLicense	Whether the software license requires that modified versions of the software remain open.
HighlyRestrictiveLicense	Whether the license requires combining code under the same license.
NoteLength	The natural log of the number of characters in the release notes.
NumDevelopers	Number of core developers participating in the project during the month of the first release.
TopicPopularity	Whether the project's topic belongs to the most popular topics at SourceForge.net, such as Internet and communications.
PjtExperience	Number of projects that the focal project's core developers are participating in.
DescLength	Natural log of the number of characters in the project description.
AudienceDefined	Whether the project has defined its target audience by the month of its first release.
TopicDefined	Whether the project has defined its topic by the month of its first release.
Control Variable	
FirstReleaseTime	Number of days between a project's registration and its first software release.
AcceptDonation	Whether the project has opted to accept donations from users

3.3. Analytical Procedures

Preliminary Regression Analyses

First we conducted linear regression analyses on 635 projects whose daily traffic information is available to examine the visibility factors that impact the potential adopters of a project. These projects released the first version of software on or before December 16, 2004. Traffic data were then aggregated for 30 days following the release date. However, this sub-

sample may be biased because they consist of only the projects that had relatively earlier releases than other projects. We plan to extend the analyses to the entire sample of 933 projects in the next step of the study.

Next, after performing an arcsine transformation on the dependent variable *AdoptionRatio*, we ran a linear regression model to examine the factors that attract potential adopters to download and adopt software. One of the assumptions for linear regression model is that data are normally distributed with no imposed limits on the dependent variable, which does not hold in our study because *AdoptionRatio* represents a percentage within the range 0% - 100%. An arcsine transformation in the form of *arcsin(SQRT(percentage variable))* is recommended especially when there are a large number of observations closer to 0% or 100%, indicating the normality assumption is severely violated. The mean AdoptionRatio in our sample is about 10%. The sample size for this step of the analysis is 627[7].

Alternative Analytical Strategies

In the preliminary analyses we treated the two-step process of users' information seeking and software adoption as two separate processes, which may not be able to reveal the connection between the two steps. In addition, project visibility is a latent variable that cannot be observed and measured directly. Therefore, next we examine the possible alternative analytical approaches that may enable us to investigate both information seeking and software adoption in a single analysis.

- Stochastic Frontier Analysis. Stochastic frontier model introduced by Aigner et al. (1977) and Meeusen and van den Broeck (1977) is a general linear model of productive output as a function of inputs and a composite error terms, which consists of an unobserved one-sided error component representing technical or cost inefficiency, and a two-sided error term representing random effect. We may model the early adoption process as a production process with potential adoption as an input and actual adoption as output. The technical inefficiency may be explained by the project attributes.
- Path Analysis. As an extension of the regression model, path analysis is used to provide estimates of the magnitude and significance of hypothesized causal

[7] 4 projects have more downloads than *ProjectTraffic*. In addition, 4 projects have file releases of 0 byte, possibly due to recording errors. Therefore, we dropped these 8 projects from the preliminary analyses.

connections between sets of variables, which are usually depicted in a path diagram. A regression is done for each endogenous variable dependent on other variables according to the path diagram. We can perform a path analysis by including project visibility variables, software complexity, compatibility, anticipated influence of project, and information availability as exogenous variables and potential adopters and actual adopters as endogenous variables.

4. Preliminary Results

4.1. Results of Analyzing Potential Adopters

Descriptive Statistics

Table 12 presents the descriptive statistics. Table 13 shows the pairwise correlations among the variables. The correlations between *TopicDefined*, *PLanguageDefined* and *OSDefined* are all above 0.7. After dropping *PLanguageDefined* and *OSDefined* from the predictor variables, we obtained additional collinearity diagnostics and found that the tolerances of all other predictor variables are above 0.7 and that the variance inflation factor values are all below 2.0, indicating that multicollinearity is not a major concern in the following analysis.

Table 12. Descriptive Statistics

	Descriptive Statistics			
Variable	**Mean**	**St. Dev**	**Min**	**Max**
ProjectTraffic	6.28	1.300	1.10	11.53
NumNews	1.89	3.935	0.00	42.00
AnnounceRelease	0.26	0.441	0.00	1.00
DescLength	5.04	0.547	3.18	5.55
TopicDefined	0.69	0.461	0.00	1.00
AudienceDefined	0.62	0.485	0.00	1.00
PLanguageDefined	0.72	0.450	0.00	1.00
OSDefined	0.61	0.487	0.00	1.00
PjtEmbeddedness	1.21	2.271	0.00	19.00
DevEmbeddedness	7.07	28.427	0.00	289.00
NumDevelopers	1.38	1.120	1.00	19.00
FirstReleaseTime	7.10	7.481	0.03	49.00

Table 13. Correlations

Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
(1) ProjectTraffic											
(2) NumNews	0.357***										
(3) AnnounceRelease	0.191***	0.356***									
(4) DescLength	0.070*	0.068*	0.043								
(5) TopicDefined	0.228***	0.143***	0.127***	0.042							
(6) AudienceDefined	0.189***	0.122***	0.068*	0.086**	0.501***						
(7) PLanguageDefined	0.205***	0.133***	0.096**	0.075*	0.857***	0.507***					
(8) OSDefined	0.224***	0.162***	0.094**	0.022	0.703***	0.459***	0.715***				
(9) PjtEmbeddedness	0.056	-0.048	-0.080**	-0.005	-0.020	0.053	-0.005	-0.005			
(10) DevEmbeddedness	-0.007	0.044	0.013	-0.004	-0.012	0.031	-0.021	0.005	0.305***		
(11) NumDevelopers	0.200***	0.036	-0.042	0.004	0.003	-0.003	0.003	0.030	0.296***	0.198***	
(12) FirstReleaseTime	-0.143***	-0.055	-0.049	-0.044	-0.083**	-0.052	-0.083**	-0.093**	0.151***	0.112***	0.073*
Significance levels: *** 0.01, ** 0.05, * 0.1											

Linear Regression Results

We estimated the parameters for the following linear regression models to examine the factors that influence the number of users seeking project information or potential adopters:

$$ProjectTraffic = \alpha + \beta_1 NumNews + \beta_2 AnnounceRelease + \beta_3 DescLength + \beta_4 TopicDefined + \beta_5 AudienceDefined + \beta_6 PjtEmbeddedness + \beta_7 DevEmbeddedness + \beta_8 NumDevelopers + \beta_9 FirstReleaseTime + \varepsilon$$

A positive and significant parameter estimate would suggest that the corresponding factor is positively related to a project's traffic. The results of the regression analyses are presented in Table 14. The model fits with the data moderately well (*F statistic* = 20.11, $p < 0.01$; R^2 = 0.225). The parameter estimate for *NumNews* is positive and significant ($\beta_1 = 0.097$, $p < 0.01$), suggesting that *ProjectTraffic* increases by a factor of $e^{0.097}$ or 10.2% for each additional news announcement prior to the release. *AnnounceRelease* is a positive and significant predictor (β_2 = 0.210, $p < 0.1$), suggesting a project that publicly announces its first release on average attracts 23.4% more traffic than a project that doesn't. *TopicDefined* has a significantly positive parameter estimate ($\beta_4 = 0.359$, $p < 0.01$), indicating that on average a project with topic defined receives 43.2% more traffic than it would with topic undefined. *AudienceDefined* is another significant predictor of the dependent variable ($\beta_5 = 0.201$, $p < 0.1$), indicating that defining its intended audience can help a project attract 22.3% more traffic. *NumDevelopers* has a positive and significant parameter estimate in the model ($\beta_8 = 0.230$, $p < 0.01$), suggesting that on average having one addition core developer increases a project's user visits by 25.9%. In addition, *FirstReleaseTime* is a significant predictor with a negative parameter estimate (β_9 = -0.021, $p < 0.01$), implying that an earlier release is associated with more user visits. *DevEmbeddedness* has a significant but negative parameter estimate (β_7 = -0.003, $p < 0.1$), suggesting that the more embedded a project's core members in the developers' network, the less amount of traffic the project is likely to receive. However, the impact is quite small with 0.3% decrease in project traffic for each additional external collaborator that the core members have.

In summary, the results suggest projects that make more news announcements before releasing software, publicly announces their first releases, have defined their topics by the time of their first release, have more core developers, and release early tend to attract more users to seek project-related information. At a higher level, pushing information to users, making

projects more searchable to users, and having a larger development team seem to help the project attract more potential adopters seeking detailed information.

Table 14. Regression Results (Dependent Variable - ProjectTraffic)

Variable	Parameter Estimate
Constant	5.138***
NumNews	0.097***
AnnounceRelease	0.210**
DescLength	0.068
TopicDefined	0.359***
AudienceDefined	0.201**
PjtEmbeddedness	0.031
DevEmbeddedness	-0.003
NumDevelopers	0.230***
FirstReleaseTime	-0.021***
Model Statistics	
Sample Size (*N*)	635
F Statistic	20.110***
R^2	0.225
Adjusted R^2	0.213
Significance levels: *** 0.01, ** 0.05, * 0.1	

4.2. Results of Analyzing Early Adopters

Descriptive Statistics

Table 15 presents the descriptive statistics. Table 16 shows the pairwise correlations among the variables. The highest correlation is between *TopicDefined*, and *AudienceDefined* ($\rho = 0.495$, $p < 0.001$).

Table 15. Descriptive Statistics

	Descriptive Statistics			
Variable	**Mean**	**St. Dev**	**Min**	**Max**
AdoptionRatio	0.15	0.137	0.003	0.962
SoftwareSize	12.11	2.279	5.971	17.974
NumFiles	1.54	1.195	1.000	14.000
NumPackages	1.84	2.730	1.000	58.000
PlatformIndep	0.50	0.500	0.000	1.000
RestrictiveLicense	0.12	0.325	0.000	1.000
HighlyRestrictiveLicense	0.72	0.449	0.000	1.000
NoteLength	2.91	2.257	0.000	6.248
NumDevelopers	1.38	1.122	1.000	19.000
TopicPopularity	0.06	0.241	0.000	1.000
PjtExperience	1.21	2.322	0.000	19.000
DescLength	5.04	0.546	3.178	5.545
AudienceDefined	0.63	0.483	0.000	1.000
TopicDefined	0.71	0.456	0.000	1.000
FirstReleaseTime	6.88	7.147	0.034	38.386
AcceptDonation	0.20	0.402	0.000	1.000

Table 16. Correlations

Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
(1) AdoptionRatio															
(2) SoftwareSize	0.107***														
(3) NumFiles	0.148***	0.277***													
(4) NumPackages	0.031	0.091**	0.049												
(5) PlatformIndep	-0.071*	-0.095**	-0.004	-0.095**											
(6) RestrictiveLicense	-0.043	0.052	0.001	0.046	0.069*										
(7) HighlyRestrictiveLicense	-0.025	-0.027	0.001	0.002	-0.092**	-0.592***									
(8) NoteLength	-0.081**	0.039	0.123***	-0.086**	0.023	0.084**	-0.089**								
(9) NumDevelopers	-0.023	0.107***	0.140***	0.046	0.056	0.044	-0.006	0.014							
(10) TopicPopularity	0.042	-0.117***	-0.034	0.010	0.019	-0.031	-0.010	0.109***	0.098**						
(11) PjtExperience	0.035	0.026	0.065	-0.039	0.109***	0.091**	-0.090**	0.027	0.299***	0.001					
(12) DescLength	0.067*	0.024	-0.006	0.053	0.028	0.006	0.036	0.051	0.039	0.021	-0.002				
(13) AudienceDefined	-0.006	-0.004	0.017	-0.003	0.036	0.027	-0.091**	0.087**	0.016	0.110***	0.054	0.077*			
(14) TopicDefined	0.064	0.009	0.048	0.026	0.085**	-0.032	0.006	0.025	0.017	0.059	-0.032	0.035	0.495***		
(15) FirstReleaseTime	-0.019	0.056	0.044	-0.020	0.109***	0.028	0.009	-0.005	0.061	-0.031	0.165***	-0.035	-0.036	-0.047	
(16) AcceptDonation	-0.182***	-0.133***	-0.055	-0.049	-0.044	-0.083**	-0.057	0.009	-0.019	0.044	-0.042	0.077*	0.082**	0.160***	-0.072*

Significance levels: *** 0.01, ** 0.05, * 0.1

Regression Results

We estimated the parameters for the following regression model to examine the factors that influence the number of users seeking project information or potential adopters:

$$\arcsin(\text{SQRT}(AdoptionRatio)) = \alpha + \beta_1 SoftwareSize + \beta_2 NumFiles + \beta_3 NumPackages + \beta_4 PlatformIndep + \beta_5 RestrictiveLicense + \beta_6 HighlyRestrictiveLicense + \beta_7 NoteLength + \beta_8 NumDevelopers + \beta_9 TopicPopularity + \beta_{10} PjtExperience + \beta_{11} DescLength + \beta_{12} AudienceDefined + \beta_{13} TopicDefined + \beta_{14} FirstReleaseTime + \beta_{15} AcceptDonation + \varepsilon$$

The results are presented in Table 17. The model exhibits a good fit with the data (F = 2.92, $p < 0.01$). *PlatformIndep* has a negative and significant parameter estimate, not supporting our hypothesis *H1a*. It seems that platform-independent software tends to have a lower adoption ratio than platform-dependent software. *RestrictiveLicense* is a significant predictor that has a negative impact on the adoption ratio, suggesting software with restrictive licenses seems to have lower adoption ratio. Thus, hypothesis *H1b* is supported. The parameter estimates for *SoftwareSize* and *NumFiles* are positive and significant, suggesting that complexity of software is positively associated with the adoption ratio. Hence, hypothesis *H2* is supported. The parameter estimates for *NoteLength* is not significant, showing support for our hypothesis *H3*. With regard to the variables representing the anticipated influence of projects, *PjtExperience* and *TopicPopularity* have a positive estimate, partially support hypothesis *H4*. *TopicDefined* is a significant predictor with positive influence on *AdoptionRatio*, showing partial support for hypothesis *H5*.

Overall the results indicate that early OSS adopters tend to choose the software that is more complex in terms of size and scale, is platform-specific, has less restrictive license, has more experienced core developers, and identifies the project topic. *Complexity*, *Compatability*, *Anticipated Influence of Project*, and *Information Availability* all seem to have some influence on attracting potential users to download and adopt OSS software in the early phase of a project.

Table 17. Regression Results (Dependent Variable: AdoptionRatio)

Variable	Parameter Estimate
Intercept	0.6390**
SoftwareSize	0.0327**
NumFiles	0.0720***
NumPackages	-0.0057
PlatformIndep	-0.1108**
RestrictiveLicense	-0.0030**
HighlyRestrictiveLicense	-0.0004
NoteLength	-0.0154
NumDevelopers	-0.0396
TopicPopularity	0.2679**
PjtExperience	0.0258**
DescLength	0.0628
AudienceDefined	-0.0493
TopicDefined	0.1493**
FirstReleaseTime	-0.0091**
AcceptDonation	-0.0333
Model Statistics	
Sample Size (*N*)	627
F Statistic	2.92***
R square	0.067
Significance levels: *** 0.01, ** 0.05, * 0.1	

CHAPTER 4. CONCLUSION

1. Current Status

Essay one, which has been completed, empirically examines the driving forces of the self-organizing process of OSSD project team formation. Our results indicate that both the existence and the process-related strength of collaborative ties between the developer and the project originator positively impact the likelihood of the developer joining the project. Although initiators' experience in terms of duration and project is not a significant predictor, their network embeddedness does have a positive and important influence on the project's probability of attracting more developers. We also find that developers' experience has a negative impact on their joining behavior.

Essay 2 is still in progress. We have developed a theoretical model of OSS users' early adoption behavior. After gathering data from SourceForge.net, we have performed some preliminary regression analyses, which show that the number of potential users as estimated by the amount of traffic to the project is positively influenced by the project's visibility created mainly through information-push and searchability. In addition, some properties of the project such as *complexity* and *anticipated influence of project* positively impact the percentage of potential users who actually choose to download the software.

However, as mentioned earlier, we are yet to apply some additional analytical strategies such as frontier estimation in order to combine the analysis of potential users and that of actual users. Furthermore, we intend to extend the analyses to our entire sample of 933 projects. In addition, we plan to investigate whether users' early adoption has any direct or indirect influences on an OSSD project's future performance.

2. Expected Contributions

Overall, this research contributes to our understanding of the behavior of OSS developers and users by studying the self-organizing process of team formation and user adoption of OSS. In addition, we focus on these important yet under-investigated aspects in OSS projects' early phase in order to reveal a clearer picture of when, how, and possibly why projects start to exhibit huge heterogeneity.

The findings from our first study not only shed light on evolution of open source software projects but also provide a possible explanation for the formation of scale-free networks prevalent in OSS networks. Furthermore, this research serves as a starting point for more in-depth analyses of how a community of developers and users gather around an OSS project over time.

Our second study borrows from and adapts the traditional, centralized innovation theory to examine the adoption of decentralized open source software innovations. Our results not only have theoretical implications in the innovation domain but also provide some guidelines for OSS project managers to attract and retain early adopters.

3. Limitations and Future Research

One limitation of the study is that we used SourceForge.net as the only data source for this dissertation. Complementing our existing data with data collected from other sources as such FreshMeat.net may make our results more generalizable. Second, combining the current results with results from additional cross sectional studies of projects at more mature development stages may reveal interesting patterns as to how developers and users behave differently over time. Third, how early adopters later become peripheral and even core members of OSS projects is another interesting direction to pursue. In addition, we intend to incorporate the influence of negative network externalities caused by competition among OSS projects into the second essay. In summary, the overall research model presented in Chapter 1 only serves as a basis for our future exploration of the dynamics in open source software development.

References:

Aigner, D., K. Lovell, and P. Schmidt. 1977. Formulation and estimation of stochastic frontier production function models. *Journal of Econometrics* **6** 21-37.

Beal, D. J., R. R. Cohen, M. J. Burke and C. L. McLendon. 2003. Cohesion and performance in groups: A meta-analytic clarification of construct relations. *Journal of Applied Psychology* **88**(6) 989-1004.

Benford, R. D. 1993. You could be the hundreth monkey: Collective action frames and vocabularies of motives within the nuclear disarmament movement. *Sociological Quarterly* **34** 195-216.

Benkler, Y. 2002. Coase's penguin, or, Linux and the nature of the firm. *Yale Law Journal* **112**(3) 369-446.

Boehm, B. W. 1987. Improving software productivity. *IEEE Computer* **20**(9) 43-57.

Borgatti, S. P. and P. C. Foster. 2003. The network paradigm in organizational research: A review and typology. *Journal of Management* **29**(6) 991-1013.

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

Colazo, J., Y. Fang, and D. J. Neufeld. 2005. The effect of copyleft license on open source software development success. *Proceedings of the 11th Americas Conference on Information Systems*, Omaha, NE, U.S.A.

Cook, K. S. and J. M. Whitmeyer. 1992. Two approaches to social structure: Exchange theory and network analysis. *Annual Review of Sociology* **18** 109-127.

Crowston, K., H. Annabi and J. Howison. 2003. Defining open source software project success. *Proceedings of Twenty-Fourth International Conference on Information Systems*, Seattle WA, 327-340.

Crowston, K. and J. Howison. 2005. The social structure of free and open source software development. *First Monday*, **10**(2).

Crowston, K. and B. Scozzi. 2002. Open source software projects as virtual organizations: competency rallying for software development. *IEE Proceedings — Software Engineering* **149**(1) 3–17.

Crowston, K., K. Wei, Q. Li, U. Eseryel and J. Howison. 2005. Coordination of free/libre open source software development. *Proceedings of Twenty-Sixth International Conference on Information Systems,* Las Vegas, NV.

Curtis, B., H. Krasner and N. Iscoe. 1988. A field study of the software design process for large systems. *Communications of the ACM* **31**(11) 1268-1287.

Faraj, S. and L. Sproull. 2000a. Coordinating expertise in software development teams. *Management Science* **46**(12) 1554-1568.

Faraj, S. and L. S. Sproull. 2000b. Coordinating expertise in software development teams. *Management Science* **46**(12) 1554-1568.

Franke, N., and S. K. Shah. 2003. How communities support innovative activities: An exploration of assistance and sharing among end-users. *Research Policy* **32** 157-178.

Granovetter, M. 1973. The strength of weak ties. *American Journal of Sociology* **78**(6) 1360-1380.

Grewal, R., G. Lilien and G. Mallapragada. 2006. Location, location, location: How network embeddedness affects project success in open source systems. *Management Science* **52**(7) 1043–1056.

Gruenfeld, D. H., E. A. Mannix, K. Y. Williams and M. A. Neale. 1996. Group composition and decision making : How member familiarity and information distribution affect process and performance. *Organizational Behavior and Human Decision Processes* **67**(1) 1-15.

Gulati, R. 1995. Social structure and alliance formation pattern: A longitudinal analysis. *Administrative Science Quarterly* **40**(4) 619-652

Hars, A. and S. Qu. 2002. Working for free? Motivations for participating in open-source projects. *International Journal of Electronic Commerce* **6**(3) 25-39.

Hassinger, E. 1959. Stages in the adoption process. *Rural Sociology* **24** 52-53.

Hertel, G., S. Niedner and S. Herrmann. 2003. Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy* **32**(7) 1159-1177.

Hinds, P. J., K. M. Carley, D. Krackhardt and D. Wholey. 2000. Choosing work group members: Balancing similarity, competence, and familiarity. *Organizational Behavior and Human Decision Processes* **81**(2) 226-251.

Howison, J. and K. Crowston. 2004. The perils and pitfalls of mining sourceforge. *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), Mining Software Repositories Workshop*, Edinburgh, Scotland.

Howison, J., K. Inoue and K. Crowston. 2006. Social dynamics of free and open source team communications. *Proceedings of the IFIP 2nd International Conference on Open Source Software*, Lake Como, Italy.

Huang, S. and G. DeSanctis. 2005. Mobilizing informational social capital in cyber space: Online social network structural properties and knowledge sharing. *Proceedings of the 26th International Conference on Information Systems*, Las Vegas, NV, 207-219.

Karau, S. J. and K. D. Williams. 2000. Understanding individual motivation in groups: The collective effort model. *Groups at work: Theory and research*. M. E. Turner, ed. Lawrence Erlbaum Associates, Mahwah, NJ, 113-141.

King, G. and L. Zeng. 2001. Logistic regression in rare events data. *Political Analysis* **9**(2) 137-163.

Koch, S. and G. Schneider. 2002. Effort, cooperation and coordination in an F/OSS software project: GNOME. *Information Systems Journal* **12**(1) 27–42.

Krishnamurthy, S. 2002. Cave or community? An empirical examination of 100 mature open source projects. *First Monday* **7(**6).

Kuk, G. 2006. Strategic interaction and knowledge sharing in the kde developer mailing list. *Management Science* **52**(7) 1031-1042.

Lakhani, K. R. and R. Wolf. 2005. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Perspectives on free and open source software*. J. Feller and B. Fitzgerald and S. Hissam and K. R. Lakhani, eds. MIT Press, Cambridge, MA, 3-22.

Lerner, J. and J. Tirole. 2002. Some simple economics of open source. *Journal of Industrial Economics* **50**(2) 197-234.

Lerner, J., and J. Tirole. 2005. The scope of open source licensing. *Journal of Law Economics & Organization* **21**(1) 20-56.

Levine, J. M. and R. L. Moreland. 1990. Progress in small group research. *Annual Review of Psychology* **41** 585-634.

Levine, J. and R. Moreland. 1991. Culture and socialization in work groups. In L. Resnick, J. Levine, & S. Teasley (Eds.), *Perspectives on Socially Shared Cognition*. Washington, DC: American Psychological Association.

Lopez-Fernandez, L., G. Robles and J. M. Gonzalez-Barahona. 2004. Applying social network analysis to the information in cvs repositories. *Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), Mining Software Repositories Workshop*, Edinburgh, Scotland.

Maas, W. 2004. Inside an open source software community: Empirical analysis on individual and group level. *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburgh, Scotland, 64–70.

Madey, G., V. Freeh and R. Tynan. 2002. The open source software development phenomenon: An analysis based on social network theory. *Proceedings of the 8th Americas Conference on Information Systems*, Dallas, Texas, USA, 1806-1813.

Manski, C. F. and S. R. Lerman. 1977. The estimation of choice probabilities from choice based samples. *Econometrica* **45**(8) 1977-1988.

McClelland, D., J. Atkinson, R. Clark and A. Lowell. 1953. *The achievement motive*. Appleton-Century-Crofts, New York, NY.

McPhail, C. and D. Miller. 1973. The assembling process: A theoretical and empirical examination. *American Sociological Review* **38** 721-735.

Meeusen, W., and J. van den Broeck. 1977. Efficiency estimation from Cobb-Douglas production functions with composed error. *International Economic Review* **18** 435-444.

Mockus A., R. Fielding and J. Herbsleb. 2000. A case study of F/OSS software development: the Apache server. *Proceedings. of 2000 International Conference on Software Engineering (ICSE 2000)*. Limerick, Ireland.

Moon, J. Y. and L. S. Sproull. 2002. Essence of distributed work: The case of the Linux kernel. *Distributed work*. P. J. Hinds and S. B. Kiesler, eds. MIT Press, Boston, MA, 381-404.

Moreland, R. 1987. The formation of small groups. In C. Hendrick (Ed.), *Review of Personality and Social Psychology*, 8, Newbury Park, CA: Sage.

Moreland, R. L. 1999. Transactive memory: Learning who knows what in work groups and organizations. *Shared cognition in organizations: The management of knowledge*. L. I. Thompson and J. M. Levine and D. M. Messick, eds. Lawrence Erlbaum Associates, Mahwah, NJ, 3-31.

Moreland, R. and J. Levine. 1992. The composition of small groups. In Advances in Group Processes, 9, 237-280. New York: JAI Press.

Moreland, R. and J. Levine. 1996. Creating the ideal group: Composition effects at work. In E. Witte & J. Davis (Eds.), Understanding Group Behavior: Small Group Process and Interpersonal Relations, 2, 11-35. New Jersey: Erlbaum.

Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida and Y. Ye. 2002. Evolution patterns of open-source software systems and communities. *Proceedings of 2002 International Workshop on Principles of Software Evolution (IWPSE 2002)*, Orlando, FL.

O'Reilly, T. 1999. Lessons from open-source software development. *Communications of the ACM* **42**(4) 33-37.

Owens, D. A., E. A. Mannix and M. A. Neale. 1998. Strategic formation of groups: Issues in task performance and team member selection. *Research on managing groups and teams: Composition*. D. H. Gruenfeld, ed., (Vol. 1). JAI Press, Stamford, CT, 149-165.

Powell, W. W., K. W. Koput and L. Smith-Doerr. 1996. Interorganizational collaboration and the locus of innovation: Networks of learning in biotechnology. *Administrative Science Quarterly* **41**(1) 116-145.

Raymond, E. S. 2001. *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary*. O'Reilly and Associates, Sebastapol, CA.

Robles, G., J. J. Merelo and J. M. Gonzalez-Barahona. 2005. Self-organized development in libre software projects: a model based on the stigmergy concept. *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005)*. St. Louis, MO, USA.

Roberts, J. A., I.-H. Hann and S. A. Slaughter. 2006. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. *Management Science* **52**(7) 984-999.

Rogers, E. M. 1983. *Diffusion of Innovations*. The free press. New York, NY.

Ruef, M., H. E. Aldrich and N. M. Carter. 2003. The structure of founding teams: Homophily, strong ties, and isolation among u.S. Entrepreneurs. *American Sociological Review* **68**(2) 195-222.

Schachter, S. 1959. *The psychology of affiliation*. Stanford University Press, Stanford, CA.

Shah, S. K. 2006. Motivation, governance and the viability of hybrid forms in open source software development. *Management Science* **52**(7) 1000-1014.

Singh, J. 2005. Collaborative networks as determinants of knowledge diffusion patterns. *Management Science* **51**(5) 756-770.

Snow, D. and R. Benford. 1992. Master-frames and cycles of protest. A, D, Morris, C, McClurg, eds. *Frontiers in Social Movement Theory.* Yale University Press, New Haven, CT, 133-154.

Stewart, D. 2005. Social status in an open-source community. *American Sociological Review* **70**(5) 823-842.

Stewart, K. and T. A. Ammeter. 2002. An exploratory study of factors affecting the popularity and vitality of open source projects. *Proceedings of the 23rd International Conference on Information Systems*, Barcelona, Spain.

Stewart, K., T. A. Ammeter and L. Maruping. 2005. A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects. *Proceedings of the 38th Hawaii International Conference on System Sciences*, Hawaii, USA.

Stewart, K. J., A. P. Ammeter and L. M. Maruping. 2006. Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. *Information Systems Research* **17**(2) 126-144.

Stewart, K. and S. Gosain. 2006a. The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly* **30**(2) 291-314.

Stewart, K. and S. Gosain. 2006b. The moderating role of development stage in affecting free/open source software project performance. *Software Process: Improvement and Practice* **11**(2) 177-191.

Thye, S. R. 2000. A status value theory of power in exchange relations. *American Sociological Review* **65**(3) 407-432.

Tsai, W. 2001. Knowledge transfer in intraorganizational networks: Effects of network position and absorptive capacity on business unit innovation and performance. *Academy of Management Journal* **44**(5) 996-1004.

van Wendel de Joode, R. 2004. Conflicts in open source communities. *Electronic Markets* **14**(2) 104–113.

von Hippel, 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 Hippel, E. and G. von Krogh. 2006. The promise of research on open source software. *Management Science* **52**(7) 975–983.

von Krogh, G., S. Spaeth and K. R. Lakhani. 2003. Community, joining, and specialization in open source software innovation: A case study. *Research Policy* **32** 1217-1241.

Walker, J., S. Wasserman and B. Wellman. 1994. Statistical models for social support networks. *Advances in social network analysis*. S. Wasserman and J. Galaskiewicz, eds. Sage, Thousand Oaks, CA., 53-78.

Wasserman, S. and J. Galaskiewicz. 1994. *Advances in social network analysis*. Sage, Thousand Oaks, CA.

Wellman, B. and S. D. Berkowitz. 1998. *Social structures: A network approach*. Cambridge University Press, Cambridge, UK.

Xu, J., Y. Gao, S. Christley and G. madey. 2005. A topological analysis of the open source software development community. *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS '05)*, Hawaii, HI.

Zander, A. and A. Havelin. 1960. Social comparison and interpersonal attraction. *Human Relations* **13**(1) 21-32.

Zhao, L., and F. P. Deek. 2004. User collaboration in open source software development. *Electronic Markets* **14**(2) 89 – 103.