A NETWORK PERSPECTIVE ON OPEN SOURCE SOFTWARE DEVELOPMENT: TEAM FORMATION AND COMMUNITY PARTICIPATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Chen Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2008

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Zhang, Chen. Ph.D., Purdue University, December, 2008. A Network Perspective on Open Source Software Development: Team Formation and Community Participation. Major Professors: Prabuddha De and Jungpil Hahn.

In the past few years there has been a surge in self-organizing voluntary teams and online communities collaborating online to produce goods and services. Open source software development has been frequently identified as an example of new way of organizing production. However, despite the increasing prevalence of open source software development, much is yet to be understood about the dynamics of self-organizing production communities. In this dissertation, we investigate the self-assembly of open source project teams and members' continued participation in open source user communities. The dissertation consists of two essays.

The first essay entitled "Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties" examines how open source project teams are formed and how individuals self-select into open source project teams. This essay focuses on how the collaborative network of developers influences their choice of newly-initiated projects to participate in. We empirically test the impact of prior collaborative ties with and perceived status of project members on the self-assembly of open source project teams. We find that a developer is more likely to join a project when he has developed strong collaborative ties with the project initiator. He is more likely to join a project when other developers with higher perceived status in the collaborative network have joined the project. In addition, projects that have attracted more developers into their teams are more likely to attract additional developers.

In order for open source projects to sustain their development they need to be able to attract and more importantly retain participation of volunteer users and developers. Despite the studies that have examined the dynamics in information-sharing online communities, little research has been conducted to investigate participation continuance in online innovation communities such as open source project communities. In the second essay, we attempt to answer the following research questions: How do communication activities with other members in an innovation community influence an individual's continued participation? How does an individual's involvement in the innovation process impact his continued participation? We find that interactions with other community members do influence an individual's return to the community. However, individual differences do exist in members' continuance behavior; their level of involvement in the OSSD process has a direct influence on their continued participation in the project community.

Overall, the findings suggest that the network that an individual is embedded in influences his choices and behavior in the context of open source software development. The open source collaboration network impacts developers' choices of new projects to join during the self-assembly process of project teams. In the open source user community interactions between an individual and other community members directly affect his continued participation in the community.

CHAPTER 1. INTRODUCTION

1.1. Background

Enabled by the unprecedented developments and increasing accessibility of information and communication technologies, a new model of production "based on community, collaboration, and self-organization" (Tapscott and Williams 2007, p. 1) has gained popularity in the past decade. Individuals are becoming increasingly connected to one another in self-organized online communities and collaborating to produce goods and services without relying on either market signals or managerial controls. A frequently cited example of this new model of production is open source software development (OSSD). Unlike the centralized cathedral-style mode of software development by commercial software companies where a small number of skilled developers are involved, OSSD is characterized as a bazaar-style software production mode involving a large number of individuals with "differing agendas and approaches" (Raymond 2001, p. 3), in which many volunteer developers usually from dispersed geographic locations and different industries or organizations collaborate online to develop software in self-forming communities (von Krogh and von Hippel 2006).

Research on OSS has found that OSSD typically involves the participation of an aggregation of contributors performing different roles (e.g., Koch and Schnider 2002, Mockus et al. 2002). For example, Nakakoji et al. (2002) classify the eight roles that an OSSD participant can take - passive user, reader, bug reporter, bug fixer, peripheral developer, active developer, core member, and project leader. Crowston and Howison (2006) propose that an OSSD community consists of project leaders, core developers, codevelopers, and active users. Because the success of an OSS project to a great extent

relies on the active involvement of its development team and user community, this dissertation focuses on these two groups of participants in the project.

After a project is initiated by an individual or a group of individuals, volunteer developers interested in the project may sign up to become members of the project team. The team mainly consists of project leaders and core developers who are responsible for designing, coding, and maintaining code as well as overseeing the evolution of the project. Many of the team members not only have commit privileges on the code repository but also serve as gate keepers who evaluate the contributions from others outside the team (von Krogh et al. 2003).

The project team acts as the cornerstone of the project and this core team of developers control the development of the majority of the source code (Koch and Schnider 2002, Mockus et al. 2002). As a distinguishing feature of OSSD, early and frequent releases allow the team to receive rapid and frequent feedback about the quality of the code, which in turn encourages the team to spend continued effort to improve the software and push the project forward (Hars and Ou 2002). Hence, the development activity performed by the team to a large extent determines the progress of the project. A project with an inactive team is unlikely to continue attracting users and developers. In addition, when the team size is small, the project may be too dependent on the effort of a few individuals; thus, it is more likely to become abandoned when some team members are no longer willing or able to contribute time and effort to the project. Furthermore, similar to that of software development teams in organizational settings (Faraj and Sproull 2000), member composition of the OSS project team can influence software development performance through its impact on administrative and expertise coordination effectiveness. The programming skills and domain experiences of the team members can also impact the project performance.

The user community consists of individuals who voluntarily participate on a less regular basis and who do not belong to the project team. It may include software developers

contributing bug fixes or implementing new features as well as users who test new releases, report software bugs, suggest new features, and submit opinions. The user community also includes users seeking software support and participating in project-related or software-related discussions. Rather than as a network of interacting equally-important peers, the user community can be characterized as an aggregation of contributors performing different activities.

The user community is an essential element in OSSD. First and most importantly, according to "Linus's Law" – "given enough eyeballs, all bugs are shallow" (Raymond 2001, p. 30), one of the key advantages of OSSD mode compared to proprietary software development is its source code accessibility that allows a large number of individuals to download, test, review, and improve the software, which then enables them to become involved in identifying errors, fixing errors, proposing new features or functionalities, and providing other feedback as the software evolves (Raymond 2001, von Hippel 2002, von Hippel and von Krogh 2003). Secondly, in the user community some users provide unpaid assistance to other users. For example, some users voluntarily provide field support, "provision of assistance to users having difficulties with a product" (Lakhani and von Hippel 2003, p. 923), to others for free. The Linux user groups studied by Bagozzi and Dholakia (2006, p. 1101) "conducts the business functions performed by the marketing, customer support, and business development groups in commercial proprietary software organizations". User-to-user assistance not only helps new users, protecting project team from possible burnout as a result of responding to all user support requests (Crowston and Howison 2006), but also helps to attract new users into the community (Bagozzi and Dholakia 2006). Thirdly, one of the reasons for enjoyment-driven participants in OSSD to contribute code is to obtain feedback from the user community confirming the usefulness of their activities to others (Shah 2006). A larger user community is more likely to generate challenging and interesting puzzles for these participants to solve, thus bring more satisfaction and enjoyment to them. Therefore, the ability of an OSS project to attract and maintain a large user community is one of the many factors that contribute to its success.

1.2. Research Questions

In the past decade OSS has captured the attention of many researchers from several disciplines such as information systems, organization science, economics, etc. (e.g., Lerner and Tirole 2002, Roberts et al. 2006, von Hippel and von Krogh 2003). Overall, these works have greatly contributed to our understanding of the OSS phenomenon and helped researchers to extend the lessons learned to other areas of peer production and innovation (von Krogh and von Hippel 2006). However, many important questions related to the OSSD project team and the user community are yet to be answered and this dissertation focuses on two of them.

First, what factors influence individual decision as to which project team to join or which user community to continue participating in? In OSSD, participants decide what they want to work on and when; they voluntarily form project teams and user communities to engage in large-scale collaborations. Prior research has examined why developers contribute to open source movement without monetary incentive or managerial controls by examining their general attitude toward OSSD and motivations to participate in OSSD (e.g., Hars and Ou 2002, Lerner and Tirole 2002, Roberts et al. 2006). However, not all OSS projects are equally attractive to developers and some may receive more contributions from developers than others. General motivations of developers to participate in OSSD are insufficient to explain the variances among projects in attracting developers. Furthermore, the exit barriers for OSSD participants are low; they can leave their teams or communities and stop participating at any time without incurring much cost. Much work is needed to understand what promotes or hinders the continued participation in project teams or user communities. Practically, understanding the factors influencing people's decisions to join a self-forming team and to remain in a self-forming community will help OSS project managers and organizations interested in the OSSD model take actions to attract and retain participants in order to achieve long-term success.

Secondly, as an increasing number of volunteer participants become connected online, forming webs of social relations in a digital environment, how do the online social

networks that participants are embedded in influence their behaviors? On one hand, developers interact with each other in the project team and form collaborative relationships with each other. On the other hand, in the user community participants interact with each other by engaging in communication activities such as seeking advice and sharing information. Individual behavior can be better understood from a network perspective that focuses on the influence of interpersonal relationships and interactions.

Therefore, the goal of this dissertation is to start a stream of research that investigates the voluntary decisions that OSSD participants make in open source project teams and communities from a network perspective.

1.3. Organization

This dissertation focuses on individual decisions with regard to two key elements of OSSD: the project team consisting of leaders and core developers who are responsible for coding, testing, and maintaining software and the project community consisting of users and developers who not formally associated with the team but contribute to the development and improvement of software at various levels.

This dissertation consists of two essays, each of which is a separate study. Each essay consists of an introduction, literature review, research hypotheses, research methodology, results, and conclusion.

Building upon findings of motivations of OSS developers and theoretical perspectives from group formation studies in organizational behavior, social psychology, and sociology, Chapter 2 explores how a project team is formed after the project is initiated by focusing on the influence of OSSD collaborative network on developers' choices of newly-initiated projects to participate in. Specifically, the study empirically tests the impact of past collaborative ties with the network members and perceived status of project members in the network on the formation of OSSD teams. This chapter focuses

on newly initiated projects so that the formation of teams and take-off of projects can be traced from the very beginning.

Drawing from information systems continuance theory, online community literature, and studies of OSS community organization, Chapter 3 investigates another key element in OSSD – the user community. In order for an OSS project to thrive, it needs to not only attract contributors to its community but more importantly to sustain their participation in the community. Hence, this chapter attempts to identify the factors driving members' continued participation in OSSD communities by focusing on the influence of community benefit provision through communication activities and the impact of member involvement in the software production process. The study extends the OSS research by identifying when and under what conditions individuals continue their active participation in OSSD user communities.

We then conclude the dissertation by summarizing our findings, discussing their implications, and identifying future research directions in Chapter 4

CHAPTER 2. EMERGENCE OF NEW PROJECT TEAMS FROM OPEN SOURCE SOFTWARE DEVELOPER NETWORKS: IMPACT OF PRIOR COLLABORATION TIES[1]

2.1. Introduction

Enabled by the unprecedented expansion of computer networks and rapid advances in modern telecommunication technologies, individuals are taking a more active role in the production and sharing of information and knowledge. Large groups of geographically dispersed people are collaborating online to produce information goods and services ranging from software to free encyclopedias without relying on either markets or firms and this phenomenon represents a new way of organizing production - commons-based peer production (Benkler 2006). Furthermore, in many other disciplines, especially marketing, users have also identified as a major source of innovation (Prahalad and Ramaswamy 2000). Having recognized the value of knowledge residing in end users, an increasing number of companies are trying to actively involve end users in the product development process by building communities of creation and encouraging co-creation with users (Gibbert et al. 2002). In both peer production and value co-creation process with end users, individuals participate voluntarily and self-select into various tasks or user groups. To better understand and sustain peer production and value co-creation processes with users, it is essential to understand what factors help attract volunteers into these groups and how these groups take form because individuals' decisions about which groups to join impact the aggregate group composition, which has been shown to be a

---

[1] Reprinted by permission, Jungpil Hahn, Jae Yun Moon, and Chen Zhang, Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties, Information Systems Research, volume 19, number 3, September, 2008. Copyright 2008, the Institute for Operations Research and the Management Sciences, 7240 Parkway Drive, Suite 300, Hanover, Maryland 21076, USA.

key determinant of group performance (Beal et al. 2003, Gruenfeld et al. 1996, Levine and Moreland 1990).

In this paper, we study how voluntary software project teams emerge from the social networks within which they are embedded in the context of open source software development (OSSD). This context is chosen because it represents a broader general phenomenon of theoretical interest and because of its practical significance in understanding team formation mechanisms that have the potential to impact OSSD success.

Theoretically, the study of OSSD project team formation enables us to examine how the existing social relationships in the network shape the emergence of new teams. Just as the social position of an individual within a network of peers influences his career advancement opportunities (Burt 1992), or as the social position of a firm within a network of organizations influences its alliance strategies and subsequent outcomes (Gulati 1995, Powell et al. 1996), the composition of an OSSD project team in terms of its developers and their positions vis-à-vis the OSSD network may influence its technical and commercial success through its impact on the likelihood of attracting developer attention (Grewal et al. 2006). Because the position of a developer in the OSSD network is a result of joint participation with other developers in past projects, we theorize that past collaborations in OSSD will impact how new project teams take form. As a case in point, the original Linux operating system kernel development group was not formed out of a social vacuum. Linus Torvalds, the project initiator, had been an active member in a related community of minix programmers and it was out of this community that the first volunteer developers emerged to form the Linux kernel developer team (Moon and Sproull 2002). A recent study that examined how sub-projects emerged within the Apache Web Server project found that most included at least a few developers who had worked together previously on other Apache sub-projects (Weiss et al. 2006). In short, the OSSD community can be regarded as a dynamic collaborative network of developers whose structure of connections shapes and is reshaped through the formation of new

project teams. The resultant structure in turn influences project outcomes through its impact on access to developers and other critical resources embedded in the network. Thus, project team emergence from OSSD communities is a natural context for studying how individuals are influenced by the network within which they are embedded. However, despite the relevance and importance of the collaborative network in OSSD, few studies have examined its antecedents and impact on project team formation mechanisms.

Practically, understanding team assembly mechanisms will have implications for the increasing number of OSSD projects, many of which fail to take off and become abandoned (Chengalur-Smith and Sidorova 2003). OSS project success can be measured through the extent to which it attracts and sustains volunteer developer interest and active contributions (Crowston et al. 2006). However, unlike organizational software development teams that are usually formed by managers who consider developer skills and experiences, an OSSD project team must rely on developers voluntarily choosing to become members of the project (Grewal et al. 2006, von Hippel and von Krogh 2003). While considerable research has been conducted to examine developer motivations to participate in the open source movement in general (e.g., Hars and Ou 2002, Hertel et al. 2003, Lerner and Tirole 2002, Roberts et al. 2006), little research has examined developer decisions as to which project to join and when to join. Examining developers' particular choices can extend this stream of research and provide a deeper understanding of developer motivations than examining their general attitudes toward OSSD.

Recent research has begun to examine how factors such as the restrictiveness of the software distribution license and the nature of organizational sponsorship may interact in influencing user interest and developer motivations to contribute to established OSSD projects (Stewart et al. 2006). There is a paucity of research, however, that examines what determines which new projects developers join or the birth of project teams. In this paper, we focus on such new project joining behaviors because of theoretical, practical, and methodological concerns. Theoretically, although most prior studies have focused on

how established teams develop over time, findings of which can help us understand how and why new members decide to join such teams, little can be said about how those teams were created and became established in the first place. Focusing on newly created projects would enable us to study team formation from a new light without the possible confounding effects of the characteristics of established teams. In terms of practice, because early marshalling of developer interest is critical for ensuring sustainability of OSS projects (Raymond 2001), examining developer choice of newly-initiated projects will not only contribute to a deeper understanding of the ecology of OSSD projects but also help devise practical guidelines for project managers to increase project attractiveness to developers. Furthermore, member composition can also affect software development performance through its impact on administrative and expertise coordination effectiveness (Faraj and Sproull 2000). Software development performance is also affected by the programming skills and application domain experiences that the team members bring to the project (Boehm 1987, Curtis et al. 1988). Therefore, from a practical standpoint, understanding new project team formation may provide insights on key problems related to early staffing of new OSSD projects and subsequent performance. Finally, in terms of research methodology, focusing on new projects enables us to construct a unique dataset of team formation with (almost) continuous tracking of developer joining events. This provides richer and more accurate insight into the team formation dynamics than would be possible with coarse-grained archival data of established projects where the sequence of member joining is imprecise, resulting in inaccurate inferences regarding the factors influencing joining decisions.

The remainder of this essay is organized as follows. In the next section, we present our theoretical background and develop our research hypotheses. Next we outline our methods and present the results. We conclude by discussing the implications, contributions and directions for future research.

2.2. Theoretical Development and Hypotheses

2.2.1. A Social Network Perspective of Open Source Software Development

The OSSD community can be understood as a collaborative network of developers working together in different project teams. Developers participate in one or more OSSD project teams and in doing so develop ties with other team members. The social nature of development activities in OSSD where developers and users form a complex network of relationships via various electronic communication channels on the Internet (von Hippel and von Krogh 2003) has spawned a set of studies examining this phenomenon using a social network perspective. Madey, Freeh, and Tynan (2002) conduct one of the first empirical investigations of the open source movement from the social network perspective and find that the OSSD community exhibits properties of self-organizing networks. Others have proposed methodological applications of social network analysis to data obtained from concurrent versions systems (CVS) code repositories of OSS projects (Lopez-Fernandez et al. 2004). Xu, Gao, Christley, and Madey (2005) explore social network properties in the open source community to identify patterns of collaborations. While early work from the social network perspective has mostly attempted to provide descriptive accounts of the network properties of the OSSD community, more recent studies have started to theorize the impact of social relations on OSSD outcomes. Grewal et al. (2006), for example, find that the network embeddedness of OSSD projects and their managers vary largely across a set of projects that have adopted the Perl programming language as the platform technology, and that this variation resulted in differential success rates. We contribute to this growing stream of studies by examining why and how the structure of existing collaborative ties in OSSD networks impacts individual developer decisions regarding the choice of new projects to join, which in turn may result in the formation of new collaborative ties.

Because of this focus on collaborative ties, we also adopt a social network perspective. Social network analysis aims to understand the implications of the relationships between

people, groups, organizations, and other types of social entities (Granovetter 1973, Wasserman and Galaskiewicz 1994, Wellman and Berkowitz 1998). A social network is modeled as a graph with nodes representing the individual actors (i.e., OSSD developers) in the network and arcs representing the relationships or ties between the actors (i.e., collaboration in an OSSD project). In a social network, the actors develop and maintain a tie by exchanging either tangible or intangible resources such as goods, services and information. In the OSSD network, the developers form ties by virtue of collaborating with other developers on software projects. The ties may vary in strength depending on a number of factors such as the amount of time, the emotional intensity, the intimacy, and the reciprocal services associated with the relationship (Granovetter 1973). 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) whereas weak ties are maintained infrequently or indirectly between the actors who belong to different social clusters. Both strong ties and weak ties play an important but differential role in a social network. Strong ties maintain and promote trust and collaboration whereas weak ties enable actors to access resources and information that may be unavailable in their immediate social circles (Burt 1992, Granovetter 1973).

In this study, we focus on prior collaborative ties among developers as a potential driver behind developer joining behavior and project team formation; in short, we examine how collaborative networks impact the project teams that emerge. When developers decide to join a project, in addition to factors regarding the use value of the project (Hertel et al. 2003, Roberts et al. 2006) and expected enjoyment and learning benefits (Hars and Ou 2002, Lakhani and Wolf 2005), they may also consider factors that are related to the composition of the project team as this may impact how the development process will unfold. For example, a developer may be concerned about issues related to coordination and communication with other team members since much of the collaboration occurs asynchronously over computer-mediated distributed networks. In general, people prefer to work with those with whom they have worked in the past because of the reduced uncertainty stemming from familiarity based on prior collaborative experiences (Hinds et

al. 2000). In addition, the social capital that the project's existing developers have developed by virtue of having collaborated with other developers in the overall OSSD network can affect project outcome through access to information and other resources critical for effective software development (e.g., Grewal et al. 2006), and may serve as a potential source of reputation enhancement by association for joining developers (Stewart 2005).

2.2.2. Perspectives on How Prior Collaborative Ties Affect Project Team Formation

In this section we integrate emergent findings from research on motivations of OSS developers with theoretical perspectives derived from organizational behavior, social psychology and sociology that have examined the mechanisms determining the formation of naturally occurring groups both within and outside organizations (e.g., Ruef et al. 2003). Conventionally, project teams in organizations are strategically formed by a manager who assigns individuals to a team based on characteristics such as expertise and personality, after which team members go through several distinct phases such as forming, storming, norming, performing, and adjourning (Tuckman 1965, Tuckman and Jensen 1977). Alternatively team members may self-select into teams. In OSSD, projects may recruit developers both formally (e.g., by broadcasting position openings and required qualifications to the OSS community),[2] and informally (e.g., by inviting other developers to join) or developers may voluntarily ask to join a project team.

Research on group formation – be it work or social, self-organized or prescribed – indicates that group formation is a result of the deliberate, strategic decisions of individuals who either self-select or assign others to a group with the purpose of

---

[2] Interestingly, the extent of formal recruiting is surprisingly low (von Hippel and von Krogh 2003). For example, based on our observation there were only about 200 position openings posted on SourceForge.net at any given time. This figure is quite inconsequential when we consider that there are over 150,000 projects of which several thousand have substantial development activity. For example, we observed over 3,000 projects that had at least 10 code commits between June 1 and June 15, 2007.

satisfying individual and group objectives (Owens et al. 1998). In naturally forming groups, the process is typically bi-directional – the individual seeks to find and join a group that will satisfy his goals and needs, the group seeks individuals that fit and fulfill its goals (Levine and Moreland 1990). Similarly, in the OSSD context, developers are motivated to choose projects and collaborators that can afford them ample opportunities to realize the intended benefits of participation; projects need developers with the required skills and experience to move development forward. At the outset we acknowledge that both developers and projects may evaluate the other in the process of project team formation (i.e., either developers request to join and project administrators decide whether or not to accept them, or project members invite developers who then decide whether or not to join), but we find little evidence of project administrators either rejecting developer requests to join or actively inviting developers to join in the context of OSSD projects.[3] Hence, we frame the remainder of our discussion in terms of developer self-selection into projects.

Developer participation in OSSD is driven by motivations ranging from enjoyment and learning from development activities, a sense of obligation toward the OSS community, a belief that code should be open, a need for the software developed, to a desire for reputation (Lakhani and Wolf 2005). Maximal realization of these goals depends on the success of projects to which they contribute, both in terms of outcome (i.e., sustained successful development of code), as well as process (i.e., how well project members work together). The former is important if one is to fulfill the need for software as well as give back to the OSS community; the latter will likely influence the degree to which participants will learn and enjoy participating in OSSD. However, from the perspective of prospective developers, it is difficult to predict the likelihood of future project success because of the inherent uncertainty in judging the true objective quality of the project and

[3] We surveyed the administrators of our study sample projects and received 384 responses with a response rate of 16.3%. Most respondents were receptive of developer join requests. Refer to Appendix A for details of the survey.

the programming skills, project administration skills, and work styles of current project team members. In order to deal with this uncertainty, developers will rely on other cues and attributes as proxies of both the project and its team members' underlying quality. Prior work on self-organizing networks suggests that people typically rely on two social cues – cohesion and status – to determine the likelihood of successful collaboration when the quality of a future collaborator is difficult to measure directly (Guimerà et al. 2005, Uzzi et al. 2007). Cohesion is related to preference for repeat collaborations in order to benefit from close work relationships; status is related to preference for collaborations with experienced, well-established actors in the network.

Past collaborative ties play a critical role in shaping both cohesion and status cues. Cohesion cues result from direct past ties between the prospective developer and existing project members; status cues result from the past collaborative ties of existing project members with the broader OSS developer network. Developers rely on cohesion cues because they are motivated to continue to work with those with whom they have successfully collaborated in the past. Cohesive ties from past interactions can result in greater trust and knowledge of the technical and organizational skills possessed by the potential collaborators (Uzzi and Spiro 2005), and hence influence the developer's perception of the likelihood of project success – both in terms of the collaboration process and outcome of the project. Developers rely on status cues because they are motivated to work with those whom they perceive to be successful. Status of existing project members as observed through their ties in the network can signal their expertise and quality as vetted by developers in the overall OSSD network, and hence affect the prospective developer's perception of the likelihood that the project will be successful. Thus, we propose that past collaborative ties of the existing developers – with the potential joiner and with other developers in the OSSD network – will influence developers' decisions to join the project. We explore how collaborative ties relate to these two types of cues, and as a consequence affect project team formation in more detail below.

2.2.2.1. Cohesion Cues: Prior Collaborative Ties between Developer and Existing Project Members

Research suggests that people are more likely to work together when they have prior social ties (McClelland et al. 1953, Schachter 1959) and that past collaborative ties tend to be repeated in the future (e.g. Kogut 1989, Uzzi and Spiro 2005). When choosing future collaborators, a developer may rely on ties developed through joint participation in past projects as a social cue to assess their underlying quality and the risks involved in working with them. Prior collaborations help him obtain information about the skills and capabilities possessed by others (Granovetter 1985, Coleman 1990). Hence, he can better rely on personal experiences with the existing project members to judge the likelihood of the successful outcome of the new project. Past experiences with the project members may also help the developer better estimate the risk and uncertainty involved in the collaboration process. Teams consisting of individuals with preexisting relationships have been shown to solve complex problems better than teams of strangers because they are able to pool information more efficiently (Gruenfeld et al. 1996). 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, Faraj and Sproull 2000). In the OSSD context in particular, due to the lack of opportunities for face-to-face contact, developers face greater barriers to effective communication and coordination and are thus more likely to be concerned about these issues (Kotlarsky and Oshri 2005). Therefore, developers will be more likely to repeat interactions with developers whom they have worked with in the past, thereby reinforcing collaborative ties. As developers interact more frequently, the strength of the collaborative tie will increase as they develop closer and more cohesive working relationships (Granovetter 1973, Hansen 1999), further increasing the likelihood that they will collaborate in the future. Thus, we hypothesize:

H1: The likelihood that a developer will join a project is positively related to the strength of his collaborative tie with the project members, controlling for outcome of his past collaborations with the project members.

Not all past collaborative experiences are necessarily successful and many projects fail due to conflict between project members, among other factors. Research on group formation in laboratory and field settings has 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) and that successful outcomes sustain group cohesion (Boone et al. 1997). If developers are motivated to maximize the chance that they gain the expected benefits from OSS project participation, then they will be more likely to choose the same collaborator if the past experience with him was satisfactory not only in terms of the quality of the collaboration process but also in terms of final project outcome. In addition, positive collaboration experiences may have created implicit obligations for future exchange due to the benefits that the developer derived from association with the project members in the past, in particular if the past collaboration experience was one in which the current project members contributed substantially to the developer's past project. Thus, we hypothesize:

> *H2: The likelihood that a developer will join a project is positively related to the outcome of his past collaborations with the project members, controlling for the strength of his collaborative tie with the project members.*

2.2.2.2. Status Cues: Prior Collaborative Ties between Existing Project Members and OSSD Network

In addition to knowledge and impressions formed through direct prior collaborations with the project members, a developer may also rely on status cues, other observable proxies to judge the quality of the prospective collaborators, in particular in the absence of direct past contact. Here status represents an individual's prestige or honor in the social network (Weber 1968). Social scientists have argued that the perceived status of an individual is associated with his relationship with others (Frank 1985) and that status can be based on the number of ties he has developed (Podolny 1993) or his position in the

social network (Stewart 2005). Prior literature in sociology has examined patterns in alliance formations and collaborative relation formation and found that high status actors with many connections in the network are more likely to attract partners than less connected actors (Moody 2004, Podolny 1993, Powell et al. 2005). In the OSSD context, the number of connections through past project participation that existing project team members have amassed may signal to others their ability for valuable contributions to OSSD projects. High status (i.e., highly connected) developers may have gained a vast store of experience through their past collaborations with others in the network, and thus will be perceived to be more competent and more likely to succeed (Stewart 2005, Thye 2000). In addition to serving as a proxy of the underlying quality of project members, the number of prior collaborative ties may also provide the project with greater access to other OSSD network participants and their resources and hence increase the likelihood that the project will succeed. Therefore, in OSSD, we propose that developers will prefer participating in projects whose existing members are perceived to have higher status based on their prior collaboration experiences in the OSSD network. Thus, we hypothesize:

> *H3: The likelihood that a developer joins a project is positively related to the perceived status of the project members in the open source software developer network.*

We tested our hypotheses empirically using data collected on a sample of newly initiated projects from an online project management platform for OSSD projects. We describe the data collection process, measures and analytical procedures in the following section.

2.3. Data and Methods

2.3.1. Study Sample

Data was collected from SourceForge.net, the largest repository of OSS projects on the Internet. At the time of data collection, SourceForge.net provided free hosting to more than 150,000 projects and more than 1,600,000 subscribers.[4] It also offers a variety of services to hosted projects such as mailing lists, bug trackers, message boards, file repositories, code CVS, and other project management tools. SourceForge.net has been an attractive source of data for many OSS researchers mainly due to the abundance of publicly accessible data (Howison and Crowston 2004).

We selected all public OSSD projects newly registered at SourceForge.net between September 30 and November 11, 2005 ($N = 2349$) as our sample of projects. A software crawler visited these projects' web pages hosted at SourceForge.net and kept track of project-related and membership information on a daily basis. This process not only enabled us to distinguish between the initiator and the developers who subsequently joined but also enabled us to capture the timing (i.e., the sequencing) of joining events as well as timing of important project events (e.g., release of code, updating of project description).[5] Daily tracking ended on January 5, 2006.

---

[4] Howison and Crowston (2004) and Rainer and Gale (2005) note however that the vast majority of these projects are inactive.

[5] Because SourceForge.net is set up in such a way that only one user can "officially" register a project, even if a project is jointly founded by two or more developers, the data would show that only one user is the initiator and the others are developers that subsequently joined, presumably very early or right after initiation. Hence, treating the other co-founders as external developers who joined the project after its initiation may lead to an underestimated impact of collaborative ties. Such co-founding situations would materialize in the dataset as a project initiation followed by a very rapid joining of a developer with ties to the initiator. In our sample we identified 20 (of 1198) joining events where developers joined within the first week and the joining developers had

At SourceForge.net a developer interested in starting a new project submits a request to the SourceForge.net staff. After the project is approved for hosting, the project initiator can start utilizing the services and tools provided by the site and upload contents to the project site. A developer wishing to become a member of the project first contacts the project administrator, who then either approves the joining request and adds him to the project membership or rejects the request. A joining event occurs when a developer's joining request is approved by the project administrator and the developer becomes a listed member of the project. A project becomes a *team* only after at least one developer joins the project. In other words, we distinguish between single-developer projects (i.e., the initiator is the only member) and multi-developer project teams (i.e., when additional developers have joined the project) and use the term *team* only when we refer to the latter.

Our data also includes all the developers who had participated in at least one project hosted at SourceForge.net prior to the new project data collection period (N = 170,741). We selected developers with project experience for two reasons. First, developers without prior project experience are unlikely to have developed the collaborative ties with other developers that are the focus of our study. Second, including the developers with past project experience enables us to operationalize measures of fit between developers' technical profiles and project technical requirements.[6]

Developers' collaborative ties were identified using data about projects and developers from the SourceForge Research Data Archive (SRDA) hosted at the University of Notre

---

past collaborative ties with the initiators (i.e., those likely to be co-founders) and discarded them from further analysis.

[6] Because only approximately 2% of developers make their technical skill profiles publicly accessible at SourceForge.net, we inferred their technical skills from their past project experience. The specific variables that we constructed for hypothesis testing are described in more detail in Section 2.3.2.

Dame (http://www.nd.edu/~oss/) (Antwerp and Madey 2008). Developers' past collaborators were identified through their co-memberships on past OSS projects. Based on this data, we constructed affiliation matrices of developers and projects that depict the existence of prior collaborative relationships between developers.

In summary, the final dataset used to construct our sample consists of 2349 projects and 170,741 developers. The unit of analysis is a triad consisting of a date, a project, and a developer – (*T*, *P*, *D*), in which *T* is a date between September 30, 2005 and January 5, 2006, *P* is one of 2349 projects, and *D* is one of 170,741 developers.

2.3.2. Measures

2.3.2.1. Dependent Variable

The final outcome of interest for a triad (*T*, *P*, *D*) is whether developer *D* joins project *P* on date *T*, which is coded as a binary variable (*Join*).

2.3.2.2. Independent Variables

2.3.2.2.1. Cohesion Cues: Strength and Outcome of Collaborative Ties

Measures of collaboration outcomes and the strength of collaborative ties between developers were computed based on developers' project histories. We defined past collaborators as those developers who had been concurrently listed as members of a common project prior to the focal date. In other words, developers who have been involved in the same project at the same time are coded as having developed collaborative relations with each other.[7] Given the self-organizing nature of OSSD

[7] This operationalization of past collaborative ties does not take into account whether the two developers have actually worked *collaboratively* during their concurrent membership in a project.

project teams, developers may join a project at their own discretion. However, when this joining decision is made, the set of project members may include the original initiator of the project as well as other developers who have joined the project before the focal developer makes his joining decision. In other words, he may join a project due to his sensitivity toward cohesion with the initiator or with the other developers who have already joined. Given the differences in roles that initiators (i.e., leadership or administrator roles) and other members play (i.e., co-worker roles), we distinguish between ties with the project initiator and with other developers that have subsequently joined the project prior to the focal developer joining the project. In other words, the focal developer may have a collaborative tie with the initiator and/or with the non-initiator members of the project.

For each pair of project member and focal developer we collected seven measures of past joint project activity that tapped into either the outcome of the prior collaboration or the strength of the collaboration tie. In order to operationalize collaborative tie outcome, we characterized the nature of the past collaborations based on measures of OSS project success (Crowston et al. 2006). The extent to which past collaborative experiences had been successful was measured by the amount of code released (*CodeBytes)*, the number of software downloads by users (*Downloads*), whether or not the project resulted in a successful release of the working program code (*HasRelease*), and the development status of the project (*DevelopmentStatus*). We measured strength of collaborative tie as a combination of the frequency of collaborative interactions and the closeness of the collaborative relationship (Cummings and Kiesler 2007, Granovetter 1973, Hansen 1999). 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* and *Duration*). Because we expect project administrators to interact more frequently and develop closer working relationships in order to coordinate the

---

Consequently, this operationalization would underestimate the impact of prior collaborative ties, rendering the test of hypotheses conservative.

project we also tracked whether the focal developer and project member jointly had project administration responsibilities in prior projects (*BothAdminRole*). Since a developer-initiator or developer-developer pair could have collaborated on multiple projects, we used the average of all measures except *NumCollaborations*.

We conducted factor analysis on these measures for all pairs of developers who have collaborated with each other at SourceForge.net to determine whether the joint project activity measures could be grouped into the higher order factors of collaborative tie strength and collaborative tie outcome as expected. Two factors emerged that collectively explained almost 70% of the variance (see Table 2.1). The first factor, Tie_{Oucome}, measures whether or not the *outcome* of the past collaboration between the focal developer and the project member was positive, that is whether the project was successful as measured through number of downloads, among other factors. The second factor, $Tie_{Strength}$, measures the closeness of the work relationship (*BothAdminRole)* and the frequency of interaction (*NumCollaborations)*. The resultant $Tie_{Strength}$ measure is similar to measures of tie strength adopted in studies of the impact of tie strength within organizations and reflects the frequency (*NumCollaborations)* and intensity (*BothAdminRoles)* of the joint collaboration (Cummings and Kiesler 2007, Hansen 1999). Contrary to what we expected, the duration of the joint projects loaded onto Tie_{Oucome} rather than $Tie_{Strength}$. We suspect that the actual project duration may not be an accurate indicator of the actual interaction frequency or intensity given the discretionary nature of participation in OSSD projects. Rather, since failed projects may be abandoned earlier, longer project durations may be indicative of positive project outcome. Hence, we included it as an item measure of $Tie_{Outcome}$.

The composite measures for $Tie_{Outcome}$ and $Tie_{Strength}$ were constructed using factor scores based on the factor loadings shown in Table 2.1. As discussed above, we distinguished between prior ties with the project initiator and those with other non-initiator members of the project team by prefixing the variables with *Initiator* or *Member*. Also, since there could be multiple non-initiator members that have already joined the project team, we

aggregated the measures by summing the values for all developer-member pairs.[8] Collaborative tie strength and outcome can be positive or negative depending on the nature of past collaborations and was set to zero when the focal developer and the existing project member had not jointly participated in any common projects.

Table 2.1 Composite Measures of Outcome and Strength of Prior Collaborative Ties with Project Members

	Factor 1	**Factor 2**
Variable	**$Tie_{Outcome}$**	**$Tie_{Strength}$**
Downloads	**0.958**	0.002
Duration	**0.647**	0.036
HasRelease	**0.893**	-0.078
DevelopmentStatus	**0.719**	-0.264
CodeBytes	**0.968**	0.034
BothAdminRole	-0.162	**0.695**
NumCollaborations	0.110	**0.826**
Eigenvalues	3.648	1.223
% Variance Explained	52.11 %	17.47 %
Cum. Variance Explained	52.11 %	69.58 %

2.3.2.2.2. Status Cues

Project members' status cues result from their prior project collaboration experiences within the SourceForge.net OSSD network. We constructed a measure of each project member's ties based on the number of distinct developers in the OSSD network with whom he has had previous joint project collaboration experience at SourceForge.net (*TieAmount*). Again we distinguished between ties of the initiator and those of the non-initiator members and prefix the variable with *Initiator* or *Member*. Also, since there could be multiple non-initiator members in the project team, we used the sum of ties across all members.[9]

---

[8] We used other aggregate measures such as maximum (for best experience) and minimum (for worst experience) to compute the tie measures between developers and non-initiator members. Sensitivity analyses presented in Table B.1 in Appendix B suggested that the results reported are robust to different ways of aggregating these measures.

[9] We also tried including developers' indirect ties (i.e., ties with other developers through a common collaborator). However, the correlation between the *direct* tie amount and the *indirect* tie amount was very high ($\rho = 0.85$ for initiators and $\rho = 0.94$ for other pre-existing members).

In addition, although our focus was on status cues resulting from developers' prior collaborative ties, we also included measures of developers' experience since these measures may also influence others' perception of developers' status within the OSSD community. We measured the prior project experiences of the project initiator and non-initiator members using the active duration since their first actual participation in an OSSD project (other than the focal project) at SourceForge.net (*InitiatorActiveTime* and *MemberActiveTime*) as well as the number of projects on SourceForge.net in which they have participated in the past (*InitiatorProjects* and *MemberProjects*).[10] Again, in cases where there are multiple non-initiator members in the project team, we used the sum of experience across all members.

2.3.2.3. Control Variables

While our study focuses on examining the impact of cohesion and status cues shaped by prior collaboration experiences of developers within the OSSD network, developers may also rely on other attributes in evaluating which project to join. Hence, we controlled for three categories of factors that may influence this choice – factors related to the project in question, those related to the focal developer, and those that relate to the fit between the project and the focal developer.

Consequently, our analyses were based on the operationalization that only accounts for the number of direct ties. Sensitivity analyses suggest that the results are robust when the number of indirect ties is used instead of the number of direct ties.

[10] Although the project founding experience of the initiator may also be an indicator of status and influence whether developers join the project, only one third of the projects hosted at SourceForge.net have their initiator information recorded in the database. Thus, due to data unavailability we were unable to consider the initiator's experience in terms of the number of projects he had founded in the past.

2.3.2.3.1. Project-Related Factors

Code Availability**.** The presence of software code may reduce the uncertainty regarding the nature of the project as well as the ability of the project members to translate project goals into actual working code. The source code also gives developers the opportunity to obtain additional technical information regarding the project itself. Some projects may be registered on SourceForge.net only after some working code has been independently developed by the initiator. In other cases, a project may be a fork of an existing project hosted at SourceForge.net or other sites. In both cases, projects would be more likely to have code present either before or within a short period of project initiation on SourceForge.net. Hence, we used a binary variable *PreexistingCode* to control for the initial presence of a project's source code. *PreexistingCode* indicates whether the project's first release date was prior to its registration date. We also controlled for whether the project has released software code (*CodeReleased*) after the project was registered at SourceForge.net by the focal date, that is, the date on which the focal developer joined the project.

Project Activity. We also controlled for other project attributes that would affect the visibility of the project to potential developers. For example, a factor that may influence project visibility on SourceForge.net is the information related to project ranking. SourceForge.net calculates project scores based on traffic, communication and development statistics. Each project's score is compared to those of other projects for the same time period to determine its ranking. Projects with high activity ranks are included in SourceForge.net's top project listing, which makes them more visible to potential developers. To capture this, we use a categorical variable (*ActivityPercentile*) that distinguishes between low, medium and high rankings.[11]

[11] Each project's ranking percentile information is displayed on its summary page. As presented in Table B.2, we also tried the nominal scale for *ActivityPercentile* and the results are unchanged.

Project Popularity. Developers may consider indicators of project popularity when deciding which projects are likely to result in positive outcomes. For example, developers may rely on the expressed preferences of other developers to reduce the uncertainty regarding viable project choices and perceive more favorably those projects where others have already joined. Therefore, we also controlled for *TeamSize,* operationalized as the number of developers in the project on the focal date. Project popularity may also be partially determined by the software domain. Some software domains may be more popular than others, increasing their attractiveness to potential developers. We controlled for domain popularity using the ratio *DomainPopularity*, which was computed as the proportion of developers for the top 1000 projects who work on the focal project's domain. For example, if a total of 2000 developers participate in the top 1000 projects at SourceForge.net and among the 2000 developers 500 are members of the projects in the "Internet" domain whereas 200 are members of the "security" domain, *DomainPopularity* would be 0.4 for the Internet domain (i.e., 500 / 2000 = 0.4) and 0.1 for the security domain (i.e., 200 / 2000 = 0.1).

Project Information Availability. We included project attributes affecting the amount of information that is available to developers. We measured the level of details available in the project description that would facilitate information gathering required for making a joining decision using *DescriptionLength*, which is operationalized as the number of characters in the project description.

Other Controls. We measured other factors that may influence developer decisions to join a particular project. Because the availability of donations as a source of external resources may influence a project's attractiveness to potential developers, we controlled for whether a project accepts donations from users using a binary variable *AcceptDonation*. Finally, *ProjectAge* represents the number of days between the project's registration and the focal date. Some developers may prefer joining a relatively new project whereas others may prefer waiting for a period of time to observe the progress of the project and reduce any uncertainty before making their joining decisions.

2.3.2.3.2. Developer-Related Factors

Besides the project attributes as discussed above, we also controlled for additional factors related to the focal developer that may affect his decision to join a particular project. A developer who has been actively participating in projects and has more experience in OSSD may consider new projects differently compared to less experienced developers. Therefore, we measured the experience of the developer in terms of how long he has participated in OSSD at SourceForge.net. *DeveloperActiveTime* is the number of months since he first became a member of an OSSD project on SourceForge.net.[12]

2.3.2.3.3. Project-Developer-Related Factors

We considered the fit between developers' skills and the focal project's requirements in order to control for the impact of developer expertise and interest on project selection. These were captured in four variables that reflected how well the technical details in terms of domain (*DomainMatch*), programming language (*ProgrammingLanguageMatch),* intended audience (*AudienceMatch*), and application platform (*OperatingSystemMatch*) of developers' past projects matched those of the focal project.

---

[12] Although we measured initiators and other project members' experience using both the number of prior projects as well as the time engaged in project participation, we only used time-based experience to control for developer experience. This is because, as mentioned earlier, when constructing the sample we sampled only the developers with past project experience in order to be able to compute the matching variables based on the properties of their past projects. However, since the event sample consisted of not only experienced but also inexperienced developers, controlling for developers' project experience in the analysis would lead to a large but biased estimate due to the fact that their project experience was already controlled for during the sampling process. Therefore, we did not include the focal developer's experience in terms of projects in the analysis.

To test the robustness of the results with regard to our sampling of developers with some experience on SourceForge.net, we constructed a new set of samples by randomly sampling all developers who have registerered at SourceForge.net regardless of their past project experience. As presented in Table B.3, the major findings related to H1, H2, and H3 still hold.

Because using binary measures to represent whether at least one past project that the developer participated in matches the technical detail of the focal project may introduce a bias in the likelihood of fit toward more experienced developers, we computed the matching variables in such a way as to be less sensitive to developers' experience. The *DomainMatch* scores were calculated as the percentage of a developer's past projects that have at least one domain in common with the focal project. If the focal project is related to multiple areas (e.g. both Databases and Internet), we computed the dominant match for each of these areas and selected the maximum so as to capture the *dominant* matching concept. For instance, assume the focal project's domains include D1, D2 and D3, and the developer's past projects include P1 and P2. Further assume P1 covers domains D1 and D4, and P2 covers domains D1 and D2. The matching score for D1 would be 1 because all of the developer's past projects are related to D1; the score for D2 would be 0.5 because only one of the two prior projects covers D2; the score for D3 would be 0 since none of the developer's past projects is related to D3. Thus, *DomainMatch* for the project developer pair would be 1, the maximum of the scores for D1 (1), D2 (0.5) and D3 (0). The same approach was applied to compute the match on other technical details. These scores are continuous values ranging from 0 to 1 with higher values indicating higher degree of match.[13]

---

[13] A developer's interests may change over time, in which case considering all of his past projects would not faithfully represent his true interests. In such cases, constructing the match variables with a developer's most recent project may be more accurate. We tried this alternative method – using only the developer's most recent project to calculate the match between the requirements of the focal project and the developer's interests. The match is the percentage of the technical details of the focal project that match those of the developer's most recent project. For example, a focal project defined its domains to be A, B, C, and D whereas the developer's most recent project defined its domains as C and E. The domain match variable would be 0.25. These values of the match variables based on the most recent project would range from 0 to 1. However, the correlation between the dominant and recent match scores was very high ($\rho = 0.90$) suggesting that, at least for developers in our dataset, their interests were quite stable over time. Table B.4 shows that when we included the most recent match scores in the analysis instead of the most dominant scores similar results were obtained.

In addition, because domains and operating systems are defined at several levels on SourceForge.net, we computed the matching variables for these two types of technical details at both the top level and the second level, which are highly correlated (*Domain*: $\rho = 0.89$, $p < 0.001$; *OS*: $\rho = 0.64$, $p < 0.001$). Hence, we only kept the top level domain matching and OS matching variables in subsequent analyses. The summary of the measures are shown in Table 2.2.

Table 2.2 Summary of Measures

Variable	Operational Definition
Join (DV)	Binary variable, which equals 1 if the developer joined the focal project on a specific date (focal date below) (between September 30, 2005 and January 5, 2006), 0 otherwise.
$InitiatorTie_{Outcome}$	The outcome of a collaborative tie with the project initiator that is associated with the success level of the project. See Table 2.1.
$InitiatorTie_{Strength}$	The strength of a collaborative tie with the project initiator that is related to the frequency and closeness of past collaboration. See Table 2.1.
$MemberTie_{Outcome}$	The sum of the outcome of collaborative ties with the non-initiator members associated with the success level of the project. See Table 2.1.
$MemberTie_{Strength}$	The sum of the strength of collaborative ties with the non-initiator members related to the frequency and closeness of past collaboration. See Table 2.1.
InitiatorTieAmount	The natural log of the number of distinct developers (+1) who have collaborated with the project initiator on OSS projects at SourceForge.net.
MemberTieAmount	The natural log of the number of distinct developers (+1) who have collaborated with the non-initiators on OSS projects at SourceForge.net.
InitiatorActiveTime	The number of months since the project initiator first became involved in a project as a member at SourceForge.net.
MemberActiveTime	The average of the non-initiators' experience in terms of the number of months since they first became involved as members of a project at SourceForge.net.
InitiatorProjects	The natural log of the number of prior projects (+1) that the project initiator has participated in at SourceForge.net.
MemberProjects	The average of non-initiators' experience in terms of the natural log of the average number of prior projects (+1) that they have participated in on SourceForge.net.
DomainPopularity[b]	Popularity of the project's domain as measured by proportion of developers working on the top 1000 projects within the domain category. *DomainPopularity* is set to 0 if the project's domain is undefined.
ActivityPercentile[c]	A categorical variable based on percentage of projects hosted on SourceForge.net with lower project scores than the focal project – "low" for projects with activity scores below the 50th percentile, "medium" for those between the 50th and 90th percentiles, and "high" for those above the 90th percentile.
DescriptionLength[b]	The natural log of the number of characters in the project description.
AcceptDonation[b]	Indicator variable that is 1 if the project has been set up to accept donations from users, 0 otherwise.
CodeReleased[c]	Indicator variable that is 1 if the project has released code prior to the focal date, 0 otherwise.
TeamSize[c]	The natural log of the number of developers (including project initiator) who are listed as project members prior to the focal date.
DeveloperActiveTime	The number of months since the focal developer first became involved as a member of a project at SourceForge.net.
DomainMatch	Percentage of the developer's prior OSS projects that have at least one domain in common with the focal project.
ProgrammingLanguageMatch	Percentage of the developer's prior OSS projects that have at least one programming language in common with the focal project.
AudienceMatch	Percentage of the developer's prior OSS projects that have at least one type of intended audience in common with the focal project.
OperatingSystemMatch	Percentage of the developer's prior OSS projects that have at least one type of operating system in common with the focal project.
ProjectAge[c]	The natural log of the number of days since the project was registered at SourceForge.net until the focal date.
PreexistingCode[b]	Indicator variable that takes 1 if the project's earliest release date is prior to its registration at SourceForge.net, 0 otherwise.
Notes: [(a)] All four *tie* variables are coded as 0 for developer-initiator or developer-developer pairs without prior ties. [(b)] Time-invariant project attribute measures. [(c)] Time-variant project attribute measures.	

2.3.3. Analytical Procedures

Given that the dependent variable is a binary variable representing the joining event, we conducted logistic regression analysis to test our hypotheses. However, the conventional logistic regression approach with random sampling is inappropriate here due to the rarity of the joining event, which causes logistic regression to underestimate event probabilities (King and Zeng 2001). One reason for the unreliable estimates under random sampling is that the maximum likelihood estimators obtained by logistic regression are biased not only in samples with fewer than 200 observations but also in large samples where the proportion of positive outcomes in the samples is very small. Random sampling, in which the selection rule is independent of all other variables, tends to generate too few instances of the event in the sample to make logistic regression analysis an optimal approach. For instance, with approximately 2,300 sample projects, 170,000 sample developers and 100 days, there would be over 39 billion (i.e., $2{,}300 \times 170{,}000 \times 100$) date-project-developer triads in total, of which the joining event occurs only in approximately 1,200 instances (i.e., less than 3×10^{-6} % of cases).

A data sampling strategy to overcome these problems is choice-based (or endogenous stratified) sampling (King and Zeng 2001, Manski and Lerman 1977). Unlike random sampling, *choice-based sampling* strategically selects observations based on values of the response variable Y. Compared with observations where the response variable is 0, observations with a response variable of 1 carry much more information for the estimation of the variables influencing the occurrence of the event. The strategy is to construct a sample by collecting a fraction e of the observations with $Y = 1$ (the event sample) and a fraction c of the observations with $Y = 0$ (the non-event controls), such that e is much larger than c.

Our dataset includes 1178 observations representing events where a developer joined a project on a particular date (i.e., *Join* = 1). We then matched each event triad with five

control triads whose date is the same as the focal date in the event triad. We also ensured that control triads have similar (or dissimilar) characteristics as the event triads. In particular, for one control triad we ensured that the existence (or nonexistence) of prior collaborative ties between the developer and the initiator was the same as that in the event triad. In another control triad the existence (or nonexistence) of prior collaborative ties between the developer and the non-initiator members was the same as that in the event triad. In addition, three random triads are selected for each event triad. The choice-based sampling procedure produced a sample of approximately 7,000 triads.[14]

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). This approach weighs each observation in the sample with the number of population observations that it represents and obtains the WESML estimator by maximizing a weighted pseudo-likelihood function.[15]

For example, the weight of a sample triad that represents 100 potential triads in the entire population is assigned 10 times the weight of a sample triad that represents 10 population triads. In addition, because in the choice-based sampling process a project may be sampled multiple times, we used the generalized estimating equations (GEE) method (Liang and Zeger 1986) to calculate the standard errors without assuming independent errors among observations.

In order to ensure the robustness of the estimation results with respect to the choice-based sampling procedure, we drew 1000 bootstrap choice-based samples of approximately 7000 observations to derive the bootstrap mean and the confidence intervals for each

---

[14] Since we have 5 control triads for each event triad, the choice-based sample size is $1178 \times 6 = 7068$. However, due to possibility of missing controls (i.e., when an observation that meets the control sampling criteria does not exist), a few observations may be dropped.

[15] Refer to King and Zeng (2001) for more technical details on WESML.

parameter estimate (Efron 1982, Efron and Tibshirani 1986). Statistical significance of the parameter estimates is based on the bootstrap confidence intervals.

2.4. Results

2.4.1. Data Sample and Descriptive Statistics

Table 2.3 presents the distribution of software domains of the projects in our sample. The sample projects covered all 19 top-level domain categories, among which *Software Development*, *Internet*, and *System* had the highest frequencies. Table 2.4 summarizes the descriptive statistics obtained from the 1000 bootstrap samples. A total of 520 projects had attracted additional developers during the data collection period and on average about 2 additional developers joined each of these projects.[16]

[16] Some projects may be founded without any intentions to continue development or attract developers to the team (e.g., class projects). We conducted a robustness check by including only the 520 projects that had successfully attracted additional developers. The major findings presented in Table B.5 are largely consistent with slight differences in the level of statistical significance of some parameter estimates.

Table 2.3 Domain Distribution of Sample Projects

Category	Freq	%	Category	Freq	%
Software Development	276	20.32	Education	48	3.53
Internet	245	18.04	Security	42	3.09
System	206	15.17	Desktop Environment	29	2.14
Communications	166	12.22	Text Editors	25	1.84
Games/Entertainment	161	11.86	Other	17	1.25
Scientific/Engineering	154	11.34	Printing	8	0.59
Multimedia	136	10.01	Terminals	6	0.44
Office/Business	111	8.17	Religion and Philosophy	2	0.15
Database	66	4.86	Sociology	1	0.07
Formats and Protocols	62	4.57			
Notes: The total number of projects is 1358, which includes only those projects that have defined their domain during our data collection period. Also since projects can have multiple overlapping domains, the cumulative frequency is greater than 100%.					

Table 2.4 Descriptive Statistics

Variable	Mean	St. Dev	Min	Max
Join	0.17	0.377	0.00	1.00
$InitiatorTie_{Outcome}$	0.00	0.117	-2.53	2.27
$InitiatorTie_{Strength}$	0.02	0.611	-0.50	42.90
$MemberTie_{Outcome}$	0.00	0.120	-2.29	1.64
$MemberTie_{Strength}$	0.02	0.349	-0.89	13.30
InitiatorTieAmount	0.48	1.058	0.00	5.84
MemberTieAmount	0.30	0.977	0.00	6.67
InitiatorActiveTime	18.03	13.780	0.00	73.64
MemberActiveTime	5.65	10.455	0.00	73.36
InitiatorProjects	0.40	0.596	0.00	3.06
MemberProjects	0.10	0.361	0.00	2.83
DomainPopularity	0.06	0.088	0.00	0.56
ActivityPercentile	2.19	0.489	1.00	3.00
DescriptionLength	4.97	0.633	1.73	5.62
AcceptDonation	0.08	0.270	0.00	1.00
CodeReleased	0.03	0.161	0.00	1.00
TeamSize	0.40	0.683	0.00	3.71
DeveloperActiveTime	27.73	20.684	0.00	73.30
DomainMatch	0.02	0.131	0.00	1.00
ProgrammingLanguageMatch	0.06	0.236	0.00	1.00
AudienceMatch	0.13	0.322	0.00	1.00
OperatingSystemMatch	0.08	0.267	0.00	1.00
ProjectAge	3.25	0.514	2.40	4.67
PreexistingCode	0.01	0.089	0.00	1.00

Table 2.5 presents the bootstrapped means of the pairwise correlations between variables. Most variables are weakly correlated with correlations below 0.5. The highest correlation among the independent variables is between *MemberTieAmount* and *MemberProjects* ($\rho = 0.77$, $p < 0.001$). There also seems to be moderate correlation between *MemberProjects* and *MemberActiveTime* ($\rho = 0.69$, $p < 0.001$) as well as between *InitiatorProjects* and *InitiatorActiveTime* ($\rho = 0.67$, $p < 0.001$). We conducted diagnostic checks for multicollinearity by obtaining the bootstrap means of the tolerances (Greene 2000) and the variance inflation factors (Greene 2000, Mansfield and Helms 1982) of all predictor variables. The resulting tolerances were all above 0.20 and the highest variance inflation factor value was 3.60, which is below the commonly used cut-off value of 10 (Neter et al. 1996), suggesting that the potential bias in parameter estimates due to multicollinearity is not problematic.

Table 2.5 Inter-Correlations

	Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
(1)	*Join*												
(2)	*InitiatorTie*$_{Outcome}$	-0.02*											
(3)	*InitiatorTie*$_{Strength}$	0.06***	0.22***										
(4)	*MemberTie*$_{Outcome}$	0.00	0.28***	0.07**									
(5)	*MemberTie*$_{Strength}$	0.07***	0.19***	0.23***	0.26***								
(6)	*InitiatorTieAmount*	0.09***	0.06**	0.08***	0.10***	0.13***							
(7)	*MemberTieAmount*	0.07***	0.09**	0.10***	0.20***	0.23***	0.30***						
(8)	*InitiatorActiveTime*	0.03***	0.04**	0.03***	0.08***	0.06***	0.57***	0.13***					
(9)	*MemberActiveTime*	0.46***	0.09***	0.08***	0.18***	0.20***	0.25***	0.63***	0.11***				
(10)	*InitiatorProjects*	0.03***	0.01	0.09***	0.04***	0.05***	0.62***	0.12***	0.67***	0.07***			
(11)	*MemberProjects*	0.25***	0.08**	0.18***	0.16***	0.24***	0.20***	0.77***	0.09***	0.69***	0.10***		
(12)	*DomainPopularity*	0.05***	0.00	-0.01	0.01	0.01	0.09***	0.06***	0.05***	0.09***	0.11***	0.05***	
(13)	*ActivityPercentile*	-0.05***	0.02	-0.01	0.04***	0.03***	-0.04***	0.01	0.01	0.00	0.00	0.02	0.23***
(14)	*DescriptionLength*	-0.01**	0.02***	0.01**	-0.01	0.00	-0.01	-0.03***	0.00	-0.01	0.01	-0.02*	0.15***
(15)	*AcceptDonation*	0.03***	0.01*	0.00	-0.01	-0.01	-0.04***	0.03***	-0.01	0.03***	-0.02	0.02***	0.12***
(16)	*CodeReleased*	0.08***	0.01	0.00	0.00	0.00	-0.02*	0.04***	0.01	0.04***	0.00	0.04***	0.05***
(17)	*TeamSize*	0.66***	-0.01	0.04***	0.03	0.07***	0.35***	0.53***	0.10***	0.64***	0.10***	0.35***	0.10***
(18)	*DeveloperActiveTime*	-0.27***	0.05***	0.01	0.09***	0.06***	-0.02**	0.01	0.02	-0.02*	0.01	0.03***	-0.01
(19)	*DomainMatch*	-0.01	0.02	0.02	0.05**	0.06***	0.01	0.08***	0.01	0.06***	0.03**	0.08***	0.21***
(20)	*ProgrammingLanguageM*	-0.03***	0.03	0.03***	0.05**	0.05***	0.04***	0.09***	0.04***	0.07***	0.04***	0.09***	0.21***
(21)	*AudienceMatch*	-0.11***	0.01	0.01	0.01	0.01	0.01	0.01	0.02*	0.01	0.05***	0.03***	0.35***
(22)	*OperatingSystemMatch*	-0.07***	-0.01	0.01	0.03*	0.03***	0.00	0.04***	0.01	0.04***	0.03**	0.05***	0.26***
(23)	*ProjectAge*	-0.16***	-0.01	0.01	0.02	0.01	-0.04***	0.05***	-0.02	-0.02*	0.00	0.01	0.10***
(24)	*PreexistingCode*	0.02***	0.00	-0.01	0.00	-0.01	-0.03***	0.02**	-0.01	0.05***	-0.02	0.04***	0.04***
		(13)	**(14)**	**(15)**	**(16)**	**(17)**	**(18)**	**(19)**	**(20)**	**(21)**	**(22)**	**(23)**	
(14)	*DescriptionLength*	0.07***											
(15)	*AcceptDonation*	0.08***	0.06***										
(16)	*CodeReleased*	0.07***	0.02**	0.04***									
(17)	*TeamSize*	-0.05***	-0.04***	0.00	0.03***								
(18)	*DeveloperActiveTime*	0.02*	0.00	-0.01	-0.01	-0.17***							
(19)	*DomainMatch*	0.05***	0.02**	0.04***	0.01	0.02**	0.05***						
(20)	*ProgrammingLanguageM*	0.09***	0.03***	0.03***	0.01	0.01	0.08***	0.22***					
(21)	*AudienceMatch*	0.15***	0.08***	0.06***	0.02*	-0.05***	0.16***	0.22***	0.28***				
(22)	*OperatingSystemMatch*	0.10***	0.04***	0.05***	0.01	-0.02**	0.13***	0.18***	0.34***	0.33***			
(23)	*ProjectAge*	0.27***	0.01	0.02*	-0.06***	-0.05***	0.04***	0.04***	0.07***	0.09***	0.07***		
(24)	*PreexistingCode*	0.04***	0.03***	0.04***	0.01	0.01**	0.00	0.00	0.00	0.00	0.01	0.00	

Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$

2.4.2. Tests of Hypotheses Predicting Developer Joining Events

As discussed in previous sections, we distinguished between the initiator and non-initiator members who have already joined the project in testing the hypotheses. To test developer sensitivity to cohesion cues, that is their reliance on direct prior collaboration experiences with current project members when joining new projects, we investigate the coefficients of the parameters related to collaborative tie strength (H1) ($InitiatorTie_{Strength}$ and $MemberTie_{Strength}$) and the actual collaboration outcome (H2) ($InitiatorTie_{Outcome}$ and $MemberTie_{Outcome}$) of developers' past collaborative experiences. To test developer sensitivity to status cues, that is their preference for collaborating with developers who have demonstrated success in past collaborations (H3), we investigate the impact of the amount of ties to other developers forged through previous collaborations (*InitiatorTieAmount* and *MemberTieAmount*) as well as the previous experience of the project members on other SourceForge.net projects (*InitiatorActiveTime*, *MemberActiveTime*, *InitiatorProjects*, and *MemberProjects*). The results of the logistic regression are presented in Table 2.6.

Table 2.6 Logistic Regression Results

Variable	Parameter Estimate	95% Confidence Interval	Odds Ratio
Constant	-8.119***	(-10.442, -5.655)	
$InitiatorTie_{Outcome}$	-0.027	(-2.289, 2.025)	0.973
$InitiatorTie_{Strength}$	0.708**	(0.164, 1.972)	2.030
$MemberTie_{Outcome}$	0.985	(-2.585, 5.802)	2.678
$MemberTie_{Strength}$	0.367	(-1.395, 3.289)	1.443
InitiatorTieAmount	-0.190	(-0.580, 0.206)	0.827
MemberTieAmount	-0.384**	(-0.742, -0.069)	0.684
InitiatorActiveTime	-0.014	(-0.042, 0.011)	0.986
MemberActiveTime	0.027***	(0.011, 0.043)	1.027
InitiatorProjects	-0.009	(-0.504, 0.410)	0.991
MemberProjects	0.678**	(0.036, 1.363)	1.970
DomainPopularity[a]	0.569	(-1.565, 2.745)	1.766
$ActivityPercentile_{Low}$[b]	-0.974***	(-2.227, -0.176)	0.379
$ActivityPercentile_{Mid}$[b]	-0.099	(-0.462, 0.289)	0.906
DescriptionLength	0.091	(-0.250, 0.405)	1.095
AcceptDonation	0.655**	(0.166, 1.145)	1.925
CodeReleased	1.280**	(0.193, 2.077)	3.597
TeamSize	2.550***	(2.167, 3.089)	12.807
DeveloperActiveTime	-0.042***	(-0.055, -0.031)	0.961
DomainMatch	-0.093	(-2.272, 1.441)	0.911
ProgrammingLanguageMatch	-0.142	(-1.263, 0.639)	0.868
AudienceMatch	-1.077***	(-1.833, -0.510)	0.343
OperatingSystemMatch	-0.774**	(-1.690, -0.03)	0.463
ProjectAge	-1.782***	(-2.316, -1.253)	0.169
PreexistingCode	0.093	(-1.330, 1.333)	1.097

Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$

Notes: [(a)] It is possible that *DomainPopularity* does not fully capture domain-specific effects. Table B.6 presents the domain distribution of sample projects and projects that had attracted additional developers. It seems that the percentage of projects with joiners (at least for those software domains with a sufficient sample size; e.g., N > 50) is fairly similar across domains. We further checked the robustness of our findings by including dummy variables representing whether a project belongs to the top seven domains from Table 2.3. The results are presented in Table B.7. None of these variables has a significant effect on whether developers joined a project.

[(b)] Since *ActivityPercentile* is a categorical variable (low, mid or high), we report the results using two dummy variables where "high" is the base case for comparison.

$InitiatorTie_{Strength}$ is positive and significant ($\bar{\beta} = 0.708$, $p < 0.05$) indicating that the strength of a collaborative tie between the project initiator and the developer has a positive influence on the developer's decision to join the project, whereas $MemberTie_{Strength}$ does not significantly influence the developer's joining behavior ($\bar{\beta} =$ 0.367, ns). In short, developers are more likely to join a project initiated by someone

with whom they have developed strong ties through repeated collaborations and joint administration of past projects. Developers however do not seem to be influenced by the strength of ties with non-initiator developers already in the project. In summary, H1 is supported: when a developer has a strong tie with the project initiator based on joint project administration and frequent collaborative experiences in the past he is more likely to join the project. H2 however was not supported. Neither $InitiatorTie_{Outcome}$ ($\bar{\beta}$ = -0.027, ns) nor $MemberTie_{Outcome}$ ($\bar{\beta}$ = 0.985, ns) affects the likelihood that a developer joins the new project.[17] Developers were not influenced by whether or not their past collaboration projects with either the initiator or other members of the new project were successful in terms of the project outcomes.

H3 posited that project member status based on past collaborative experiences measured as the number of collaborative ties to other developers in the SourceForge.net developer network and the extent of project-related experience (in terms of time and number of projects) would positively affect a developer's likelihood of joining a project. There was no significant effect of the initiator's status operationalized as the number of collaborative ties on the likelihood that a developer would join the project (*InitiatorTieAmount:* $\bar{\beta}$ = -0.19, ns). Furthermore, the initiator's status operationalized based on past project experience did not have the expected significant impact (*InitiatorActiveTime*: $\bar{\beta}$ = -0.014, ns; *InitiatorProjects*: $\bar{\beta}$ = -0.009, ns). Contrary to what

[17] Because our survey of the administrators of our project sample indicated that some proportion of project joining events result from pre-existing collaborations that started outside of SourceForge.net and migrate with the project to SF.net, we conducted further robustness analyses. Specifically, we tested a model including interaction terms of *ProjectAge* with our collaborative tie strength and outcome variables. Significant interaction effects, i.e., a difference in the effect of prior collaborative ties for early joiners (i.e., potential migrators) compared to late joiners, would be indicative of project migration driving our observed results. As shown in Table B.8, none of the interaction terms of *ProjectAge* with the collaborative tie strength and outcome variables (H1, H2) was significant. Moreover, the inclusion of the interaction terms did not reduce the main effect of *InitiatorTieStrength*. Hence, while project migration occurs to some extent, it is not prevalent, and our findings remain robust in the presence of this phenomenon for a fraction of our study sample.

we predicted, the non-initiators' status operationalized through the number of collaborative ties has a significant but negative impact on developer joining (*MemberTieAmount*: $\bar{\beta}$ = -0.384, $p < 0.05$). A unit increase in *MemberTieAmount* decreases the likelihood of developer joining by 31.6%. Non-initiators' status operationalized as past project experience has a significant positive impact on developers' joining behavior (*MemberActiveTime*: $\bar{\beta}$ = 0.027, $p < 0.01$; *MemberProjects*: $\bar{\beta}$ = 0.678, $p < 0.05$). Thus, H3 is partially supported when status of non-initiator members is operationalized based on the number of past project collaborations and the actual time spent in these endeavors, but is not supported when operationalized as the number of developers they have developed ties with through these projects.

We further examined the impact of additional project and developer attributes on the likelihood that a developer joins a project in order to arrive at a deeper understanding of how they influence developer choice of new OSSD project teams. We focused on factors that were found to have a significant impact among those attributes that we considered. We examined visible project attributes that may signal a project's potential to result in positive collaborative process and outcome – namely that related to code availability and project popularity. The significant and positive parameter estimate for *CodeReleased* ($\bar{\beta}$ = 1.280, $p < 0.05$) suggests developers are more likely to join a project that has released some software code. The significant positive effect of *TeamSize* ($\bar{\beta}$ = 2.550, $p < 0.01$) indicates that a developer prefers joining those newly-initiated projects that have already been successful at attracting additional developers. A unit increase in project team size increases the likelihood of developer joining by almost 12 times. Other project attributes related to the activity levels and available resources of a project were also found to affect the likelihood of a developer joining. Developers were more likely to join projects earlier (*ProjectAge*: $\bar{\beta}$ = -1.782, $p < 0.01$) and active projects ($ActivityPercentile_{Low}$: $\bar{\beta}$ = -0.974, $p < 0.01$). Furthermore, a project set up to accept

donations from users is more likely to attract additional developers into the team (*AcceptDonation*: $\bar{\beta} = 0.655$, $p < 0.05$).

We also examined how developers' prior experiences may influence their decisions to join new OSSD project teams. *DeveloperActiveTime* is negatively associated with developers' joining behavior ($\bar{\beta} = -0.042$, $p < 0.01$), suggesting that less experienced developers are more likely to join a new project. The negative estimates for *AudienceMatch* ($\bar{\beta} = -1.077$, $p < 0.01$) and *OperatingSystemMatch* ($\bar{\beta} = -0.774$, $p < 0.05$) indicate that developers tend to participate in projects designed for intended audience and operating systems that are different from those of their previous projects.

In summary, the results suggest that OSS developers prefer joining a project whose initiator has developed a strong tie with them through repeated collaborations and shared administration of past projects. They are also more likely to join a project whose non-initiators have more project experience. In addition to ties with the initiator and experience of the non-initiators, other project characteristics such as team size, code release, project age, and whether projects accept donations also seem to play a role in influencing developer decisions to join an OSSD project.

2.5. Discussion

2.5.1. Summary of Results

In this study, we examined the impact of the online collaboration network on how individuals embedded in the network self-organize into newly-formed teams in the OSSD context. Specifically, we empirically investigated whether past ties with and status of existing project members influence developers' decisions to join the project. Overall, our results supported our main thesis that the network of collaborative ties affects developer decisions to join new projects. We found that developers rely on both their past

collaboration experiences with project initiators and other members' status cues when choosing which project to join. We conclude by offering some general observations regarding developer choice of new OSSD projects based on the collective results.

In OSSD projects, the most critical resource is its members. We find that developers rely on a variety of cues when evaluating the attractiveness of a project based on their potential collaborators, and that the cues they focus on may be different depending on the role of the potential collaborator (i.e., project leader/initiator vs. other developer). In the case of the role of project leader, developers are prone to participate in new OSSD projects that are initiated by developers they have interacted intensely with in the past regardless of how successful the previous project has been (H1, H2). A developer is more likely to join a project when he has developed a strong collaborative tie with the project initiator through repeated collaborations and shared administrative roles in the past, demonstrating the vital role of initiators in attracting new developers. There may be various reasons why developers repeat collaborations with project initiators they have worked with in the past. First, software development is not only a production process but also a social process that relies heavily on interpersonal communication and coordination (Curtis et al. 1988, Robey and Newman 1996, Sawyer et al. 1997, Sawyer and Guinan 1998). Moreover, in the OSSD context, the difficulty inherent in the social process becomes even greater than in traditional software projects since members of an OSSD project are typically from geographically dispersed locations (Crowston and Scozzi 2002) and have limited (if any) face-to-face interactions (Scacchi 2002). From the developer's perspective, a large amount of uncertainty exists with regard to how smooth and how efficient it will be to interact with the existing members of a project. Developers who have built strong ties with the project initiator through past interactions and joint project administration experiences may have developed a sense of trust and norms of behavior with the project initiator that can improve team effectiveness (Adler and Kwon 2002). Their familiarity with and trust in the project leader can mitigate the risks and reduce the uncertainties in the collaboration process. Second, the positive impact of strong ties with the initiator may also be caused by a sense of belonging to the project group that

developers participated in with the initiator. A survey of Linux developers found that active developers identified strongly not only with the Linux community broadly but also with the specific subsystem team (Hertel et al. 2003). This sense of group identity will lead OSS developers to evaluate the projects initiated by those they perceive to be in-group members more favorably than projects initiated by out-group members (Hogg and Abrams 1988). However, whether or not the past collaborations that developers engaged in with the project initiator were successful in outcome did not seem to influence developers' joining behaviors (H2). If developers were motivated mainly by the use-value need in participating in OSSD, i.e., successfully developing software that addresses their personal needs, then we would expect that the positive outcome associated with prior collaborative ties with the initiator would impact project joining decisions. The fact that this relationship is insignificant may indicate that for OSSD participants, the eventual outcome of the OSSD project might be less important than the fun, enjoyment and learning benefits they derive through the process of jointly developing code and participating in OSSD projects (Lakhani and Wolf 2005). An alternative explanation for the apparent lack of impact of prior collaboration success may be that our operational measures do not accurately reflect actual project success and the outcome dimensions that OSS developers value. Because most OSS projects release code early and often, our outcome measure, which places heavy emphasis on code released, may not be an indicator of success, but rather of ongoing development processes. Developers were not influenced by status cues of project initiators when selecting projects to join (H3). In other words, developers relied on their own personal experiences and joined projects initiated by developers they knew and had built strong collaborative ties with, but did not rely on the implicit recommendation of others in the OSSD network based on the initiator's prior project collaboration experiences.

A different pattern emerges in the case of developers' choice of non-initiator collaborators in new OSSD projects. Collaborative tie strength with other members of the project did not have a significant effect on whether a developer joined a new project (H1). This may be because leaders play a more clearly discernible critical role in the

OSS development process relative to other developers. The leader, who in most cases is also the project initiator, guides the overall direction of the project and influences how the project will progress in the future (Lerner and Tirole 2002, Nakakoji et al. 2002, Xu et al. 2005). Although all developers in a project need to coordinate with the project leader, they do not necessarily need to achieve close coordination with all other project members, depending on project and code structure. Hence, when choosing a new project to join, developers may exhibit a wider variation in their sensitivity toward whether or not they share a strong collaborative tie with other members of a project based on how critical they perceive other members' roles in project communication and coordination to be. In addition, initiators may be perceived by some developers as being more committed to the project than other members of the project who may lose their interest in the project after joining it. Therefore, while prior collaborations with initiators may drive the choice of project to join for most developers, due to possible differences in developers' perceived roles of other members within the project, developers do not appear to uniformly rely on their prior collaboration experiences with non-initiator members to make their joining decisions. In other words, prior collaborations with other project members may be an important consideration in choice of new OSSD projects only when the focal developer expects to engage in close coordination with these members. As was the case with prior collaboration outcomes with initiators, whether or not a prior collaboration with non-initiator members of a project resulted in a positive outcome also did not seem to significantly affect developers' joining decisions (H2). A developer may rely on status cues of the non-initiator members when deciding whether or not to join a particular project (H3). In particular, developers are more likely to join a project whose non-initiator members have more project experience based on both actual time invested in project participation and the number of projects. Developers may not only perceive non-initiator members with accumulated OSSD project experience to be more competent and capable but also rely on their experience to judge the attractiveness of the project and its likelihood of success. However, developers were less likely to join when non-initiator members had a greater number of past collaborative ties. One possible explanation for this may be related to our operationalization of the status cues. While all three indicators

of status are based on involvement in past projects, the number of past collaborative ties may also dilute the perceived experience level of the project members. In other words, insofar as the competence and skills of the non-initiator members based on their previous project experiences have been captured through the measures of their actual project experience, the number of past collaborative ties may be measuring the significance of the contributions made to the projects they have participated in. More collaborative ties could be construed as an indicator of having played a less significant role in the previous collaborations; hence developers may perceive non-initiator members with greater number of ties given the same project experience as less competent and capable.

In summary, the results indicate that developers may use different cues in evaluating potential collaborators in new OSSD projects, depending on their perceived role within the projects. Developers join projects that are initiated by developers they have established strong collaborative ties with, but rely on other status cues when evaluating other project members.

We offered one possible explanation for the significant negative impact of the number of collaborative ties that non-initiator members of a project have on the likelihood of joining that project above. An alternative explanation is offered based on expectancy value models (Karau and Williams 2000). While from the project perspective a greater number of collaborative ties may increase the likelihood of technical and commercial success (Grewal et al. 2006), from the perspective of a potential developer contributing to the project, the presence of other skilled developers with greater access to resources might reduce his opportunities to contribute to and gain recognition from the project. According to expectancy value models the motivation to contribute to a group will be positively influenced by the developers' expectation that their contributions are unique and hence valued (Karau and Williams 2000). This explanation appears somewhat plausible when considered together with the significant negative effect of project age (*ProjectAge)* on the likelihood that a developer joins the project. Developers were less likely to join projects when projects were already well under way. This may be because

the earlier they join, the more impact their contributions may have and hence the more visible their contributions to the project may become as the project progresses and keeps attracting more developers in the future (Lerner and Tirole 2002). Newer projects may also have lower technical barriers to joining as the software code and architecture are relatively easier to understand when they are still in their early stages. As the software grows more complex, fewer developers are able to fully understand the architecture and effectively contribute code because of a high learning curve (Kohanski 1998, von Krogh et al. 2003). Hence, developers may have lower expectations of being able to make meaningful contributions that have an impact as the project age increases and thus be less likely to join the project.

Developers also are more likely to join projects they are able to judge as having a higher probability of success. We found that a project with more developers is more likely to attract additional developers than a project with fewer developers, a finding consistent with the "rich get richer" mechanism proposed by Barabási and Albert (1999) for many self-organizing networks.[18] Developers are more likely to join a project that exhibits higher than average development activity levels and has released some initial software code that may outline the functionalities envisioned by the project initiator and demonstrates the potential merits of the project (Lerner and Tirole 2002). This is consistent with the argument that some minimal code needs to be assembled in order for the project to receive reaction from the OSSD community (Raymond 2001). The existing activity and initial code may facilitate developers' judgment regarding a project's probability of success. Additionally, projects accepting donations from users are more

---

[18] On the surface this may appear to be inconsistent with the argument that developers wish to increase the chance that their contributions will be meaningful and visible and hence are less likely to join projects where the non-initiator members have more collaborative ties within the OSSD network. However, having more developers on the team who have less experience and resources would not necessarily result in lower chances for one to make meaningful contributions. In short, it is not the number of developers per se but the potential for their differential ability to access resources in the OSSD network, based on their collaborative ties, that determines whether one's chance for making an impact on the project outcome is in fact reduced.

likely to attract developers to join as such donations may improve a project's chances by providing needed external resources.

Finally, developers also seek to increase the variety of their OSSD experience and acquire new skills. They are less likely to join new projects if these are developing software that is based on an operating system and intended for an audience for which they have previously developed software as members of a different project. However, with increased experience developers may no longer need to participate in new projects for the purpose of learning. Developers with longer participation history are less likely to join new projects, as they may have gained some expertise and experience in OSSD and may consequently prefer to continue with their current projects or initiate their own projects rather than joining others' new projects.

2.5.2. Contribution

The study has both theoretical and practical implications. Theoretically, this study contributes to the growing stream of studies examining why and how the structure of social relations impact individual behavior in a self-organizing network by employing a social network perspective to examine the interactions among interdependent OSS developers embedded in an online collaborative network. The findings from our study also contribute to an enriched understanding of the ecology of OSS projects by providing one perspective on attributes of new projects and their members that attract developers. Although a growing body of research has explored the motivations of OSS developers, little research to date has attempted to understand how developers choose which of the myriad of possible projects available to contribute to. Additionally, our findings also provides empirical evidence for one mechanism that could explain the reason behind the preferential attachment processes leading to the formation of scale-free networks prevalent in digital and social networks. OSS developers choose to continue collaborating with project leaders with whom they have developed strong collaborative ties. This finding is consistent with research in the formation of founding teams for new

firms that finds a tendency for entrepreneurs to prefer people with whom they already have strong ties (Ruef et al. 2003).

Although our main focus here was to investigate developers' decisions to join newly-created projects and the results may not apply to developers joining established projects, our findings also have important practical implications that may help new projects obtain early momentum. Developers interested in launching new OSS projects may be well-advised to first establish strong collaborative ties with other developers in the OSSD network, and thereby increase the size of the development team through these collaborative ties. In addition, releasing some early working code of the project would increase the probability that the project would attract additional developers.

2.5.3. Limitation and Future Research

We end our discussion by noting some limitations and additional directions for future research. First, in this study we focused on the factors that affect a developer's choice of new projects to join and treated the new project team's existing members and initiator as passive entities. The significant impact of collaborative tie strength is also consistent with project initiators seeking out developers with whom they have worked in the past to help them in their new project. While this is plausible, we found in a survey of our sample project administrators that this rarely occurred, and hence the one-sided process presented in this study was more appropriate in our context of archival field study design. Future research employing other methods should extend the model of new project joining to account for the active role that existing project members and the initiator play in new developers joining a project. A second limitation is that we only examined joining behavior within the initial months after project initiation. The joining behavior may differ depending on stages of project development. While controlling for development stage would offer richer theoretical insights, practically, many newly registered projects do not define their development stages explicitly, which limits our ability to incorporate this factor into the analysis. An important extension of this essay would be to study the effect of developer joining behavior on the network structural characteristics within

project team as well as in the OSS developer network. Third, our data only include information available from SourceForge.net. Even though SourceForge.net is currently the largest repository of open source software, constraining the data collection to one site limits our ability to capture the interactions among developers at other OSS hosting sites. For example, even if two developers have no prior collaborations at SourceForge.net, they may have already formed relations with each other through collaborations on other projects hosted elsewhere. But since we cannot observe their past collaborations and code them as having no ties, our estimates of the impact of past collaborative ties would be underestimated and the test of hypothesis conservative. Finally, an interesting avenue of future research would be to investigate how early team formation influences a project's subsequent performance and sustainability.

Overall, this study contributes to a richer understanding of the assembly mechanisms for self-organizing teams embedded in an online collaborative network. It is one of the few studies that examined how social networks influence the joining activity of OSS developers. The findings in the research also offer some guidelines to OSSD project managers that can help them attract developers into the teams and gain early momentum.

CHAPTER 3. SHOULD I STAY OR SHOULD I GO: AN EMPIRICAL INVESTIGATION OF MEMBERS' CONTINUED PARTICIPATION IN OPEN SOURCE PROJECT COMMUNITIES

3.1. Introduction

User innovation refers to products or product modifications developed by end users and consumers instead of commercial manufacturers (von Hippel 2005). It spans from industrial products such as library information systems (Morrison et al. 2000) to consumer products such as sports equipment (Shah 2000). Innovation by users is not a new phenomenon but it has been growing rapidly in the past decade due to developments in computer and communication technologies. Innovative users sharing common interest often form innovation communities, which are characterized by "voluntary participation, the relatively free flow of information, and far less hierarchical control and coordination than seen in firms" (Shah 2005, p. 343) and consist of "individuals or firms interconnected by information transfer links which may involve face-to-face, electronic, or other communication" (von Hippel 2005, p. 96). Examples of communities formed by innovative users can be found in fields such as video games (Jeppesen and Molin 2003), sports equipments (Shah 2005), and open source software (von Hippel 2007).

This study focuses on members' repeated participation behavior in online innovation communities and its impact on communities' sustainability. In these communities innovation development, production, distribution, and consumption is distributed horizontally across many users and innovators (von Hippel 2007) and innovation cannot succeed without active involvement of members. Furthermore, instead of innovating in isolation, members in these communities do receive assistance and advice from other members (Franke and Shah 2003), which requires effective transfer of information related to the innovation itself. Due to the high cost incurred in such information transfer

(von Hippel 1994), as the innovation process continues and the product grows more complex, learning cost for prospective community members becomes much higher. Hence, retaining members who already possess significant amount of sticky information becomes even more beneficial to providing assistance to other members and to sustaining the success of communities.

Extensive research has been conducted to examine members' participation and dynamics in online communities. Some studies focus on understanding members' behavior at a micro level. Researchers investigate intrinsic and extrinsic motives for participation in online communities such as generalized reciprocity, enjoyment (e.g. Wasko and Faraj 2000), and community citizenship (e.g. Ardichvili et al. 2003). Koh et al. (2007) distinguish viewing activity and posting activity by community members and find that the former is influenced by the perceived usefulness of the community whereas the latter is associated with level of members' offline interaction and quality of IT infrastructure. Other studies have tried to understand online communities at a macro level by examining the network aspect of online communities. In order for online communities to be sustainable they need to be able to create benefit for their members mainly through communication activities among members. For example, Butler (2001) examines the communication activities in email-based Internet listservs and finds that community membership size is associated with both positive and negative effect on the development of community.

Online innovation communities are similar to other online communities in that they are formed by "group of people with common interests and practices that communicate regularly and for some duration in an organized way over the Internet through a common location or mechanism" (Ridings et al. 2002, p. 273). However, unlike most online communities that can be formed around a number of shared interests and purposes (Porter 2004) or "discourse focuses" (Jones and Rafaeli 2000), online innovation communities are formed to facilitate a production process. In other words, in many online communities that have been studied by researchers, members are embedded in networks

and they derive benefits from communication activities that take place in networks. In contrast, in product-centered innovation communities, besides communication activities between individuals and networks, another aspect of these communities – involvement of individuals in the innovation process – also plays a role in shaping individuals' behavior. Hence, factors driving members' participation behavior in innovation communities may not be limited to their individual characteristics and their interactions with others in the network; findings from other online communities regarding individual participation patterns may not be applicable to innovation communities.

In this essay we examine members' continued participation behavior in online innovation communities by focusing on the benefit creation aspect of communities and, more importantly, the involvement of individuals in the innovation process. On one hand, community members obtain benefits such as informational benefits and recognition from communication activities with others. On the other hand, community members are also involved in the production process at various levels and they obtain benefits from such involvements such as sense of accomplishment and learning effects besides the benefits they receive from interactions with other members. The research questions we attempt to answer in this essay are:

- How do communication activities with other members in an innovation community influence an individual's continued participation?
- How does an individual's involvement in the innovation process impact his continued participation?

We choose open source software development (OSSD) as the research context because of theoretical and practical considerations. OSSD has been frequently referred to as an example of user innovation networks or community-based innovations (e.g., Benkler 2006, Franke and Shah 2003, von Hippel 2007). Practically, although open source software adoption has been on the rise and individuals especially software developers are participating in OSSD on an unprecedented scale, there is still much to be learned about the dynamic process and organization of OSSD. Despite the success stories of Apache

web server, PhP, and other well-known open source software, many projects were abandoned due to lack of user interest among other factors. How to improve the sustainability of OSSD projects is of practical concern to many open source entrepreneurs and commercial software companies interested in investing in OSSD. Some studies have attempted to examine the performance differences of OSS projects 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. 2006). We extend this stream of research by focusing on the community aspect of OSS projects. OSSD has been considered as an example of commons-based peer production model, a new model of economic production that is different from both markets and firms where a large number of people self-select into projects without traditional hierarchical organization or financial compensation (Benkler 2006). One of the key advantages of developing software in the open source model is its ability to actively involve other users or developers and obtain timely feedback and/or contributions from them in the development process. Success of peer-based production cannot be achieved without active participation of large groups of individuals. To the best of our knowledge how to attract and retain individuals in OSS projects is yet to be understood and even less is known about how individuals interact with each other in peer-based production mode.

The essay is organized as follows. Section 3.2 introduces the theoretical background and section 3.3 develops our research model and hypotheses. Research methodology is explained in detail in section 3.4 and section 3.5 presents our results. We conclude the essay in section 3.6.

3.2. Theoretical Background

Our research touches upon three streams of research in IS: IS continuance, benefit provision in online communities, and open source software development (OSSD) communities. We first review the literature related to IS adoption continuance to relate continued participation to continuance in IS use. We then review the literature related to member participation in online communities to identify the relationship among network

externality, community benefit provision, and member participation. Next we review the existing literature on OSSD communities to identify the distinctions between these communities and other online communities that have been examined before.

3.2.1. IS Continuance

Expectation confirmation theory (ECT) (Oliver 1980) has been widely used to study consumer satisfaction and post-purchase behavior. The four main constructs in the model include expectation, perceived performance, confirmation and satisfaction. ECT posits that consumers' intention to repurchase a product or service is determined by their post-purchase satisfaction, which is determined by initial expectation and confirmation of expectation. Confirmation of expectations is formed after consumers assess the perceived performance against their initial expectation of the product or service (Oliver 1980). ECT has been applied to examine consumer repurchase intention for a wide range of products and services such as camcorder (Spreng et al. 1996) and business professional services (Patterson et al. 1997).

Recognizing the similarity between consumers' repurchase decision and IS users' continuance decision, Bhattacherjee (2001) is one of the first studies to adapt ECT and propose a post-acceptance model to examine continuance of information systems (IS) adoption. The study finds that users' decision to continue using IS is determined by their satisfaction with initial IS use and perceived usefulness of continued IS use. User satisfaction is primarily influenced by users' confirmation of expectations and secondarily by perceived usefulness of IS. Since then IS researchers have adapted ECT to examine IS users' continuance in a variety of contexts such as usage of web portals (Lin et al. 2005), e-negotiation systems (Doong and Lai 2008), and mobile Internet (Hong et al. 2006).

Most of the prior studies on IS continuance examine consumers' continued adoption of products or services that are fairly static over time. When the product or service in question is constantly evolving, the existing ECT framework may not be fully applicable

because users need to re-evaluate their confirmation level and adjust their expectation after every use based on the current characteristics of the product or service. In addition, in the IS continuance literature, individuals are often treated as stand-alone entities whose continuance decisions mainly depend on their own perceptions of and experiences with IT products or services. In contrast to the IT products/services examined in most IS continuance studies, online communities are more dynamic in nature and evolve on a constant basis with addition and attrition of members and changes in individual behavior. Individuals are no longer isolated entities; they influence and at the same time are influenced by behavior of other community members. To the best of our knowledge few scholars have looked at IS continuance in a dynamic, networked environment with the exception of Tiwana and Bush (2005) who incorporate the network aspect of post-adoption investments to explain continuance intention. They borrow from sunk cost theory and develop an expertise-sharing network continuance model that identifies reputation, relational capital, and personalization as the key factors that impact network continuance at the individual level. However, formulated from the perspective of individual perceptions, the model does not explicitly incorporate the interdependence among members or the influence of other members on individual continuance intention. In addition, the model does not capture the dynamic interactions among network participants over a period of time.

3.2.2. Member Participation and Network Externality in Online Communities

Research in social psychology has identified various benefits that social groups and communities can provide for their members. Traditional communities satisfy individuals' need for affiliation with others (McClelland 1985), help individuals obtain information and achieve goals (Watson and Johnson 1972), and offer social support (Wellman and Wortley 1990). Similar benefits also motivate participation in online communities. Prior online community research has attempted to identify three types of factors motivating member participation. The first type includes extrinsic motivational factors include informational benefit (e.g., Butler et al. 2007, Ridings and Gefen 2004, Wasko and Faraj 2000), relational benefit such as emotional support (Preece 1999,

Wellman and Gulia 1999) and sense of belonging (Ridings and Gefen 2004, Wellman et al. 1996) obtained from online social interactions, as well as reputational benefit such as recognition and visibility (Butler et al. 2007, Hars and Ou 2002, Wasko and Faraj 2005). The second type includes intrinsic motivations include altruism and enjoyment (Butler et al. 2007, Lerner and Tirole 2002, Wasko and Faraj 2000, Wasko and Faraj 2005). The third type consists of community-related contextual factors such as social identity, group norms, shared vision, structural social capital (Chiu et al. 2006, Dholakia et al. 2004, Wasko and Faraj 2005), community size, and communication activity (Butler 2001, Gu et al. 2007, Jones et al. 2004). This study extends the existing research of member participation in online communities by empirically testing the association between community-related contextual factors such as network effects, benefit provision to members, and members' continued participation in communities.

Furthermore, because most of empirical studies on member motivations tend to be cross-sectional with the exceptions of Butler (2001) and Jones et al. (2004), they offer limited insights on when and under what conditions community members continue or stop their participation. Hence, in order to identify the factors contributing to changes in members' participation behavior over time, we conduct a longitudinal analysis of members' continued participation in this study. Next, we will review the prior studies that focus on how network externalities affect benefit provision in online communities.

Many online communities such as newsgroups, discussion forums, and professional networks are established to facilitate information sharing among their members. Prior research has found that membership size and communication activity in online information-sharing communities have both positive and negative effects on the informational benefits obtained by their members (e.g. Butler 2001, Gu et al. 2007). In other words, these communities exhibit both positive and negative network externalities.

Positive network externalities exist when the benefits individuals obtain from the network is an increasing function of network size or the number of members in the network (e.g.,

Katz and Shapiro 1995, Kauffman et al. 2000). In the context of information-sharing online communities, the number of members participating in the communities represents the aggregate amount of potential information resource that their members may tap into. In a larger community individuals seeking information are more likely to find someone possessing that piece of information and willing to share it. Furthermore, the extent to which potential information resources can create actual value for community members depends on the communication activities in the community. A community with abundant resources may not produce sufficient informational benefits for its members unless people are willing to and able to share their knowledge and information.

However, network externality is not always positive. For example, although an increasing number of subscribers in a computer network may allow them to access more resources, an overloaded network will deteriorate the service quality and impose congestion cost on its subscribers (e.g., Liebowitz and Margolis 1994). In online information exchange communities, negative externalities are mainly caused by information overload, "the state of an individual (or system) in which not all communication inputs can be processed and utilized" (Jones et al. 2004). Information overload results from an individual's limited capability to effectively store and process information and communication messages. When participants are involved in extensive communication activities in online communities, the level of effort required to read and process all relevant messages may become prohibitively high for some of them, thus reduce their participation as well as the benefits they obtain from the community. In addition, communications in online discussion communities depend on members' voluntary participation. In other words, instead of being assigned by someone else to respond to a request, a member typically self-selects to do so. Consequently, an increasing level of communication activity in online communities may impose extra burden on their members to search for the messages that they are interested in responding to. Together, information overload and high search cost reduces the amount of benefit that individuals obtain from their participation in such networks after the network size reaches a threshold.

Butler (2001) is among the first studies to examine the dynamics in online communities. His proposed model of sustainable online structure consists of a feedback loop linking resource availability, benefit provision, and the change in the structure's membership. Membership size represents resource availability in the community and individuals benefit from the community by obtaining values through communication activities with other members. The findings based on community-level longitudinal data suggest that community size and communication activities have both positive and negative impacts on attracting and retaining members. The negative impact is due to higher information processing cost and decreased opportunities to participate, which lead to lower levels of contribution. The positive impact is caused by increased potential interactions with others. Jones et al. (2004) focus on the collective impact of individual information-overload coping strategies on the dynamics of online interaction spaces. They find that as the overloading increases individuals are more likely to respond to simpler messages, generate simpler responses, and to end active participation. Gu et al. (2007) further this line of work by incorporating the quality of messages as well as competitions among communities. Consistent with the findings in Butler (2001), their results suggest both positive and negative network externalities exist in online communities. Furthermore, quality of posted messages plays a key role in determining the network externalities. Different from Butler (2001) and Jones et al. (2004), Gu et al. (2007) adopt individual as the unit of analysis and analyze individual's valuation of different online communities.

Our work extends prior empirical work on network effects in information-sharing communities in two aspects. First, few studies with the exceptions of Joyce and Kraut (2006) and Wang (2007) have distinguished between first-time participants and returning participants and empirically examined factors driving members' repeated participation in online communities. Joyce and Kraut (2006) empirically test whether the responses that new members receive to their first posts influence their likelihood of posting again. Wang (2007) focuses on the impact of contextual factors such as current participation, prior participation, and availability of alternative communities on the individual's

continuance behavior in online newsgroups over time. Our research furthers this line of work by investigating members' repeated participation behavior in innovation communities instead of information-sharing communities, which was the focus in Joyce and Kraut (2006) and Wang (2007).

Secondly, information-sharing communities mainly involve interactions between an individual and the community whereas innovation communities involve interactions among the individual, the community, and the product. In the information-sharing community an individual derives informational benefit from communications with others. The value creation process involves two parties: the individual and the rest of the community. Resources available in the community cannot be converted to informational benefit without communication activity. In contrast, we focus on the innovation community, in which participants share a common goal and contribute to the product being innovated. The product is constantly evolving as a result of activities that take place in the community. The individual interacts with the community and more importantly participate in the innovation process. Hence, benefits obtained by the individual may come from his interactions with the community or his involvement in the innovation process or both. Therefore, the relationship between network externality and community benefit provision through communications may no longer be sufficient to explain the individual's participation in innovation communities.

In order to examine how individuals participate in the innovation process in OSSD communities, next we review the research on the organization of OSSD communities and members' involvement in the software development process.

3.2.3. Organization of Open Source Communities and Member Involvement

Open source software projects are "innovation development, production, distribution and consumption networks that are distributed horizontally across many software users" (von Hippel 2007, p. 293). OSSD projects are typically initiated by an individual or a group of individuals trying to find a solution to his or their specific needs (Raymond 2001). The

initiators and others who are interested in the project and whose joining requests are approved by the initiators generally become the project' owners or core members who take on responsibility for coding, debugging, and maintaining the software. The project members develop a preliminary version of the code outlining the basic functionalities and release it to the public. Individuals who are interested in using or improving the code may seek additional information and/or contribute information or code.

The organization of project communities based on the roles played by individuals has been investigated in the past few years (e.g., Crowston and Howison 2006, Moon and Sproull 2002, Nakakoji et al. 2002, Xu et al. 2005). Communities usually have an onion-shaped structure consisting of project members, code contributors, active users, and passive users. At the onion's center are the project members or core developers who are responsible for coding, debugging, and maintaining the source code. They are formally listed as members of the project and they typically have commit privileges to the source code repository. Around the core developers are code contributors or co-developers who write code but do not have commit privileges to the code repository. Having some understanding of the inner-working of the code, they create new and modified code and voluntarily contribute their code to project members for review. The code whose quality and functionalities are considered to be sufficient by project members is then incorporated into the existing code base (Crowston and Howison 2006, von Krogh et al. 2003). Around co-developers are a larger number of active users who contribute by testing new releases, submitting bug reports, suggesting additional features, and helping answering questions from passive users regarding setup, configuration, and build. The community also consists of passive users who use the code without contributing to its development. Some of them may post requests for information related to setup and configuration of the software. Overall, this stream of research distinguishes among OSSD participants based on their *level of involvement* in the software production process and their level of influence over the final product – the software – itself.

However, as Ducheneaut (2005) points out, this view of the community does not fully capture the dynamic nature of the open source software development process. It does not explain how the software production system is sustained over time. Neither does it illustrate how individuals gradually become involved and engaged in the production process (e.g., how passive users become active users, how active users become code contributors, and how code contributors become project members). Recognizing that individuals may behave differently and OSSD communities evolve over time, a few researchers have examined the dynamics in these communities. For example, von Krogh et al. (2003) study the strategies and processes by which external members join the existing project community of developers. They analyze the participation activities of individuals and find a high turnover in the community, which calls for further understanding of the joining and contributing behavior of newcomers. Furthermore, after identifying different types of activities based on email contents, they propose that participants behaving according to a certain joining script in terms of level and type of activity are more likely to be allowed access to the code repository than others who do not follow the joining script. Jensen and Scacchi (2007) analyze the role migration of developers from passive users toward more central roles in the software development process and identify the different paths of role migration that developers may take based on their observations of three large OSSD project organizations. Similar to these studies, our research also distinguishes among types of participants based on their level of involvement in the OSSD process. Rather than looking at how external members become accepted by the existing project developers or the role migration paths, we attempt to identify the factors that promote or hinder the repeated participation of community members. We are especially interested in the impact of individual-community interactions as well as the impact of individual involvement in the process on his continued participation in the community.

3.3. Research Model and Hypotheses

Drawing from the literature in IS continuance, online community dynamics, and OSS community organization, we argue that an individual's repeated participation in the

OSSD community is driven by both his interactions with the community and his involvement in the innovation process. On one hand, he is embedded in the community instead of being an isolated and independent entity. His subsequent participation to some extent is shaped by the benefit provided by the community, which are influenced by the resource availability in the community. On the other hand, the process that converts community resources to member benefits is not identical for all individuals. Instead, it is moderated by individuals' involvement in the innovation process (i.e. the level of influence they have over the final product). Our proposed research model is depicted in Figure 3.1.

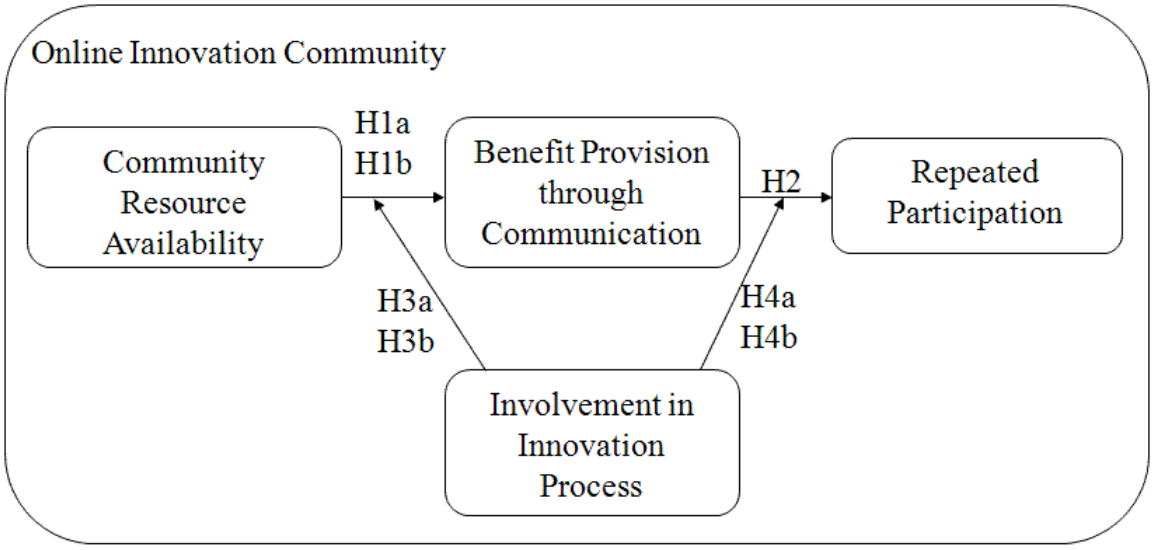

Figure 3.1 Research Model

3.3.1. Community Benefit Provision

3.3.1.1. Community Resource Availability and Benefit Provision

Online communities can be viewed as social structures embedded with resources such as knowledge, time, effort, which are typically provided by community members. In many cases the amount of resources available in communities is determined by their

membership size. For example, in online communities dedicated to information sharing, the aggregate amount of information and knowledge possessed by their members tends to be greater in larger communities. For innovation communities a larger pool of participants often represents more potential expertise, knowledge, time, and effort that they can tap into.

However, sufficient resources do not necessarily guarantee that community members will obtain benefits from communities. Members must be engaged in communication activities in order for communities to successfully convert resources into benefits for members (Butler 2001). The same holds true for innovation communities where contribution and participation are essential to their sustained development because contribution and participation have to happen through communications among members. The significance of communication activities is even greater for open source communities collaboratively creating knowledge-intensive software products.

In larger communities the resource pool provided by their members often is bigger and more diverse, so it is more likely that at least someone in communities has the capability and willingness to provide benefits by responding to an individual's participation. Hence, we hypothesize that:

> *H1a: The amount of community resources increases the likelihood that a participant will receive benefits from other community members.*

Nonetheless, due to negative network externality caused by information overload and high search cost, once a certain threshold of community resource level is reached, having additional resources in the community yields benefits at a decreasing rate. Furthermore, larger communities are more likely to suffer from free-riding problems because their members expect others to engage in communication activities and provide benefits. Hence, they tend to contribute less time and other resources (Petty et al. 1977). Thus the

marginal positive impact of community resources on the likelihood that a participant will receive benefits is reduced as more resources become available.

H1b: The marginal effect of community resources on the likelihood that a participant will receive benefits from other community members decreases with the total amount of resources.

In summary, consistent with existing research on network externalities in online communities, we hypothesize that resource availability in innovation communities increases the likelihood of benefit provision to their members at a decreasing rate due to positive and negative network externalities.

3.3.1.2. Community Benefit Provision and Members' Repeated Participation

We propose that, similar to consumers' repeated purchase of a product and IS users' continued adoption of an information system, members' repeated participation in innovation communities is also influenced by their satisfaction with prior participation. Their satisfaction with prior participation is determined by their expected benefits and the actual benefits provided by communities.

In online communities, participation means "generating messages, responding to messages, organizing discussion, and offering other online activities of interest to members" and participation by generating and consuming content is essential to the sustainability of online groups (Butler et al. 2007, p. 174). In the OSSD context, when an individual participates in the project community by posting a message, he often expects others in the community will read and consume his message and react to it in appropriate ways. For example, if he posts a request for information or technical support, he would expect to receive an answer from the community. If he reports a problem or to requests a new feature to be added to the software, he hopes that his participation will receive attention from the community and, if possible, some members will take it into consideration during future software development. If he spends effort to develop a

solution to a known bug or create a new feature for the software and shares it with the community, by contributing his own work, he seeks to receive feedback from others affirming the usefulness of his activities or further improving his solution (Shah 2006).

When the individual receives the benefits he expected prior to posting the message, his exchange experience with the community is successful, reinforcing his future participation behavior. When the individual fails to receive the expected benefits, he may be less likely to initiate further exchange activities with the community because the cost exceeds the actual benefits received. Furthermore, regardless of the purpose and content of his participation, the potential benefit that he seeks may not be realized unless other members engage in communications with him. When his participation leads to successful communication with the community and a satisfactory exchange experience, he is more likely to continue his participation in the future.

H2: A participant who received benefit through communications with others is more likely to repeat his participation in the community in the future.

3.3.2. Member Involvement in Innovation Process

3.3.2.1. Moderating Effect of Member Involvement on Relationship between Community Resource Availability and Benefit Creation

Unlike many online communities dedicated to information exchange and expertise sharing, open source software (OSS) project communities are formed around a software product – a technical artifact consisting of code. In other words, OSSD community members collaboratively produce a product. As summarized in Section 2, research has identified the various roles played by these members who are involved in the innovation process at different levels. Some members are highly involved in the process and have direct influence over the software code; some members are moderately involved in the process and provide feedback that may indirectly result in changes in the code; others

may not be involved at all and they have no direct or indirect influence over the final product.

In the information systems development process, user participation, which refers to "the behaviors and activities that the target users or their representatives perform" (Barki and Hartwick 1989, p.59), varies in the amount of influence that users have over the final product or the amount of direct control that they have over the development process (Ives and Olson 1094). Similarly, in the context of open source software development, we distinguish among three levels of OSSD community members' involvement in the innovation process, namely pre-usage information seeking, usage feedback, and product modification.

First of all, these three levels vary in the specific activities that members perform during the OSSD process. A few researchers have identified different types of communication activities that occur in OSSD. For example, von Krogh et al. (2003) categorize 22 types of email messages posted to the mailing lists of an open source software project – Freenet. These types include bug report, code contribution, expression of interest to contribute to the project, suggestion for improvements, etc. Sagers (2007) distinguishes between two broader types of communication activities in OSSD: support communication activities that assist users with problems during software usage and development communication activities that involve discussion of development issues. Similarly, we identify the types of activities associated with each level of involvement as listed in Table 3.1. Prior to adopting the software, the individual may be interested in assessing whether the software can satisfy his needs (i.e. will this software help him accomplish a task or is this an interesting project that is worth his participation). After confirming his need for the software, he may also request for assistance during the process of downloading, installing, and configuring the software for the first time. During the software usage process, he serves as a test bed of the software and may seek help on how to use a specific functionality of the software, report problems that he encountered, and suggest features that he desires. After he gains a deeper understanding of the software and is

interested in becoming more involved in the production process, he not only uses the software but may also attempt to extend the code to enhance its functionality and improve its quality either to satisfy his personal needs or to give back to the community.

Table 3.1 Activities Associated with Levels of Member Involvement in OSSD

Level of Involvement	Detailed Activity
Pre-usage Information Seeking	General question about project and/or software
	Request for help about download, setup, configuration, and compilation
Usage Feedback	Question about project specifics
	Problem report
	Feature request
Modification	Questions related to extending or modifying the existing code
	Contribution of code or other implementation details

More importantly, these levels of member involvement vary in the amount of control members have over the product and the level of knowledge they have about the product. When an individual is involved in the production process at the pre-usage level, responding to his participation does not require others to have very in-depth understanding of the product. Participation is driven by his personal software needs rather than enjoyment of coding, altruism, or the need to become involved in the software development process. He obtains personal benefits once his information request has been satisfied. The information sharing activity usually does not lead to any modifications to the software. This type of community is similar to online knowledge sharing groups and the ties formed through information sharing tend to be weaker. Hence the interactions that he has with other community members tend to be driven by his specific goals and the resulting relationships tend to be "narrowly functional and tenuous" (Bagozzi and Dholakia 2006).

When an individual is involved with the actual usage of the software, problems often arise either due to defects in the software itself or his lack of understanding of the software. He may need technical support on how to accomplish a specific task using the software. He may also suggest desirable features to be implemented in the future version of the software. All of these questions or requests cannot be raised without actually using

the software. Compared with users requesting project information and technical support, users submitting problem report and feature request have some familiarity with the software since they have been using the software for some time. Similarly, in order to provide answers to these questions or requests, community members need to have more experience with the software. Such participation in the form of providing product feedback and requesting support is likely to be driven by the individual's personal need or by his intent to help improve the software by providing inputs or both. His product feedback may or may not lead to modification of the software depending on whether other members acknowledge the importance of the feedback and, more importantly, whether someone is capable of and willing to make the corresponding changes in the code. Compared with members involved at the pre-usage level, he is more actively involved in the software production process by providing feedback to the developers even though he does not directly contribute to the code.

Compared with members involved at the pre-usage or the usage level, an individual involved at the modification level not only has a better understanding of the code but also has the technical expertise to understand and modify the code. He also reveals his modifications to the community or the project developers with an expectation that his contributions will be acknowledged and valued. In order to respond to his modification-related question, community members need to know the technical details of the software. If the participation is in the form of code contribution, usually only those members with code access privileges can make a decision as to the quality of the contributed code and whether the contributed code should be incorporated into the next release. Participation associated with code modification is likely to be driven by a desire to get feedback from others and seek a better solution or enjoyment of solving challenging problems. Among all the participants, this type of participants requires the most knowledge about the software and is most actively engaged in the production process.

We propose that OSSD communities may respond differently to the three types of member involvement. First, the effective size of potential respondents who have enough

experience with and knowledge of the software becomes smaller as the level of participation involvement increases. So a pre-usage-related message is more likely to receive a response from the community than a usage-related message or a modification-related message. Secondly, the cost incurred in responding to different types of messages is likely to be different. In a study of individual motivation and behavior in Apache field support activities such as those activities at the pre-usage or usage level, Lakhani and von Hippel (2003) find that many information providers spent small amount of time generating and posting an answer possibly because they already had the answer and did not need to research for an answer. However, the amount of cognitive effort required to respond to a modification-related message may be much greater. Hence, due to the differences in the size of potential respondent pool and the level of effort required to respond, we hypothesize that the relationship between community resource availability and community response varies with the level of member involvement. Specifically, the positive effect of community resource availability on benefit provision is weaker when the level of involvement is higher.

H3a: The relationship between community resource availability and community benefit provision is weaker when the individual is involved at the usage level than at the pre-usage level.

H3b: The relationship between community resource availability and community benefit provision is weaker when the individual is involved at the modification level than at the usage level.

3.3.2.2. Moderating Effect of Member Involvement on Relationship between Community Benefit Provision and Members' Repeated Participation

Marketing researchers have examined the role of switching costs in customer retention (Anderson 1994, Fornell 1992). Switching costs in general can be defined as the perceived economic and psychological costs associated with changing from one alternative to another. Jones et al. (2002) identify three dimensions of switching costs –

continuity costs, learning costs, and sunk costs. Continuity costs result from the loss of benefits accrued from continued usage of the product or service when the relationship is terminated (Maute and Forrester 1993). Learning costs are associated with pre-switching information search and evaluation as well as post-switching learning. Sunk costs are associated with the irretrievable resources such as money, effort, and time that people have invested in the current product or service (Arkes and Blumer 1985). Sunk cost effect refers to people's tendency to continue being engaged in an activity once they have invested irretrievable resources (Arkes and Blumer 1985). Although prior investments should not influence people's rational evaluation of current alternatives, several researchers have shown that people tend to take prior investments into consideration when deciding which alternative to take.

We borrow from the marketing literature on switching cost to examine how individual-product relationship may influence people's retention in innovation communities. In an OSSD community, when members stop participating they incur continuity costs and sunk costs, both of which are positively associated with their relationship with the product. For example, when they are involved in the innovation process at the modification level, the knowledge and expertise they have about the product cannot be easily transferred to another product and stopping participating may cause them to lose more benefits accrued from their deeper understanding of the current product than people involved at the pre-usage level or usage level. With regard to sunk costs, as people's knowledge about the product accumulates and their influence over the final product becomes greater, so do their irretrievable investments in the current project and product. When the individual is less involved in the innovation process, the cost already incurred is relatively low, in which case it may be easier for him to discontinue participating in the project community. When the individual is more involved in the process, the high cost he has already incurred in the project and the product may postpone or prevent his decision to discontinue participating in the future.

Hence, while benefit provision through communications with others may act as a positive reinforcement that increases the chances of repeated participation, the sunk costs already

incurred by the individual, which are associated with the level of involvement in the innovation process, moderates the positive influence of benefit provision on his repeated participation. We propose that

H4a: The influence of community benefit provision on members' repeated participation is greater for usage-level involvement than pre-usage-level involvement.

H4b: The influence of community benefit provision on members' repeated participation is greater for modification-level involvement than usage-level involvement.

In summary, we not only hypothesize the influence of innovation community benefit provision on community members' continued participation over time but, more importantly, propose that the magnitude of this influence depends on how members are involved in the innovation process.

3.4. Research Methodology

3.4.1. Data Source

We tested our research hypotheses by analyzing data collected from the online discussion forums of OSSD projects hosted at SourceForge.net, the largest open source software development web site in the world. SourceForge.net, currently hosting more than 180,000 projects and having more than 1.9 million registered users, provides a centralized platform for managing OSSD projects and facilitating project-related communications among developers and users. It offers a variety of services to hosted projects such as site hosting, mailing lists, bug tracking, message boards, file archiving,

and other project management tools. SourceForge.net has been an attractive data source for many OSS researchers (Howison and Crowston 2004).

An online discussion forum is an asynchronous electronic bulletin board system where individuals can post and respond to messages. It is used for communication and collaboration among developers and users by many OSSD projects. An individual with a question or request or other communication needs posts a message on the appropriate discussion forum. Any interested community member can read both the original message and any follow-up messages, and can respond to the message or further the discussion. Posters of most messages can be identified by their user names. In these discussion forums, messages are organized into threads, i.e., groups of messages that are responses, responses to responses, etc. to some initial message. Our focus is to examine the thread-initiating messages and analyze the follow-up messages in order to further explore the impact of community responses to earlier messages on subsequent participation by the message poster.

3.4.2. Sample

The focus of this chapter is open source project communities that are observed through activities in online discussion forums. We identify an individual as a member of an OSSD community if he is not formally listed as a project member and has posted at least one message in the community. We examined the messages posted to project forums during the observation period from February 7, 2005 to December 17, 2007. In total, 1,515,823 messages were posted in the discussion forums of 75,022 projects hosted at SourceForge.net[19]. The distribution of forum activity level in terms of number of messages in forums is presented in Table 3.2. Given the fact that the open source project population is highly heterogeneous and that the amount of communication activity in online forums varies significantly among projects, a random sample is likely to yield very few projects with highly active forums and much more projects with inactive forums. In order to provide better coverage of the project population with uneven distribution of forum activities, we obtained a stratified sample by dividing all SF.net projects into non-overlapping bins based on the volume of activity in their discussion forums and obtaining a proportion (3%) of projects from each bin[20]. The initial sample consisted of 423 projects.

In this study we are interested in the discussions initiated by non-project members in open source project communities and the response they receive from communities. Because the discussion forums of 111 projects in our initial sample contained messages posted by either project members or anonymous individuals exclusively, they were

---

[19] Over 60,000 SF projects either do not use online discussion forums at all or use discussion forums hosted at other sites.

[20] Because project age varies among all projects, it may be necessary to generate the strata based on the activity frequency (activity amount adjusted by project age) since a one-month-old project that has 100 forum messages may be very different from a one-year-old project with 100 messages. However, projects with low activity frequency may be older projects that were once highly active but eventually lose participation in the communities. And that change in the participation level is relevant to our study examining individual participation continuance. So what we are really interested in is to ensure that our sample covers a variety of projects with different levels of community activity, regardless of their age.

dropped from the sample and the effective project sample size became 312 projects. The distribution of the entire project population and the project sample based on their forums' activity level is presented in Table 3.2.[21] In total there were 5925 thread-initiating messages posted to the forums of the project sample during the observation period.

Table 3.2 Distribution of Forum Activity Volume

Number of Messages	**Number of Projects (Percentage)**	**Number of Sampled Projects (Percentage)**
3-10	12913 (59.86%)	143 (45.83%)
11-50	5135 (23.80%)	78 (25.00%)
51-250	2190 (10.15%)	59 (18.91%)
251-1250	960 (4.45%)	25 (8.01%)
1251-6250	319 (1.48%)	6 (1.92%)
6251-12500	26 (0.12%)	1 (0.32%)
12501-62500	29 (0.13%)	0 (0.00%)

Next, to ensure that our sampling approach did not bias toward projects of certain topics and targeted at certain audience, we checked the representativeness of the project sample in terms of target audience and project topic. Of the 312 projects, 165 projects target at end users including advanced end users and quality engineers, 158 target at developers, and 51 target at system administrators. Topics of these projects span from communications to formats and protocols and the distribution of their top-level topic category is summarized in Table 3.3. The topic distribution of the sample projects is consistent with that of the project population.

[21] Projects that have 1 or 2 forum messages are excluded from the sampling because these messages are usually welcome messages posted by project administrators to announce the purposes of forums.

Table 3.3 Distribution of Sample Projects Based on Topics

Top-Level Topic Category	Sample Frequency	Sample Percentage (%)	Population Frequency As of June 19, 2008	Population Percentage (%)
Communications	47	15.1	26,031	14.5
Security	13	4.2	4,876	2.7
Software Development	95	30.4	44,563	24.8
Desktop Environment	15	4.8	5,371	3.0
Text Editors	12	3.8	4,662	2.6
Database	17	5.4	10,192	5.7
Education	6	1.9	8,437	4.7
Games/Entertainment	38	12.2	26,975	15.0
Internet	65	20.8	39,611	22.0
Scientific/Engineering	35	11.2	24,310	13.5
Multimedia	45	14.4	23,518	13.1
Office/Business	37	11.9	16,846	9.4
Religion and Philosophy	1	0.3	560	0.3
System	70	22.4	32,146	17.9
Printing	3	1.0	751	0.4
Terminals	2	0.6	967	0.5
Other/Nonlisted Topic	2	0.6	5,377	3.0
Sociology	0	0.0	606	0.3
Formats and Protocols	13	4.2	5,932	3.3
Number of Projects	312		179,979	

3.4.3. Analytical Approach

There are two different dependent variables – a binary variable capturing whether or not a message received a response from the community and a variable representing the time until the occurrence of an event, repeated participation. Therefore, we adopted two different analytical approaches.

We adopted the logistic regression model to examine the impact of community resource availability on the likelihood that a message will receive a response as well as the moderating effect of member involvement on such an impact (H1 and H3). The model has the following form:

$$\log\left[\frac{p}{1-p}\right] = \alpha + \beta_1 x_1 + \ldots + \beta_p x_p$$

We then adopted the proportional hazards regression model (Cox 1972) to identify the factors influencing the likelihood that an individual will continue his participation in the future. According to the model that is generally used to investigate the probability and the timing of an event, the hazard rate for a subject is given by

$$h(t \mid x_1 \ldots x_p) = h(t)\exp(\beta_1 x_1 + \ldots + \beta_p x_p)$$

Where X is a vector of independent variables and the βs are the regression coefficients. The Cox regression model makes no assumption about the form of the underlying hazard function.

3.4.4. Measures

The unit of analysis is message. For each project, we identified all the messages posted by the same person. Then for each unique poster (community member who is not a project member) in each project's forums, we identified all the messages he posted and when they were posted. Figure 3.2 illustrates an example consisting of two projects – project 1 and project 2. Project 1 has two members – A and B - in its community and project 2 has one community member – C. Member A posted three thread-initiating messages to project 1 during our data collection period, of which only the first message received two responses before the next message was posted. Member B posted one thread-initiating message to project 1. Member C posted two messages to project 2 and the second message was responded to before the data collection ended. Then the sample would consist of all six messages. Each message is associated with a specific time interval. For instance, for member A in project 1, message 1is associated with time interval (T_1, T_2), message 2 is associated with time interval (T_2, T_3), and message 3 is associated with time interval (T_3, T_{end}). The outcomes we are interested in are: community benefit provision, which is operationalized as whether each message had received a response before the same member posted another message or before the data collection ended (i.e. during the associated time interval), and repeated participation,

which is operationalized as the occurrence and timing of the same member posting another thread-initiating message again before the data collection ended.

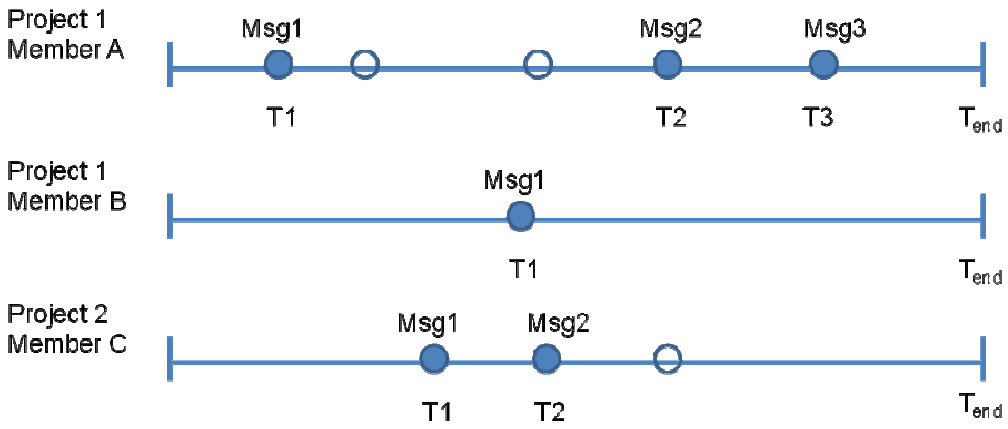

Figure 3.2 An Illustrative Example

3.4.4.1. Dependent Variables

Response – a binary variable indicating whether a message posted at T_1 received a valid response during time interval (T_1, T_2), where T_2 represents when next message is posted by the same person to the same project or when the data collection ends. When testing hypotheses H1a, H1b, H3a, and H3b, we are interested in the impact of community resource availability on community benefit provision as well as the moderating effect of member involvement. We operationalize community benefit provision as the ability of the community to generate a response to the message within the associated time period. Hence, we used *Response* as the dependent variable to test hypotheses H1a, H1b, H3a, and H3b.

Duration – the length of time in months from an individual's posting of a thread-initiating message to his posting of the next thread-initiating message, or until the end of data collection period, which was December 17, 2007, if no subsequent participation was observed. When testing hypotheses H2, H4a, and H4b, we are interested in the impact of community benefit provision on members' repeated participation and the moderating

effect of member involvement. And repeated participation can be operationalized as the return of a member to initiate another discussion in the same project again. In survival analysis censoring occurs when a subject's event time is unknown. One main cause of censoring is that some subjects will experience the event after the data collection ends. In other words, some members who had not posted another message by the end of our data collection period may did so after the data collection ended but we were unable to observe these events.

The censoring variable here is *Return*, indicating whether the subject has returned to the community to initiate another discussion. *Return* takes a value of 1 if the event occurred by the end of the data collection period and 0 if no event occurred before the data collection ended.

We recognize that OSSD participants are encouraged to post a message containing only one subject or question to improve the message clarify and to facilitate the community in understanding and responding to it. Hence, when some members have multiple requests that they need answers for, they tend to post multiple messages to initiate a discussion for each of their concerns within a very short period of time. For example, a participant A posted a message M_1 at time T_1 and posted another message M_2 to the same project at time T_2. If these two events occur within one hour, it is more likely that A posted two messages almost simultaneously to discuss two topics that he was interested in than A became more engaged in the community and continued to be an active participant.[22] Hence, *Return* is set to 0 for the event of posting M_2. There are 58 such events.

---

[22]Although a few messages received a response within a few minutes, 67.4% of the responses were posted between 1 hour and 6 days after the message was posted. And the mean of the duration between posting of a message and receiving the first follow-up message is 9.69 hours. As shown in Table D.1, as we increase the cut-off value for the interval between two postings when deciding whether the postings indicate continued participation or simultaneous postings of multiple questions, *Response* became a significant and positive predictor when the cut-off value was 20 minutes. As the threshold increased the magnitude of the impact of *Response* also increased.

3.4.4.2. Independent Variables

CommunitySize – number of non-project members who posted at least one message during the month prior to the posting of the focal message. Anonymous posters are counted as one community member. This measure approximates the amount of resources available in the community.

$CommunitySize^2$ – quardratic term of *CommunitySize*. This variable is used to test the marginal effect of resource availability on community benefit provision.

Involvement – a categorical variable representing the three possible levels of involvements for the individual, namely pre-usage, usage, and modification. In order to perform comparisons among the three types, we used the usage-level involvement as the base case. Hence, in the following analysis two dummy variables were used to capture the level of involvement – *Preusage* and *Modification*. *Preusage* indicates whether the message poster is involved in the OSSD process at the pre-usage level. Its value is set to 1 for messages containing general questions about the project and requests for help with setting up the software, 0 otherwise. *Modification* represents whether the message poster is involved at the modification level. It takes the value of 1 for messages containing code contributions as well as questions related to code modifications, and takes the value of 0 for all other messages. Section 3.4.5 explains the detailed procedures used to code this variable.

Response – a binary variable indicating whether the message posted at t_1 received a valid response during time interval (T_1, T_2). Section 3.4.5 contains the procedures of coding this variable.

3.4.4.3. Control Variables

We also controlled for the factors related to the message itself, the individual who posted the message, and the project associated with the message that may influence community responses to the message and the poster's continued participation in the community.

MessageLength – the natural log of the number of characters contained in the message. We controlled for the length of a message because often it may serve as an approximate measure of the amount of content in the message. A message with more detailed content may be easier to understand and respond to.

PriorMessage – the number of thread-initiating messages that the individual has posted to the project prior to the focal message. This variable captures the level of experience of the message poster with regard to the specific project instead of experience with the open source software development paradigm. The more discussions the poster has started in the past, the more familiar he is with the community and the project. And he may be more likely to participate again in the same project community.

PriorReturn – the individual's propensity toward repeated participation in other projects prior to the posting of the focal message. This is approximately measured by calculating the ratio of the number of other projects where he has posted at least two thread-initiating messages to the number of other projects where he has contributed at least one thread-initiating message. If a person has a tendency to remain in a project by contributing multiple messages over a period of time in the past, he may be more likely to continue his participation in the focal project community as well.

OtherProjectMembership – the number of projects that the message poster is affiliated with as a formal project member when the focal message is posted. This approximately measures the level of effort that the person needs to spend on other projects. The more projects that he is working on, the less time that he may have to spend on the focal project and the less likely that he may come back and participate again in the focal project.

Downloads – the natural log of the number of times the software has been downloaded in a month prior to the posting of the focal message. This measure approximates the

diffusion success of the product immediately before the posting of the message. Communities producing more successful products may be more responsive to members' requests and suggestions. Successful projects may also be considered more favorably when members decide whether to remain in their communities or to quit.

ProjectStage – a categorical variable ranging from 1 to 6 representing the development stage of the project, namely planning (1), pre-alpha (2), alpha (3), beta (4), production (5) and mature (6). Some projects have been in existence for a long period of time and have developed mature software whereas others are more recent and only have some preliminary code. A person's choice of remaining in a project community may be influenced by the maturity of the project. Some people may prefer staying in the community when the project is less developed because the code is easier to understand and extend over. Others may prefer repeatedly contributing to the community when the project is more established and the software is more stable and functional.

ProjectAge – the number of months between the project's registration at SF.net and the posting of the focal message. Because *ProjectStage* is determined by the project administrator, it may not accurately capture the actual development stage of the project. Hence we also controlled for *ProjectAge* that may serve as a proxy for the development status of the project.

TargetDeveloper – a binary variable representing the intended audience of the project. Its value is 1 if the project is targeted at developers and system administrators and 0 if targeted at end users. Controlling for this factor is because software developed for developers and system administrators may have "strong community appeal" (Lerner and Tirole 2005, Stewart et al. 2006), thus are more likely to attract people to return to the community and remain active over time.

RestrictiveLicense – a binary variable indicating whether the project has a restrictive or nonrestrictive license. This variable is included because license restrictiveness has been

found to influence user perceptions of OSSD projects (Stewart et al. 2006), which may influence how likely members may continue their participation in OSSD communities. We used the categorization adopted by Lerner and Tirole (2005) to code this variable. Licenses such as the GNU GPL license and the LGPL license are categorized as restrictive and licenses such as the BSD license are categorized as nonrestrictive.

3.4.5. Content Analysis of Messages and Responses

Level of a member's involvement in the OSSD process may be determined by examining the content of his message. However, due to the lack of a common set of keywords or phrases that community members used in their messages and due to the fact that many messages contain segments of various software code in different programming languages, coding the level of involvement using text mining tools is relatively difficult and may yield unreliable results.

Therefore, content analysis, which has been widely adopted to provide systematic and objective description of the content of communication, was performed on all messages posted by non-project members to the sample projects' forums to categorize the level of involvement in the OSSD process based the coding scheme as presented in Appendix C. Coders included the author of this dissertation familiar with open source software development and a domain expert (a software developer with 10 years of programming experience).

Prior to the actual coding process, the coders went through the coding scheme (Appendix C) and thoroughly discussed each category as well as the similarities and differences among the categories. Then they coded a training sample of 23 messages consisting of several messages from each category to further familiarize themselves with the coding scheme. Following the training session, they completed coding for a pilot sample consisting of 90 messages with Cohen's kappa (Cohen 1960) being 0.36. To further clarify the criteria used in the content analysis and the boundaries of each category, they carefully reviewed the pilot sample and explained to each other the rationale for their

coding decision. After revising the coding scheme, another pilot sample of 90 messages were coded. Cohen's kappa (Cohen 1960) increased to an acceptable level ($k = 0.89$).

In the next step, the coders independently read the 5925 thread-initiating messages and identified the types of participation these messages represented.[23] The coders agreed on 5222 of the 5925 messages (88.1%) and Cohen's kappa was 0.84, suggesting adequate inter-rater reliability. Then coding discrepancies were reviewed and corrections were made to these messages' types.

The distribution of sample messages based on their level of involvement is summarized in Table 3.4. We dropped the irrelevant messages from further analysis and the final sample for further analysis included 5369 messages.

[23] During the process if the coders did not think a message fell into any of the eight identified categories, they were asked to summarize the purpose of the message. Then the two coders went through the short summaries together and as a result added three additional categories so that the coding scheme can better reflect the content of all sample messages. These three categories include: questions and comments about project administration related issues such as release plan and project status, information and experience sharing among users of the software, and requests to join the project. However, because the relatively small proportion (4.10%) of messages that belong to these three categories, we decided to keep the first seven categories presented in Table 3.4 and coded all other messages as irrelevant.

Table 3.4 Distribution of Sample Thread-Initiating Messages Based on Member Involvement

Type of Involvement	Detailed Message Content	Frequency	Percentage (%)
Pre-Usage	General question about software such as software and hardware compatibility, general purpose, documentation	159	2.68
	Questions related to download, installation, configuration, compilation, demonstration, example, tutorial, training sample	454	7.67
Usage	Questions about how to use specific features of software	2068	34.90
	Request for features that have not been implemented in software	251	4.24
	Request for help with trouble-shooting and report of problems during usage (either due to user's lack of understanding or a bug in the code)	2016	34.03
Modification	Solutions to problems encountered, contribution of patches, code fixes, and contribution of user guide	242	4.08
	Questions related to modifying and extending the software such as a code change proposal, design change proposal	179	3.02
Irrelevant	All other messages	556	9.38

Next, we retrieved all follow-up messages posted to the threads initiated by these messages in order to examine their contents more closely. Of these 5369 messages 1231 messages did not have any follow-up messages posted by members other than the original poster during the observation period. Together the remaining 4138 messages received 8448 follow-up messages from others (subsequent messages posted by the original poster were excluded). On average each message had 2 follow-up messages posted to the same discussion thread.

Building upon past studies of face-to-face and online discussions, we categorize responding messages based on their knowledge content. Knowledge content refers to whether the information contained in the message is new contribution, old repetition, or null content with regard to the original message (Chen and Chiu 2008, Chiu 2000). Contributions may include new ideas, justifications, critiques, alternatives, assessments, and so on. A responding message containing new knowledge contribution directly

provides the information requested by the original poster or evaluates the content contained in the original message. Or it may provide a confirmation or promise that progress is being or will be made to directly respond to the original message. Repetitions repeat the knowledge content in the original message (Schegloff 1996). A responding message containing repeated content is often posted by someone else sharing similar questions or opinions with the original poster. Null content does not include any new information directly related to the original message. A responding message with null content may be a request for clarification due to missing or ambiguous information in the original message, or an off-topic message, or simple evaluations with no detailed information (Schellens and Valcke 2005), and so on. In this study we are most interested in the responses containing new knowledge content that directly responds to the original message. The term "response" used in the following discussions refers to such responses unless noted otherwise.

Basing on the knowledge content contained in a follow-up message, the responding messages were coded as either valid or invalid responses. Valid responses are those follow-up messages that directly provide the information requested by the original poster or evaluate the content contained in the original message or provide a confirmation or promise that progress is being or will be made. The domain expert independently examined a random subset of 100 responses and coded them into valid and invalid responses. The two coders agreed on coding of 92 responses. Given the high inter-rater reliability with the sample, the domain expert did not need to code the full set of responses.

3.5. Analysis and Results

3.5.1. Descriptive Statistics

Descriptive statistics are presented in Table 3.5. 62% of the sample messages received a response before their posters participate again. 37% of the message posters initiated another discussion thread during the observation period. On average approximately 16 individuals who are not formally listed as project members contributed at least one message in the one-month period. Pairwise correlations reported in Table 3.6 show that all correlations are below 0.46. We also checked the Variance Inflation Factor (VIF) values for all the predictor variables in each regression model and all VIFs are below 2.0, indicating that multi-collinearity is not a major concern.

Table 3.5 Descriptive Statistics

Variable	Mean	Std. Dev.	Min	Max
Response	0.62	0.484	0.00	1.00
Return	0.37	0.484	0.00	1.00
CommunitySize	15.70	14.613	0.00	60.00
Preusage	0.11	0.318	0.00	1.00
Modification	0.08	0.269	0.00	1.00
PriorMessage	1.89	4.386	0.00	38.00
MessageLength	6.13	0.902	1.39	10.40
PriorReturn	0.06	0.213	0.00	1.00
OtherProjectMembership	0.24	0.844	0.00	11.00
Downloads	6.86	2.053	0.00	12.10
ProjectAge	37.12	19.700	0.04	97.23
TargetDeveloper	0.71	0.455	0.00	1.00
RestrictiveLicense	0.64	0.480	0.00	1.00
ProjectStage	5.34	0.929	1.00	7.00

Table 3.6 Pairwise Correlations

Variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Response							
Return	0.118***						
CommunitySize	0.091***	0.266***					
Preusage	0.008	-0.076***	-0.138***				
Modification	-0.022	0.016	-0.036***	-0.105***			
PriorMessage	0.044***	0.331***	0.184***	0.069***	0.031**		
MessageLength	0.020	0.018	0.080***	-0.073***	0.122***	-0.049***	
PriorReturn	-0.022	-0.027*	-0.101***	0.020	0.007	-0.010	-0.027**
OtherProjectMembership	-0.001	0.014	-0.015	0.000	0.013	0.004	-0.005
Downloads	-0.029**	0.016	0.365***	-0.131***	0.001	0.012	0.026*
ProjectAge	-0.094***	-0.035**	0.147***	-0.037***	0.021	0.071***	0.048***
TargetDeveloper	-0.062***	-0.080***	0.273***	-0.132***	-0.038***	-0.181***	0.073***
RestrictiveLicense	-0.143***	-0.174***	-0.458***	0.123***	0.017	-0.072***	-0.084***
ProjectStage	-0.064***	-0.096***	-0.012	-0.051***	0.002	-0.026*	0.034**
	(8)	**(9)**	**(10)**	**(11)**	**(12)**	**(13)**	**(14)**
OtherProjectMembership	0.015						
Downloads	-0.061***	-0.016					
ProjectAge	-0.063***	-0.006	0.418***				
TargetDeveloper	0.047***	-0.018	0.319***	0.234***			
RestrictiveLicense	-0.010	0.012	-0.006	0.072***	-0.311***		
ProjectStage	-0.036***	0.005	0.178***	0.366***	0.100***	0.142***	
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$							

3.5.2. Results of Hypothesis Testing

Table 3.7 summarizes the results for the logistic regression used to test hypotheses H1 and H3. As we hypothesized, whether a message can receive a response from the community depends on community membership size that indicates resource availability. The parameter estimate for *CommunitySize* is positive and significant ($\beta = 0.018$, $p < 0.01$), suggesting larger communities are more likely to generate a response to the message, consistent with positive network externalities found in online communities (Gu et al. 2007). Hence, H1a is supported. Moreover, the marginal benefit of community resources decreases with membership size because the parameter estimate for the quadratic term of *CommunitySize* is significantly negative ($\beta = -0.001$, $p < 0.01$), consistent with the finding of diminishing marginal return on network size (e.g., Asvanund et al. 2004). Therefore H1b is also supported.

In order to test H3a and H3b associated with the impact of member involvement, we added the dummy variables *Preusage* and *Modification* to the regression model as well as the interaction terms between them and *CommunitySize*. The results show that the parameter estimates and significance levels of the other independent and control variables are consistent with those prior to introducing *Preusage* and *Modification* to the model. Level of member involvement does seem to have a significant direct effect on the likelihood of receiving a response. Specifically, compared with usage-level relationship, when *CommunitySize* remains the same, pre-usage-level relationship is 20.0% more likely to receive a response from the community ($\beta = 0.183$, $p < 0.10$) whereas modification-level relationship is 21.1% less likely to receive a response ($\beta = -0.236$, $p < 0.05$). The interaction effect between *CommunitySize* and *Preusage* is insignificant whereas the interaction effect between *CommunitySize* and *Modification* is significantly positive ($\beta = -0.028$, $p < 0.01$). In other words, the impact of *CommunitySize* is not statistically different for pre-usage involvement and usage involvement; the influence of *CommunitySize* on community benefit provision is significantly smaller for modification-level involvement than usage-level involvement. Hence, H3a is not supported; H3b is supported.

Table 3.7 Results of Logistic Regression on Community Response (H1a, H1b, H3a and H3b)

Variable	H1a, H1b		H3a, H3b[24]	
	Parameter Estimate	**Odds Ratio**	**Parameter Estimate**	**Odds Ratio**
Intercept	1.862***		1.766***	
CommunitySize	0.018***	1.018	0.020***	1.020
CommunitySize^2	-0.001***	0.999	-0.001***	0.999
Preusage			0.183*	1.200
Modification			-0.236**	0.789
CommunitySize × *Preusage*			0.006	1.006
CommunitySize × *Modification*			-0.028***	0.973
PriorMessage	0.000	1.000	0.002	1.002
MessageLength	0.032	1.033	0.044	1.045
PriorReturn	-0.139	0.871	-0.135	0.874
OtherProjectMembership	0.003	1.003	0.003	1.003
Downloads	-0.006	0.994	-0.004	0.996
ProjectAge	-0.008***	0.992	-0.008***	0.992
TargetDeveloper	-0.417***	0.659	-0.420***	0.657
RestrictiveLicense	-0.676***	0.509	-0.684***	0.505
ProjectStage	-0.046	0.955	-0.043	0.958
Likelihood Ratio	248.867***		266.576***	
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$				

In addition, the results suggest that some project characteristics such as project age, license, and target audience all influence the likelihood of generating responses to members' participation. Specifically participation is more likely to receive a response from the projects that are newer, adopt a less restrictive license, and target at end users.[25]

[24] When including the interaction terms between *Preusage* and *CommunitySize* and between *Modification* and *CommunitySize*, we centered *CommunitySize* by subtracting the mean from the original value to minimize correlations and ensure stable results.

[25] In the above analysis we assume that the observations are independent. However, in the data set because a single individual may contribute multiple events or a single project may be associated with multiple repeated participation events, the error terms may be correlated rather than being independent among observations. Hence, we used the generalized estimating equations (GEE) method (Diggle et al. 1994, Liang and Zeger 1986) to produce standard errors and test statistics that are adjusted for observation dependence. Assuming that observations are independent between clusters and correlated within clusters and each cluster is distinguished by

Table 3.8 presents the results of fitting the Cox proportional hazard model to test H2, H4a and H4b. Hazard ratios range from 0 to positive infinity and measure the impact of predictor factors on the relative risk of an event occurrence. The results suggest that receiving a valid response to a message increases the likelihood of the message poster participating again in the community by 11.3% ($\beta = 0.107$, $p < 0.05$). Hence, H2 is supported.

---

different message poster or different project, we obtained similar results as those presented in Table 7.

Table 3.8 Results of Cox Proportional Hazards Model on Repeated Participation (H2, H4a and H4b)

Variable	**H2**		**H4a, H4b**	
	Parameter Estimate	**Hazard Ratio**	**Parameter Estimate**	**Hazard Ratio**
Response	0.107**	1.113	0.116**	1.123
PreusageParticipation			-0.333**	0.717
ModificationParticipation			0.090	1.094
Response×*Preusage*			0.061	1.062
Response×*Modification*			-0.117	0.889
PriorMessage	0.071***	1.074	0.070***	1.073
MessageLength	0.008	1.008	0.003	1.003
PriorReturn	-0.181	0.835	-0.176	0.839
OtherProjectMembership	0.020	1.020	0.018	1.018
Downloads	0.048***	1.049	0.042***	1.043
ProjectAge	-0.002	0.999	-0.001	0.999
TargetDeveloper	-0.378***	0.685	-0.396***	0.673
RestrictiveLicense	-0.619***	0.538	-0.607***	0.545
ProjectStage	-0.101***	0.904	-0.107***	0.899
Observations	5369		5369	
χ^2	699.876***		713.656***	
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$				

After introducing the two dummy variables indicating involvement levels (pre-usage and modification) in OSSD process and their interaction effects with *Response* into the regression model, we find that compared with individuals participating at the usage level, participants at the pre-usage level are 28.3% less likely to return to the community and initiate discussions again (β = -0.333, $p < 0.05$). Individuals participating at the usage level and modification level are not statistically different in terms of their likelihood of continuing participating in the community (β = 0.090, n.s.). Surprisingly, the parameter estimates of the two interaction terms are not statistically significant (*Response*×*Pre-usage*: β = 0.061, n.s.; *Response*×*Modification*: β = -0.117, n.s.), suggesting that the impact of *Response* does not depend on member involvement. In other words, regardless of type of involvement, receiving a response to members' participation increases their likelihood of repeating participation by 12.3%. And among all participants those individuals posting pre-usage related messages are the least likely to return and participate again. Therefore, neither H4a nor H4b is supported.

In addition, the results seem to suggest that individual participation history in the community, *PriorMessage*, positively influences his repeated participation behavior in the future. The more thread-initiating messages he has posted in the past, the more likely he will continue participating in the future. Some project characteristics such as number of software download, project maturity, license, and target audience all play a role in driving participants' repeated participation behavior. Individuals are more likely to continue participating in the projects that have more downloads, adopt a less restrictive license, target at end users and are in an earlier stage of development.[26]

3.6. Conclusion

Motivated by the increasing prevalence of online communities collaboratively developing innovations and the relatively high rate of member turnover in these communities, this study focuses on members' repeated participation behavior in open source software development communities. We empirically investigate the roles of community benefit provision and individual involvement in the innovation process in sustaining members' participation. Specifically we attempt to examine how direct communication activities among members convert community resources into benefit obtained by members and how such benefit provision induces members' continued participation in the future. We also try to identify how members' level of involvement in the innovation process

---

[26] However, because 837 individuals (24% of all individuals associated with our message sample) contributed at least two participation events to our data set, there may be some dependence among the events associated with these repeated observations. As pointed out in Allison (2005), failure to adjust for the possible dependent error terms may cause underestimated standard errors and p-values. However, the approach proposed in Allison (2005) that introduces fixed effects into the Cox regression model by using stratification to divide the observations into subgroups based on message posters has limited applicability to our data set. First, like other fixed effects models, fixed effects Cox regression model suffers from a loss of statistical power because any individual who contributed only one message in the sample is excluded from the analysis. Even for the remaining observations the model only uses information about variation within each individual rather than across each individual. Furthermore, the estimates of variables may be biased especially when the average number of events per individual is low and the proportion of censored observations is high, two characteristics exhibited by our data set.

influences the impact of community benefit creation on members' continued participation.

Consistent with our argument that participants in OSSD communities are not isolated entities, their subsequent participation behavior is to some extent influenced by how other members react to their prior participation. When the community generates the benefit such as informational benefit that an individual expects to receive, he is more likely to continue his participation in the community. Then what influences the benefit creation process in the community? We find that consistent with resource-based view of online social structures proposed in Butler (2001) the amount of resource plays a key role. Communities with more resources tend to be more responsive to members' participation but the marginal effect of resources exhibits a diminishing pattern as more resources become available.

Besides the community-related factors, the individual's involvement in the production process also influences how the community responds to his participation and how likely he will remain in the community. Level of involvement not only directly impacts the likelihood of receiving responses from the community, which is more likely to respond to messages posted by individuals less involved in the software production process, but also moderates the impact of community resource availability on community response. The benefit of having more resources is weaker for individuals involved at the modification level than for those involved at the pre-usage and usage level. Furthermore, we find that participants at the pre-usage level are much less likely to return to the community in the future than other participants. Interestingly, unlike what we hypothesized, the impact of community response on repeated participation does not seem to vary among pre-usage, usage, and modification levels of involvement.

The lack of interaction effect between level of involvement and community response may be due to the fact that some benefits obtained by participants are difficult to observe directly. In the essay we only considered the benefit that is created through direct interactions among community members. However, besides informational benefit and community acknowledgement or feedback offered by responses to their participation,

individuals may obtain learning benefit or enjoyment benefit through their interactions with the software itself. For example, hobbyists involved in modification of software code often enjoy coding as an intellectually satisfying activity. It is likely that as the relationship between the individual and the software grows stronger, the benefit obtained from interactions with the software increases, thus the expected benefit of remaining in the community increases, leading to a greater likelihood of repeated participation.

This research theoretically contributes to a deeper understanding of member participation dynamics in online innovation communities. Although many cross-sectional studies have examined motivations of member participation in general online communities, little is known about when and under what conditions members continue their participation in online communities and even less is known about what factors help sustain member participation in online innovation communities. We identify that innovation communities differ from information-sharing communities in that members in innovation communities are collaboratively developing a central product. We propose that benefit provision through communications between an individual and other community members is not sufficient to explain the individual's behavior. The involvement of individual in the production process also plays a key role in influencing his behavior. Therefore, we extend the existing literature on benefit provision in online communities by highlighting and distinguishing levels of member involvement in the innovation process.

Practically, our findings help guide OSSD projects in facilitating the benefit provision process and attracting and retaining participants. First and most importantly, improving the responsiveness of the project community to members' participation will likely help promote their retention in the future. However, since the community often relies on a self-selection mechanism to provide responses to members' participation and project developers are often occupied with code development and maintenance, it may not be possible for the community to increase its responsiveness to the fullest extent. Given that members having more experience with the software are more likely to continue their active involvement in the community, project managers and developers may encourage these members' retention by assigning personnel to actively respond to their messages in

addition to relying on the community's self-selection mechanism. Secondly, we find that communities are most likely to respond to messages posted by people with least amount of knowledge about the software but they are also the people who are the least likely to maintain their participation in the future regardless of the responses they receive from communities. While the self-selection mechanism in choosing which messages to respond to may be effective in enabling community members to help novice users, its role in attracting the repeated participation by novice users is limited. Sometimes it may be best to complement community's self-selection mechanism in allocating resources with some monitoring and intervention effort by the project administrators. Thirdly, we also find that members prefer remaining in communities of projects that are in an earlier stage, target at end users, and adopt a less restrictive open source license, which may help explain why some projects tend to have larger and more active communities than others. This finding may suggest that some project characteristics affect their ability to build and sustain successful communities; some projects are more likely to attract and sustain member participation and more suitable for community-based production mode than other projects.

Our findings also reveal that not all individuals behave the same in the OSSD community. Some may obtain great benefits and satisfaction from their involvement in the software production process, hence they may care less about the community's responses to their participation and remain active in the community even in the absence of such responses. Others may value the community's responses to their participation much more than the benefits they derive from participating in the production process, so their decision to stay or leave the community is highly dependent on the community's reaction. Future research may examine the composition of OSSD project communities in terms of these two types of members and identify how such composition can impact the long-term success and sustainability of OSSD projects. In addition, it would be interesting to explore the progression of member involvement from a lower level to a higher level over time or the regression from a higher level to a lower level. So far little is known about the contextual factors that influence the member's role migration in OSSD process. For example, is role migration driven by interactions with the project

core members or interactions with the project community or the evolution of the software itself? How can project managers encourage role progression through appropriate monitoring of the community activity and allocation of project resources?

We also acknowledge that the current study has a few limitations. First, the repeated participation event under study is limited to the posting of another thread-initiating message. But continued participation may be also reflected from the activity of answering questions and participating in (not necessarily initiating) discussions. While including repeated participation in terms of both initiating and participating in discussions will likely enrich our understanding of participants' behavior in OSSD communities, due to the focus of this study on community benefit creation, we chose to examine only discussion-initiating behavior of community participants. Secondly, we mainly examined community response in terms of existence of a valid response directly responding to the original message. However, there are many other aspects related to community response such as source of response (i.e. whether another community member responds or a project member responds), timeliness of response, amount of discussion involved in the entire thread, number of participants involved in the thread discussion, which are yet to be examined.

CHAPTER 4. CONCLUSION

4.1. Summary

In the first essay, we explore the influence of the OSSD collaboration network on individual choices of newly-created projects to join. The findings suggest that the network of collaborative ties among developers does affect developer decisions to join new projects and form project teams. Developers rely on their relationships with others forged through past interactions to evaluate the quality of their prospective collaborators as well as the attractiveness of a new project. Overall, the strategic decisions made by developers in the self-assembly of OSS project teams are influenced by their past collaborative interactions in the OSSD network. These decisions in turn influence the formation of future interactions among developers, which then impacts the overall structure of the entire network.

In the second essay, we investigate the impacts of community benefit provision and individual involvement on individual continued participation in OSS user communities. On one hand, we find that the interactions with other community members do influence an individual's return to the community. Specifically, the benefit received by the individual through direct communications with other members helps keep the individual in the community; receiving a response to a member's prior participation does attract him to return to the community and participate again. Furthermore, such a benefit provision process is subject to both positive and negative network externality. On the other hand, individual differences do exist in members' returning behavior; they behave differently depending on their level of involvement in the OSSD process.

In summary, from a network perspective this dissertation investigates the voluntary decisions by participants with regard to two key elements of OSSD – project team and user community. Overall, the findings suggest that the networks that an individual is embedded in influence his choices and behavior. The OSSD collaboration networks impact developers' choices of new projects to join during the self-assembly process of new project teams. In the OSS user community interactions between an individual and other community members directly affect his continued participation in the community.

4.2. Contributions and Implications

Theoretically, despite the increasing pervasiveness of self-organizing teams and communities in the digital environment during the past few years, little is understood about the preferences of individuals in this self-organizing process. Even less is known about their self-selection into online peer productions and their continued participation in these productions. The findings in this dissertation shed light on individual behavior in forming and sustaining online self-organizing groups or communities. Hence, this dissertation contributes to a deeper understanding of the self-selection behavior in the new mode of production through large-scale voluntary collaborations in online communities. Furthermore, this dissertation proposes to investigate individual self-selection behavior in a networked environment from a network perspective instead of investigating it as a stand-alone phenomenon. The web of relationships provides opportunities and place constraints on people's behavior, hence their behavior should not be analyzed in a segmented fashion.

In addition, given the dynamic nature of the OSSD process, this dissertation analyzes longitudinal data sets collected from real-world OSS projects. Methodologically it is able to distinguish the network that exists before the behavior occurs and the network after the behavior occurs. Hence, the causal relationships between networks and individual behavior can be better understood. More importantly, the longitudinal data sets capture the sequence of events under study, analysis of which can better reveal the temporal relationships between events that occur during the observation period. For example, the

data set in Chapter 2 allows us to identify the collaborative relationships each individual had prior to joining a project as well as the time-varying project and developer characteristics at the time of joining, which allowed us to identify the causal relationships between the predictors and the joining event. The data set in Chapter 3 consisted of all participation events associated with members of the OSSD project community, enabling us to examine the relationship between the previous participation and the current participation, the current participation and the next participation, and so on. In summary, this dissertation complements the existing OSS research, most of which is cross-sectional, by analyzing multiple events over time.

Furthermore, in the digital environment an unprecedented amount of research data is being captured online and stored, bringing researchers many opportunities to directly observe individual online behavior in highly complex real-world settings. This dissertation demonstrates a proof-of-concept of how to use these web-enabled data to reconstruct and examine interesting phenomena without having to take subjects out of their real-world context (Hahn and Zhang 2008).

Practically, this dissertation provides some insights to OSS project managers and organizations interested in the open source mode of software production. Despite the fact that many OSS projects fail to take off and become abandoned (Chengalur-Smith and Sidorova 2003), the existing OSS literature offers limited guidance on how to attract volunteers into newly-initiated projects and even less guidance on how to sustain volunteer participation in user communities formed around OSS projects. Our findings regarding early team formation mechanisms offer some actionable guidelines that may help project initiators increase the attractiveness of their new projects and obtain early momentum by attracting developers into their teams early on. Our findings in the second essay help guide OSS projects in terms of how to facilitate the community benefit provision process by improving community responsiveness, which in turn attracts members' continued participation in the community.

4.3. Future Research

This dissertation is our first step in a stream of research that investigates team and community dynamics in the OSSD process. Many interesting and important research questions remain open and invite future investigations.

For example, although we have explored team formation mechanisms and community participation patterns in two individual essays, the interaction between the two key elements in OSSD – core project teams and larger-scale user communities – is yet to be explored. In this dissertation we have treated these two elements separately. Future research may target at the interplay between teams and communities by answering questions such as "how do successful interactions between community members and team members help promote community members' contributions" and "how does an active community help attract developers into the project team". In addition, team formation and community building may have different performance implications during different stages of project development. Examining them in a longer time interval may reveal the critical events related to attracting developers and community participation.

Furthermore, this dissertation focuses on the entry decisions of project team members and the continuance behavior of user community members. Future studies may gather additional data from multiple sources to further investigate the continuance behavior of project team members (i.e. why do members stop participating?) and the joining decisions of user community members (i.e. what triggers members' initial participation?), which was not observable in our current data sets.

In addition, OSSD projects do not exist in isolation; many similar or dissimilar projects co-exist in a competitive environment. This dissertation looks at the influence of interpersonal relationships on OSS team and community dynamics. Extending this line of research, future research may investigate the influence of external environment where the project is situated on team and community dynamics. For example, examining how inter-project competitions alter the self-selection behavior of team members may provide

some explanations for the large number of OSSD projects that failed to attract developers and became abandoned.

LIST OF REFERENCES

LIST OF REFERENCES

Adler, P. S., and S.-W. Kwon. 2002. Social capital: Prospects for a new concept. *Academy of Management Review* **27**(1) 17-40.

Anderson, E. W. 1994. Cross-category variation in customer satisfaction and retention. *Marketing Letters* **5**(1) 19-30.

Antwerp, M. V. and G. Madey. 2008. Advances in the SourceForge Research Data Archive (SRDA). *Proceedings of the 4th International Conference on Open Source Systems*, Milan, Italy, September 10, 2008, 22 - 27.

Ardichvili, A., V. Page, and T. Wentling. 2003. Motivation and Barriers to participation in virtual knowledge sharing teams. *Journal of Knowledge Management* **7**(1) 64-77.

Arkes, H. R. and C. Blumer. 1985. The psychology of sunk cost. *Organizational Behavior and Human Decision Process* **35**(1) 124–140.

Asvanund, A., K. Clay, R. Krishnan, and M. D. Smith. 2004. An empirical analysis of network externalities in peer-to-peer music-sharing networks. *Information Systems Research* **15**(2) 155-174.

Bagozzi, R. P. and U. M. Dholakia. 2006. Open source software user communities: A study of participation in Linux user groups. *Management Science* **52**(7) 1099-1115.

Barabási, A.-L. and R. Albert. 1999. Emergence of scaling in random networks. *Science* **286**(5439) 509-512.

Barki, H. and J. Hartwick. 1989. Rethinking the concept of user involvement. *MIS Quarterly* **13**(1) 53-63.

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.

Benkler, Y. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, New Haven, CT.

Bhattacherjee, A. 2001. Understanding information systems continuance: An expectation-confirmation model. *MIS Quarterly* **25**(3) 351-370.

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

Boone, K. S., P. Beitel and J. S. Kuhlman. 1997. The effect of the win/loss record on cohesion. *Journal of Sport Behavior* **20**(2) 125-143.

Burt, R. 1992. *Structural Holes: The Social Structure of Competition*. Harvard University Press, Cambridge, MA.

Butler, B. S. 2001. Membership size, communication activity, and sustainability: The internal dynamics of networked social structures. *Information Systems Research* **12**(4) 346-362.

Butler, B., L. Sproull, S. Kiesler, and R. Kraut. 2007. Community effort in online groups: Who does the work and why? *Leadership at a Distance: Research in Technologically-Supported Work.* S. Weisband, ed. Lawrence Erlbaum, Mahwah, NJ, 171-194.

Chen, G. and M. M. Chiu. 2008. Online discussion processes: Effects of earlier messages' evaluations, knowledge content, social cues and personal information on later messages. *Computers and Education* **50**(3) 678-692.

Chengalur-Smith, S. and A. Sidorova. 2003. Survival of open-source projects: A population ecology perspective. S. T. March, A. Massey and J. I. DeGross, eds. *Proceedings of the 24th International Conference on Information Systems*, Seattle, WA, December 14-17, 2003, 782-786.

Chiu, M. M. 2000. Group problem solving processes: Social interactions and individual actions. *Journal for the Theory of Social Behavior* **30**(1) 27-50.

Chiu, C., M. Hsu, and E. Wang. 2006. Understanding knowledge sharing in virtual communities: An integration of social capital and social cognitive theories. *Decision Support Systems* **42**(3) 1872-1888.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* **20**(1) 37-46.

Coleman, J. S. 1990. *Foundations of Social Theory*, Harvard University Press, Cambridge: MA.

Cox, D. R. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society* **34**(2) 187-220.

Crowston, K. and J. Howison. 2006. Assessing the health of open source communities. *IEEE Computer* **39**(5) 89-91.

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

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

Cummings, J. N. and S. Kiesler. 2007. Who works with whom? Collaborative tie strength in distributed interdisciplinary projects. *E-Social Science*, Ann Arbor, MI, October 7-9, 2007.

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.

Dholakia, U. M., R. P. Bagozzi, and L. K. Pearo. 2004. A social influence model of consumer participation in network- and small-group-based virtual communities. *International Journal of Research in Marketing* **21**(3) 241-263.

Doong H. and H. Lai. 2008. Exploring usage continuance of e-negotiation systems: Expectation and disconfirmation approach. *Group Decision and Negotiation* **17**(2) 111-126.

Ducheneaut, N. 2005. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work* **14**(4) 323-368.

Efron, B. 1982. *The Jackknife, the Bootstrap and Other Resampling Plans.* SIAM, Philadelphia, PA.

Efron, B. and R. J. Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science* **1**(1) 54-75.

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

Fornell C. 1992. A national customer satisfaction barometer: The Swedish experience. *The Journal of Marketing* **56**(1) 6-21.

Frank, R. H. 1985. *Choosing the Right Pond: Human Behavior and the Quest for Status*. Oxford University Press, New York.

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

Gibbert M., M. Leibold, and G. Probst. 2002. Five styles of customer knowledge management, and how smart companies use them to create value. *European Management Journal* **20**(5) 459-469.

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

Granovetter, M. 1985. Economic action and social structure: The problem of embeddedness. *American Journal of Sociology*, **91**(3) 481-510.

Greene, W. H. 2000. *Econometric Analysis*, 4th ed. Prentice-Hall, Upper Saddle River, NJ.

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

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.

Gu, B., P. Konana, B. Rajagopalan, and H. Chen. 2007. Competition among virtual communities: The case of investing-related communities. *Information Systems Research* **18**(1) 68-85.

Guimerà, R., B. Uzzi, J. Spiro and L. A. N. Amaral. 2005. Team assembly mechanisms determine collaboration network structure and team performance. *Science* **308**(5722) 697-702.

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

Hahn, J. and C. Zhang. 2008. Choice-based sampling and estimation of choice probabilities: Applications to information systems and electronic commerce research. *Economics, Information Systems and Electronic Commerce: Empirical Advances.* R. J. Kauffman and P. P, Tallon, eds. M. E. Sharpe Publishers, Armonk, NY, 249-270.

Hansen, M. T. 1999. The search-transfer problem: The role of weak ties in sharing knowledge across organization subunits. *Administrative Science Quarterly* **44**(1) 82-111.

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

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.

Hogg, M. A. and D. Abrams. 1988. *Social Identifications: A Social Psychology of Intergroup Relations and Group Processes*. Routledge, London.

Hong, S. J., J. Y.L. Thong, and K. Y. Tam. 2006. Understanding continued information technology usage behavior: A comparison of three models in the context of mobile internet. *Decision Support Systems* **42**(3) 1819-1834.

Howison, J. and K. Crowston. 2004. The perils and pitfalls of mining Sourceforge. A. E. Hassan, R. C. Holt, and A. Mockus, eds. *Proceedings of the 2004 Mining Software Repositories Workshop, International Conference on Software Engineering*, Edinburgh, Scotland, May 23-28, 2004, 7-11.

Ives, B. and M. Olson. 1984. User involvement and MIS success: A review of research. *Management Science* **30**(5) 586-603.

Jensen, C. and W. Scacchi. 2007. Role migration and advancement processes in OSSD projects: A comparative case study. *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*, Minneapolis, MN, 364-374.

Jeppesen, L. B. and M. J. Molin. 2003. Consumers as co-developers: Learning and innovation outside the firm. *Technology Analysis and Strategic Management* **15**(3) 363-384.

Jones, M. A., D. L. Mothersbaugh, and S. E. Beatty. 2002. Why customers stay: Measuring the underlying dimensions of services switching costs and managing their differential strategic outcomes. *Journal of Business Research* **55**(6) 441-450.

Jones, Q. and S. Rafaeli. 2000. Time to split, virtually: "Discourse architecture" and "community building" create vibrant virtual publics. *Electronic Markets* **10**(4) 214-223.

Jones, Q., G. Ravid, and S. Rafaeli. 2004. Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration. *Information Systems Research* **15**(2) 194-210.

Joyce, E. and R. E. Kraut. 2006. Predicting continued participation in newsgroups. *Journal of Computer-Mediated Communication* **11**(3) 723-747.

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.

Katz, M. L., C. Shapiro. 1995. Network externalities, competition, and compatibility. *American Economic Review* **75**(3) 424–440.

Kauffman, R., J. McAndrews, Y.-M. Wang. 2000. Opening the "black box" of network externalities in network adoption. *Information Systems Research* **11**(1) 61–82.

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.

Kogut, B. 1989. The stability of joint ventures: Reciprocity and competitive rivalry. *Journal of Industrial Economics* **38**(2) 183-198.

Koh, J., Y. Kim, B. Butler, and G. Bock. 2007. Encouraging participation in virtual communities. *Communications of the ACM* **50**(2) 68-73.

Kohanski, D. 1998. *Moths in the Machine*. St. Martin's Press, New York.

Kotlarsky, J. and I. Oshri. 2005. Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. *European Journal of Information Systems* **14**(1) 37-48.

Lakhani, K. R. and E. von Hippel. 2003. How open source software works: "free" user-to-user assistance. *Research Policy* **32**(6) 923-943.

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, B. Fitzgerald, 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, and Organization* **21**(1) 20-56.

Levine, J. M. and R. L. Moreland. 1990. Progress in small group research. *Annual Review of Psychology* **41**(1) 585-634.

Liang, K.-Y. and S. L. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* **73**(1) 13-22.

Liebowitz, S. J. and S. E. Margolis. 1994. Network externalities: An uncommon tragedy," *Journal of Economic Perspectives*, **8**(2) 133-50.

Lin, C. S., S. Wu, and R. J. Tsai. 2005. Integrating perceived playfulness into expectation-confirmation model for web portal context. *Information and Management* **42**(5) 683-693.

Lopez-Fernandez, L., G. Robles and J. M. Gonzalez-Barahona. 2004. Applying social network analysis to the information in CVS repositories. A. E. Hassan, R. C. Holt, and A. Mockus, eds. *Proceedings of the 2004 Mining Software Repositories Workshop, International Conference on Software Engineering*, Edinburgh, Scotland, May 23-28, 2004, 101-105.

Madey, G., V. Freeh and R. Tynan. 2002. The open source software development phenomenon: An analysis based on social network theory. R. D. Banker, H. Chang, and Y. Kao, eds. *Proceedings of the 8th Americas Conference on Information Systems*, Dallas, TX, August 9-11, 2002, 1806-1813.

Mansfield, E. R. and B. P. Helms. 1982. Detecting multicollinearity. *The American Statistician* **36(**3) 158-160.

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

Maute, M. F. and W. R. Forrester. 1993. The structure and determinants of consumer complaint intentions and behavior. *Journal of Economic Psychology* **14**(2) 219-247.

McClelland, D. C. 1985. How motives, skills, and values determine what people do. *American Psychologist* **40**(7) 812-825.

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

Moody, J. 2004. The structure of a social science collaboration network: Disciplinary cohesion from 1963-1999. *American Sociological Review* **69**(2) 213-238.

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, Cambridge, MA, 381-404.

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, J. M. Levine and D. M. Messick, eds. Lawrence Erlbaum Associates, Mahwah, NJ, 3-31.

Morrison, P. D., J. H. Roberts, and E. von Hippel. 2000. Determinants of user innovation and innovation sharing in a local market. *Management Science* **46**(12) 1513-1527.

Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida and Y. Ye. 2002. Evolution patterns of open-source software systems and communities. *Proceedings of the Workshop on Principles of Software Evolution, International Conference on Software Engineering*, Orlando, FL, May 19-20, 2002, 76-85.

Neter, J., M. H. Kutner, C. J, Nachtsheim, and W. Wasserman. 1996. *Applied Linear Statistical Models*. McGraw-Hill, Boston, MA.

Oliver, R. L. 1980. A cognitive model of the antecedents and consequences of satisfaction decisions. *Journal of Marketing Research* **17**(3) 460-469.

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.

Patterson, P. G., L. W. Johnson, and R. A. Spreng. 1997. Modeling the determinants of customer satisfaction for business-to-business professional services. *Journal of the Academy of Marketing Science* **25**(1) 4-17.

Petty, R. E., S. G. Harkins, K. D. Williams, and B. Latane. 1977. The effects of group size on cognitive effort and evaluation. *Personality and Social Psychology Bulletin* **3**(4) 579–582.

Podolny, J. M. 1993. A status-based model of market compensation. *American Journal of Sociology* **98**(4) 829-872.

Porter, C. E. 2006. A typology of virtual communities: A multi-disciplinary foundation for future research. *Journal of Computer-Mediated Communication* **10**(1). http://jcmc.indiana.edu/vol10/issue1/porter.html

Powell, W., D. White, J. Owen-Smith and K. W. Koput. 2005. Network dynamics and field evolution: The growth of inter-organizational collaborations in the life sciences. *American Journal of Sociology* **110**(4) 1320-1350.

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.

Prahalad, C. K. and V. Ramaswamy. 2000. Co-opting customer competence. *Harvard Business Review* **78**(1) 79-87

Preece, J, 1999. Empathic communities: Balancing emotional and factual communication. *Interacting With Computers* **12**(1) 63-78.

Rainer, A. and S. Gale. 2005. Evaluating the quality and the quantity of data on open source software projects. M. Scotto and G. Succi, eds. *Proceeding of the 1st International Conference on Open Source Systems*, Genova, Italy, July 11-15, 2005, 29-36.

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.

Ridings, C. and D. Gefen. 2004. Virtual community attraction: Why people hang out online. *Journal of Computer-Mediated Communication* **10**(1). http://jcmc.indiana.edu/vol10/issue1/ridings_gefen.html

Ridings, C., D. Gefen, and B. Arinze. 2002. Some antecedents and effects of trust in virtual communities. *Journal of Strategic Information Systems* **11**(3-4) 271-295.

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.

Robey, D. and M. Newman. 1996. Sequential patterns in information systems development: An application of a social process model. *ACM Transactions on Information Systems* **14**(1) 30-63.

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.

Sagers, G. 2007. Is bigger always better? Toward a resource-based model of open source software development communities. Doctoral dissertation. The Florida State University.

Sawyer, S., J. Farber and R. Spillers. 1997. Supporting the social processes of software development. *Information Technology and People* **10**(1) 46-62.

Sawyer, S. and P. J. Guinan. 1998. Software development: Processes and performance. *IBM Systems Journal* **37**(4) 552-569.

Scacchi, W. 2002. Understanding the requirements for developing open source software systems. *IEE Proceedings Software* **149**(1) 24-39.

Schachter, S. 1959. *The Psychology of Affiliation*. Stanford University Press, Stanford, CA.

Schegloff, E. A. 1996. Confirming allusions: Toward an empirical account of action. *American Journal of Sociology* **102**(2) 161-216.

Schellens, T. and M. Valcke. 2005. Collaborative learning in asynchronous discussion groups: What about the impact on cognitive processing? *Computers in Human Behavior* **21**(6) 957-975.

Shah, S. 2000. Sources and patterns of innovation in a consumer products field: Innovations in sporting equipment. Working Paper 4105. Massachusetts Institute of Technology, Sloan School, Cambridge, MA.

Shah, S. 2005. Open beyond software. *Open Sources 2.0: The Continuing Evolution*. D. Cooper, C. DiBona, and M. Stone, (eds.) O'Reilly Media, Sebastopol, CA, 339-360.

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.

Spreng, R. A., S. B. MacKenzie, and R. W. Olshavsky. 1996. A reexamination of the determinants of consumer satisfaction. *Journal of Marketing* **60**(3) 15-32.

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

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.

Tapscott, D. and A. D. Williams. 2006. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover, New York.

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

Tiwana, A. and A. A. Bush. 2005. Continuance in expertise-sharing networks: A social perspective. *IEEE Transactions on Engineering Management* **52**(5) 85-101.

Tuckman, B.W. 1965. Development sequence in small groups. *Psychological Bulletin* 63(6) 384-399.

Tuckman, B.W. and M. C. Jensen. 1977. Stages of small-group development revisited. *Group & Organization Management* 2(4) 419-427.

Uzzi, B., R. Guimerà, J. Spiro and L. A. N. Amaral. 2007. The emergence of self-organizing networks: Small worlds (Working Paper), Northwestern University. Evanston, IL

Uzzi, B. and J. Spiro. 2005. Collaboration and creativity: The small world problem. *American Journal of Sociology* **111**(2) 447-504.

von Hippel, E. 1994. "Sticky information" and the locus of problem solving: Implications for innovation. *Management Science* **40**(4) 429-439.

von Hippel, E. 2005. *Democratizing Innovation*. MIT Press, Cambridge, MA.

von Hippel, E. 2007. Horizontal innovation networks - By and for users. *Industrial and Corporate Change* **16**(2) 293-315.

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 Krogh, G. and E. von Hippel. 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**(7) 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.

Wang. 2007. An ecological perspective on online communities. Doctoral dissertation. University of Pittsburgh.

Wasko, M. M. and S. Faraj. 2000. "It is what one does": Why people participate and help others in electronic communities of practice. *Journal of Strategic Information Systems* **9** (2-3) 155-173.

Wasko, M. M. and S. Faraj. 2005. Why should I share? Examining social capital and knowledge contribution in electronic networks of practice. *MIS Quarterly* **29**(1) 35-57.

Wasserman, S. and J. Galaskiewicz. 1994. *Advances in Social Network Analysis*. Sage, Thousand Oaks, CA.

Watson, G. and D. Johnson. 1972. *Social psychology: Issues and insights*. J. B. Lippincott, Philadelphia, PA.

Weber, M. 1968. *Economy and Society: An Outline of Interpretive Sociology*. Bedminster Press, New York.

Weiss, M., G. Moroiu and P. Zhao. 2006. Evolution of open source communities. *Open Source Systems*. E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto and G. Succi, eds., (Vol. 203). Springer, Boston, MA, 21-32.

Wellman, B. and S. D. Berkowitz. 1998. *Social Structures: A Network Approach*. Cambridge University Press, Cambridge, UK.

Wellman, B. and M. Gulia. 1999. Virtual communities as communities: Net surfers don't ride alone. Communities in Cyberspace. M. Smith and P. Kollock, eds. Routledge, New York, NY. 163-190.

Wellman, B., J. Salaff, D. Dimitrova, L. Garton, M. Gulia, and C. Haythornthwaite. 1996. Computer networks as social networks: Collaborative work, telework, and virtual community. *Annual Review of Sociology* **22** 213-238.

Wellman, B. and S. Wortley. 1990. Different strokes from different folks: Community ties and social support. *The American Journal of Sociology* **96**(3) 558-588.

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, January 3-6, 2005.

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

APPENDICES

Appendix A. Survey of Project Initiators

A possible caveat in our research method for essay 1 is that some projects may be registered at alternative hosting sites and later ported to SourceForge.net in order to utilize the download bandwidth and project management services that provided by SourceForge.net. It is likely that members of these projects, who had been collaborating at the original hosting site, also migrated to SourceForge.net with the projects in order to continue working on the projects. As a result, the team formation process depicted by our dataset may not be accurate for these projects since their teams might have been assembled at the original hosting site. The joining events that we observed in the dataset may simply result from team members' migration to SourceForge.net with their projects instead of developers self-selecting into newly-assembled teams at SourceForge.net. In order to rule out this possibility and verify that our findings are not primarily driven by the migrated projects and developers, we conducted a survey of project initiators.

Another possible caveat is related to the bi-directional nature of OSSD team formation process, in which developers request to join projects and project administrators, who are often project initiators for newly-created projects, either approve or reject requests based on their own criteria and project needs. When developers' requests to join a project are approved by project initiators, they are formally listed as project members on the project's web site and recorded by the software crawler that was used to collect daily membership information. However, joining requests that were rejected by project initiators were not captured in our dataset. In other words, in our dataset we are unable to directly observe project initiators' filtering of developers' joining requests. Although we anticipate that project initiators may be less likely to reject others' requests to join newly-created projects because of their need to attract more developers and gain early momentum to push the projects forward, it is necessary to confirm such an expectation by surveying initiators of these projects in order to demonstrate that our dataset can reflect the actual team formation process.

Therefore, we conducted an online survey of the sample projects' initiators to further assess the bi-directional nature of the OSSD team formation process and to assess the extent of project migration from elsewhere to SourceForge.net.

A week prior to the start of the survey, we emailed the initiators of our project sample (N=2349) personalized invitations to participate in the online survey. In the invitation, we outlined the purpose of the survey and gave the respondents the option to opt out by following a link in the email invitation. All respondents were also offered an executive summary of the research findings and to be entered into a drawing to win a US$200 gift certificate. A total of 125 respondents opted out of the survey. A week after the invitation had been sent, we emailed the link to the Web-based survey to the initiators who had not opted out. Survey responses were collected from December 6 to December 16, 2007. The survey generated 384 valid responses (response rate of 16.3% (384/2349) or effective response rate of 17.3% (384 / (2349 – 125)).

The survey included both closed-ended and open-ended questions asking project administrators about the specific projects, whether the projects were ported from elsewhere to SourceForge.net, and their usual administrative practices regarding accepting and rejecting joining requests. Because many individuals are involved in multiple open source projects, the first set of questions in the survey instructed the project administrators to respond with reference to the particular OSSD project that was in our project sample. The second set of questions asked the respondents to answer with reference to their general administrative practices in all the OSSD projects that they had been in charge of.

To verify whether the projects of the survey respondents (n = 384) were representative of the broader study sample (N = 2349), we conducted two sets of analyses. As presented in Table A.1, we first compared the descriptive statistics of the variables used in our statistical model to check whether there were any substantial differences. We also checked for non-response bias by comparing the responses received in the first five days (December 6 – December 10) with those received in the last five days (December 11 – December 16) of the survey data collection. χ^2 tests of independence showed no significant differences between the early responses and the late responses, implying that non-response bias is not a problem.

We find that of the 384 initiators, 140 initiators (36.5%) stated that they had received requests to join their projects. Of these 140 initiators, only 23 (16.4%) stated that they had actually rejected joining requests submitted by other developers. In terms of their general attitude toward developers' interest in joining their projects, 41.1% expressed they were unlikely to do so; 30.6% of the respondents expressed a neutral attitude; 28.3% indicated they were likely to reject such requests. The initiators gave similar responses when asked about their attitude toward requests in the later stage and mature stage of projects. Overall, most of them were receptive of developers' requests to join, which can be illustrated by some of their comments: "volunteers are almost always welcome", "everyone should be allowed to participate in open source". When deciding whether to accept or to reject such requests, our respondents used criteria such as project team size (i.e. whether the project has enough developers), past experience with the requesting developer, the developer's skill set, past contributions, and willingness to contribute. With regard to project migration from other hosting sites to SourceForge.net, we find that although 180 projects (48.9%) were originally initiated at an alternative site, only 58 (15.1%) initiators indicated that other project members also migrated to SourceForge.net with their projects. Furthermore, according to our dataset, the joining events associated with the 58 projects constituted no more than 29% of total joining events associated with the respondents' projects, suggesting that project migration is unlikely to be the main driver of our results.

Overall, the survey results suggest that although OSSD team formation involves both developers' self-selection and project initiators' filtering of developers, the extent of the former is far greater than the extent of the latter since most initiators were quite receptive of developers' joining requests and they selectively filtered only infrequently. Therefore,

this study focuses on developers' self-selection behavior and our findings can be interpreted from developers' standpoint. Moreover, the survey results also indicate that majority of the joining events in our dataset were driven by developers' self-selection at SourceForge.net rather than project members migrating with projects initially founded elsewhere.

Table A.1 Descriptive Statistics – Projects Associated with Survey Respondents vs. Sample Projects in Essay 1

	Projects Associated with Survey Respondents		**Sample Projects in Essay 1**	
Variable	**Mean**	**St. Dev**	**Mean**	**St. Dev**
Join	0.13	0.340	0.17	0.377
$InitiatorTie_{Outcome}$	0.00	0.075	0.00	0.117
$InitiatorTie_{Strength}$	0.00	0.107	0.02	0.611
$MemberTie_{Outcome}$	0.00	0.014	0.00	0.120
$MemberTie_{Strength}$	0.00	0.014	0.02	0.349
InitiatorTieAmount	0.38	0.981	0.48	1.058
MemberTieAmount	0.19	0.721	0.30	0.977
InitiatorActiveTime	17.95	13.663	18.03	13.780
MemberActiveTime	4.27	8.105	5.65	10.455
InitiatorProjects	0.39	0.609	0.40	0.596
MemberProjects	0.07	0.260	0.10	0.361
DomainPopularity	0.07	0.087	0.06	0.088
ActivityPercentile	2.27	0.541	2.19	0.489
DescriptionLength	5.06	0.591	4.97	0.633
AcceptDonation	0.11	0.313	0.08	0.270
CodeReleased	0.04	0.195	0.03	0.161
TeamSize	0.28	0.493	0.40	0.683
DeveloperActiveTime	28.00	20.684	27.73	20.684
DomainMatch	0.02	0.133	0.02	0.131
ProgrammingLanguageMatch	0.08	0.255	0.06	0.236
AudienceMatch	0.17	0.364	0.13	0.322
OperatingSystemMatch	0.11	0.307	0.08	0.267
ProjectAge	3.26	0.512	3.25	0.514
PreexistingCode	0.01	0.084	0.01	0.089
Sample Size (*N*)	384		2349	

Appendix B. Robustness Checks for Essay 1

This appendix presents the results of the robustness checks conducted for essay 1.

In the study when computing the tie measures for the projects that have multiple non-initiator members, we aggregated the measures by summing the values for all developer-member pairs. As a robustness check, we also used other aggregate measures such as maximum (for best experience) and minimum (for worst experience) to compute the tie measures between developers and non-initiator members. Results of the sensitivity analysis presented in Table B.1 suggested that the findings are robust to different ways of aggregating these measures when there are multiple non-initiator members in the project.

Table B.1 Maximum and Minimum Used as Aggregate Measures to Compute Tie Measures Between Developers and Non-Initiator Members

Aggregate Measure Used in Factor Analysis of Ties	**Sum (Same as Table 2.1)**	**Maximum**	**Minimum**
Variable	**Parameter Estimate**	**Parameter Estimate**	**Parameter Estimate**
Constant	-8.119***	-7.843***	-7.802***
$InitiatorTie_{Outcome}$	-0.027	-0.030	-0.045
$InitiatorTie_{Strength}$	0.708**	0.711**	0.843**
$MemberTie_{Outcome}$	0.985	0.942	0.712
$MemberTie_{Strength}$	0.367	0.372	0.055
InitiatorTieAmount	-0.190	-0.194	-0.191
MemberTieAmount	-0.384**	-0.391**	-0.391***
InitiatorActiveTime	-0.014	-0.015	-0.015
MemberActiveTime	0.027***	0.026***	0.026**
InitiatorProjects	-0.009	-0.006	-0.007
MemberProjects	0.678**	0.687**	0.694***
DomainPopularity[a]	0.569	0.693	0.696
$ActivityPercentile_{Low}$[b]	-0.974***	-1.089***	-1.105***
$ActivityPercentile_{Mid}$[b]	-0.099	-0.246	-0.263
DescriptionLength	0.091	0.090	0.088
AcceptDonation	0.655**	0.673**	0.675**
CodeReleased	1.280**	1.415**	1.430**
TeamSize	2.550***	2.559***	2.556***
DeveloperActiveTime	-0.042***	-0.043***	-0.042***
DomainMatch	-0.093	-0.088	-0.087
ProgrammingLanguageMatch	-0.142	-0.144	-0.147
AudienceMatch	-1.077***	-1.084***	-1.089***
OperatingSystemMatch	-0.774**	-0.778**	-0.782**
ProjectAge	-1.782***	-1.820***	-1.826***
PreexistingCode	0.093	0.006	-0.054
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$			

In the study a categorical variable *ActivityPercentile* was included in the analysis to distinguish among low, medium, and high level of activity for the project. We further checked the robustness of this operationalization by using nominal scale for this measure. Table B.2 showed that the results are similar to those of the original model where *ActivityPercentile* is a categorical variable.

Table B.2 *ActivityPercentile* as a Continuous Variable

Model	**ActivityPercentile as a Categorical Variable (Same as Table 2.1)**	**ActivityPercentile as a Continuous Variable**
Variable	**Parameter Estimate**	**Parameter Estimate**
Constant	-8.119***	-8.766***
$InitiatorTie_{Outcome}$	-0.027	-0.030
$InitiatorTie_{Strength}$	0.708**	0.709**
$MemberTie_{Outcome}$	0.985	0.908
$MemberTie_{Strength}$	0.367	0.375
InitiatorTieAmount	-0.190	-0.196
MemberTieAmount	-0.384**	-0.394***
InitiatorActiveTime	-0.014	-0.015
MemberActiveTime	0.027***	0.026***
InitiatorProjects	-0.009	0.002
MemberProjects	0.678**	0.692**
DomainPopularity[a]	0.569	0.683
$ActivityPercentile_{Low}$[b]	-0.974***	
$ActivityPercentile_{Mid}$[b]	-0.099	
ActivityPercentile		0.011**
DescriptionLength	0.091	0.085
AcceptDonation	0.655**	0.664**
CodeReleased	1.280**	1.397**
TeamSize	2.550***	2.563***
DeveloperActiveTime	-0.042***	-0.042***
DomainMatch	-0.093	-0.087
ProgrammingLanguageMatch	-0.142	-0.146
AudienceMatch	-1.077***	-1.085***
OperatingSystemMatch	-0.774**	-0.774**
ProjectAge	-1.782***	-1.827***
PreexistingCode	0.093	-0.050
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$		

In essay 1, when constructing the dataset we sampled only the developers with past project experience on SourceForge in order to be able to compute the matching variables based on the properties of their past projects. In order to test the robustness of the results associated with the restriction in sampling developers, we constructed a new set of samples by randomly sampling all developers who have registered at SourceForge regardless of their past project experience. In the regression analysis the four matching variables and *DeveloperActiveTime* had to be dropped but we included *DeveloperProjectExperience*.

As shown in Table B.3, *DeveloperProjectExperience* did not have a significant effect on the joining behavior. In addition, our major findings related to $InitiatorTie_{Strength}$, *MemberActiveTime, MemberTieAmount,* and *MemberProjects* held when this alternative sampling pool of developers was used.

Table B.3 All Developers at SourceForge

Model	**Developers with Past Project Experience (Same as Table 2.1)**	**All Developers**
Variable	**Parameter Estimate**	**Parameter Estimate**
Constant	-8.119***	-9.142**
$InitiatorTie_{Outcome}$	-0.027	0.234
$InitiatorTie_{Strength}$	0.708**	1.078***
$MemberTie_{Outcome}$	0.985	-0.439
$MemberTie_{Strength}$	0.367	-0.500
InitiatorTieAmount	-0.190	-0.146
MemberTieAmount	-0.384**	-0.458***
InitiatorActiveTime	-0.014	-0.017
MemberActiveTime	0.027***	0.022***
InitiatorProjects	-0.009	-0.123
MemberProjects	0.678**	1.455***
$DomainPopularity^{a}$	0.569	-0.817
$ActivityPercentile_{Low}{}^{b}$	-0.974***	-0.078
$ActivityPercentile_{Mid}{}^{b}$	-0.099	0.872**
DescriptionLength	0.091	0.152
AcceptDonation	0.655**	0.601***
CodeReleased	1.280**	1.258***
TeamSize	2.550***	2.469***
DeveloperActiveTime	-0.042***	
DeveloperProjectExperience		0.179
DomainMatch	-0.093	
ProgrammingLanguageMatch	-0.142	
AudienceMatch	-1.077***	
OperatingSystemMatch	-0.774**	
ProjectAge	-1.782***	-1.810***
PreexistingCode	0.093	-0.260
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$		

When computing the fit between developers' skills/interest and the focal project's requirements, we adopted the dominant matching approach. Since developers' skills and interest may change over time, we also constructed the matching variables using their most recent project experience. Table B.4 showed a comparison between the results when the dominant and the recent matching variables were used. It suggested that similar results were obtained when we used the most recent matching scores.

Table B.4 Recent Matching Scores

Model	**Dominant Matching (Same as Table 2.1)**	**Recent Matching**
Variable	**Parameter Estimate**	**Parameter Estimate**
Constant	-8.119***	-8.725***
$InitiatorTie_{Outcome}$	-0.027	-0.031
$InitiatorTie_{Strength}$	0.708**	0.685**
$MemberTie_{Outcome}$	0.985	1.359
$MemberTie_{Strength}$	0.367	0.394
InitiatorTieAmount	-0.190	-0.154
MemberTieAmount	-0.384**	-0.467***
InitiatorActiveTime	-0.014	-0.015
MemberActiveTime	0.027***	0.024**
InitiatorProjects	-0.009	-0.067
MemberProjects	0.678**	0.670*
$DomainPopularity^{a}$	0.569	0.695
$ActivityPercentile_{Low}{}^{b}$	-0.974***	-0.965***
$ActivityPercentile_{Mid}{}^{b}$	-0.099	-0.089
DescriptionLength	0.091	0.109
AcceptDonation	0.655**	0.630*
CodeReleased	1.280**	1.514**
TeamSize	2.550***	2.624***
DeveloperActiveTime	-0.042***	-0.051***
DomainMatch	-0.093	-0.392
ProgrammingLanguageMatch	-0.142	-0.960
AudienceMatch	-1.077***	-1.065**
OperatingSystemMatch	-0.774**	-0.735*
ProjectAge	-1.782***	-1.863***
PreexistingCode	0.093	-0.005
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$		

A large number of sample projects did not attract any additional developers during the observation period, which may be due to the fact that some projects are not intended for others to join. For example some projects may be intended to be single-developer projects by their initiators; some project may be founded as class projects that do not require contributions from developers outside the class. So we conducted the analysis with only those projects that had successfully attracted developers (*N*=520). Overall, Table B.5 showed that the major findings are consistent with those for the full sample, suggesting that they are unlikely to be driven by the projects whose initiators have no intention of attracting more developers. Furthermore, Table B.6 suggested that the domain distribution of these 520 projects is consistent with that of the full project sample.

Table B.5 Full Sample vs. Sub-Sample (Projects with at Least One Joiner)

Model	**Full Sample (N=2349)**	**Projects That Attracted At Least One Developer (N=520)**
Variable	**Parameter Estimate**	**Parameter Estimate**
Constant	-8.119***	-7.366***
$InitiatorTie_{Outcome}$	-0.027	-0.323
$InitiatorTie_{Strength}$	0.708**	0.961***
$MemberTie_{Outcome}$	0.985	-0.046
$MemberTie_{Strength}$	0.367	0.315
InitiatorTieAmount	-0.190	-0.147**
MemberTieAmount	-0.384**	-0.164***
InitiatorActiveTime	-0.014	-0.011
MemberActiveTime	0.027***	0.010***
InitiatorProjects	-0.009	0.133
MemberProjects	0.678**	0.405**
$DomainPopularity^{a}$	0.569	0.907*
$ActivityPercentile_{Low}{}^{b}$	-0.974***	0.067
$ActivityPercentile_{Mid}{}^{b}$	-0.099	0.984***
DescriptionLength	0.091	0.058
AcceptDonation	0.655**	0.554***
CodeReleased	1.280**	1.418***
TeamSize	2.550***	1.757***
DeveloperActiveTime	-0.042***	-0.035***
DomainMatch	-0.093	0.050
ProgrammingLanguageMatch	-0.142	0.206
AudienceMatch	-1.077***	-0.613***
OperatingSystemMatch	-0.774**	-1.447***
ProjectAge	-1.782***	-1.864***
PreexistingCode	0.093	-0.195
Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$		

Table B.6 Domain Distribution of Sample Projects and Projects with Joiners

Category	Projects in Sample	Projects w/ Developer Joining	Percentage (%)
Software Development	276	70	25.4
Internet	245	57	23.3
System	206	38	18.44
Communications	166	51	30.7
Games/Entertainment	161	53	33.0
Scientific/Engineering	154	44	28.6
Multimedia	136	32	23.5
Office/Business	111	28	25.2
Database	66	18	27.3
Formats and Protocols	62	13	21.0
Education	48	9	18.8
Security	42	11	26.2
Desktop Environment	29	9	31.0
Text Editors	25	3	12.0
Other	17	4	23.5
Printing	8	4	50.0
Terminals	6	1	16.7
Religion and Philosophy	2	1	50.0
Sociology	1	0	25.4
Notes: The total number of projects is 1358, which includes only those projects that have defined its domain during our data collection period.			

In the analysis we controlled for project popularity,which may be partially determined by the software domain. Some software domains may be more popular than others, increasing their attractiveness to potential developers. We controlled for domain popularity using the ratio *DomainPopularity*. We further checked the robustness of our findings by including dummy variables representing whether a project belongs to the top seven domains from Table 2.3. The results are presented in Table B.7. None of these variables has a significant effect on whether developers joined a project, suggesting that no particular domains are more attractive to developers than others, consistent with our finding that *DomainPopularity* does not seem to influence developers' joining decisions.

Table B.7 Domain Dummies Used Instead of *DomainPopularity*

	Original Model (*DomainPopularity*)	**Modified Model (Domain Dummies)**
Variable	**Parameter Estimate**	**Parameter Estimate**
Constant	-7.843***	-7.83***
$InitiatorTie_{Outcome}$	-0.030	-0.092
$InitiatorTie_{Strength}$	0.711**	0.703**
$MemberTie_{Outcome}$	0.942	0.766
$MemberTie_{Strength}$	0.372	0.338
InitiatorTieAmount	-0.194	-0.203
MemberTieAmount	-0.391**	-0.405**
InitiatorActiveTime	-0.015	-0.017
MemberActiveTime	0.026***	0.027***
InitiatorProjects	-0.006	0.064
MemberProjects	0.687**	0.73**
DomainPopularity[a]	0.693	
$ActivityPercentile_{Low}$[b]	-1.089***	-1.122***
$ActivityPercentile_{Mid}$[b]	-0.246	-0.244
DescriptionLength	0.090	0.104
AcceptDonation	0.673**	0.679**
CodeReleased	1.415**	1.438**
TeamSize	2.559***	2.660***
DeveloperActiveTime	-0.043***	-0.044***
DomainMatch	-0.088	
ProgrammingLanguageMatch	-0.144	-0.183
AudienceMatch	-1.084***	-1.077***
OperatingSystemMatch	-0.778**	-0.822**
ProjectAge	-1.820***	-1.868***
PreexistingCode	0.006	0.044
Domain_commu		0.016
Domain_games		0.401
Domain_internet		-0.198
Domain_multimedia		-0.285
Domain_other		-0.086
Domain_scientific		0.436
Domain_software		0.030
Domain_system		0.478
Significance levels: *** 0.01, ** 0.05, * 0.1		

We also conducted additional sensitivity analyses by including interaction terms of *ProjectAge* with other variables of interest. The rationale is that if migration was driving the results, we would see significant interaction effects – in other words, the effect of some variable (e.g., prior collaborative ties) would be more significant for early joiners (i.e., developers who used to be team members of a project that was initially hosted elsewhere and later ported to SourceForge.net followed the project to SourceForge.net and joined the team again) compared to later joiners. The results are presented in Table B.8.

The interaction between *ProjectAge* and the tie variables (H1, H2) were not significant (See Model C). So the impact of collaborative tie did not seem to be different for late vs. early joiners (i.e., potential migrators).

When we included the interaction between *ProjectAge* and *TeamSize* (Model D) – with the rationale that if non-migrating actual joiners (i.e., late joiners; or for larger values of *ProjectAge*) are more concerned with the existence of a critical mass of developers in the project then we would see a significant (positive) coefficient for the interaction term (*ProjectAge* x *TeamSize*) – the results showed that while the main effect of *TeamSize* was positive (and significant, $p < 0.01$), the interaction effect was in fact negative (and significant, $p < 0.05$) suggesting that the impact of *TeamSize* is smaller for later joiners. So again, the migration effects do not seem to be explaining the results.

Finally, we also included the interaction between *ProjectAge* and *CodeReleased* (see Model E) with the rationale that if migration effects were driving the results, early joiners (i.e., migrators) would care less about the release of code whereas late joiners (i.e., actual joiners) would care more about the existence of code; in other words we would see a significant (positive) interaction effect with an insignificant main effect. The results showed that the coefficient of the interaction (*ProjectAge* x *CodeReleased*) was only marginally positive ($p < 0.1$) whereas the main effect of *CodeReleased* was positive ($p < 0.01$).

Overall, the results indicate that while we cannot preclude the existence of project migration, this is not the main driver of the results. Rather, our hypothesized effects of prior collaborative ties are still significant and meaningful.

Table B.8 Interaction Effect Between *ProjectAge* and Other Variables

Variable	**Parameter Estimate**						
Model	**Original**	**A**	**B**	**C**	**D**	**E**	**F**
Constant	-8.119***	-13.870***	-14.086***	-13.891***	-12.715***	-13.976***	-12.734***
(1) $InitiatorTie_{Outcome}$	-0.027	-0.093	-0.096	-0.218	-0.034	-0.035	-0.191
(2) $InitiatorTie_{Strength}$	0.708**	0.833**	0.800***	0.859**	0.701**	0.708**	0.848**
(3) $OtherTie_{Outcome}$	0.985	0.882	3.168	3.413	0.996	0.973	3.512
(4) $OtherTie_{Strength}$	0.367	0.567	0.573	0.680	0.380	0.377	0.694
(5) *InitiatorTieAmount*	-0.190	-0.189	-0.174	-0.197	-0.170	-0.194	-0.182
(6) *OtherTieAmount*	-0.384**	-0.380**	-0.385**	-0.383**	-0.308*	-0.380**	-0.305*
(7) *InitiatorExpActiveT*	-0.014	-0.015	-0.018	-0.015	-0.019	-0.014	-0.019
(8) *OtherExpActiveT*	0.027***	0.027***	0.026***	0.026***	0.024***	0.026***	0.024**
(9) *InitiatorExpP*	-0.009	-0.012	0.012	-0.001	0.038	-0.015	0.038
(10) *OtherExpP*	0.678**	0.665**	0.660**	0.649*	0.526	0.674**	0.501
(11) *DomainPopularity*	0.569	0.611	0.612	0.629	0.849	0.689	1.000
(12) $ActivityPercentile_{Low}$	-0.974***	-0.979***	-0.959**	-0.967***	-0.760	-0.954**	-0.741
(13) $ActivityPercentile_{Mid}$	-0.099	-0.103	-0.071	-0.081	-0.049	-0.051	0.006
(14) *DescDetail*	0.091	0.087	0.104	0.087	0.087	0.092	0.081
(15) *AcceptDonation*	0.655**	0.656**	0.611**	0.672**	0.675**	0.653***	0.688**
(16) *CodeReleased*	1.280**	1.280**	1.290**	1.284**	1.439**	1.831**	1.840***
(17) *TeamSize*	2.550***	2.557***	2.564***	2.570***	2.424***	2.557***	2.445***
(18) *DeveloperActiveT*	-0.042***	-0.043***	-0.041***	-0.043***	-0.043***	-0.043***	-0.044***
(19) *DomainMatch*	-0.093	-0.089	0.446	-0.095	-0.043	-0.086	-0.030
(20) *PLMatch*	-0.142	-0.151	-0.146	-0.220	-0.139	-0.142	-0.215
(21) *AudienceMatch*	-1.077***	-1.089***	-1.125***	-1.105***	-1.117***	-1.092***	-1.153***
(22) *OSMatch*	-0.774**	-0.784**	-0.533**	-0.811**	-0.801**	-0.785**	-0.846*
(23) *ProjectAge*	-1.782***	-1.793***	-1.794***	-1.808***	-1.003***	-1.898***	-1.164***
(24) *PreexistingCode*	0.093	0.096	0.046	0.123	0.182	0.097	0.211
(23)×(1)		-0.618		-1.666			-1.602
(23)×(2)		0.539		0.327			0.271
(23)×(3)			12.450	13.949			14.708
(23)×(4)			0.393	0.380			0.565
(23)×(17)					-0.974**		-0.928**
(23)×(16)						1.961*	1.636*

Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$

Notes: Model A includes interaction between *ProjectAge* and tie with project initiators ($InitiatorTie_{Outcome}$ and $InitiatorTie_{Strength}$); Model B includes interaction between *ProjectAge* and ties with other developer members already in the project ($OtherTie_{Outcome}$ and $OtherTie_{Strength}$); Model C combines Model A and B; Model D includes interaction between *ProjectAge* and *TeamSize*; Model E includes interaction between *ProjectAge* and *CodeReleased*; finally, Model F includes all interactions above to check robustness of parameter estimates across models.

Appendix C. Coding Scheme for Message Content Analysis in Essay 2

In essay 2, we distinguish among three levels of member involvement in the OSSD process, namely pre-usage, usage, and modification. Then we identify different types of activities associated with these levels of involvement. Pre-usage involvement is associated with general questions about software/project and request for help related to download, setup, configuration, and compilation of the software. Usage involvement is related to request for information about software details, feature suggestion with no implementation details, and problem report with no suggestions about solutions. Modification involvement is associated with activities such as contributing code and asking questions about modifying code. In order to code the level of involvement based on the sample messages, we developed a coding scheme as follows.

1. **Request for general resources about the software (such as documentation and technical articles) or general question about the software (such as hardware and software compatibility).** The main purpose of the request is to seek extra materials that can help users become familiar with the software and evaluate whether the software is appropriate for potential adopter's need.

   Sample statements:
   - "Is there any documentation on how to …?"
   - "I don't really understand a lot of the terminology used throughout the documentation. Could someone refer me to a simple primer on all this stuff?"
   - "I'm a newbie of the project. I've found XXX but not the documentation about it. Where can I find it?"
   - "Is there separate documentation for XXX? Or do we have to rely on the examples for understanding the usage?"

2. **Request for help related to download, setup, configuration, and compilation of the software.** The main purpose of asking the question is to seek technical support in order to properly install the software.

   Sample statements:
   - "I downloaded the file but my computer is unable to run it. What program opens the file?"
   - "I have installed XXX package from XXX site. Can someone help me how to configure XXX for my clients websites."
   - "I downloaded the XXX library but can't get this to build using XXX."
   - "Could anyone tell me where I can download the latest version of XXX?"

- "I can't compile because XXX is missing. Where are the files needed for XXX to work?"

3. **Request for information about software details.** It usually represents a question that users have when using the software and/or studying the code. These questions are more specific than requests for resources. They are typically about one particular feature of the software or one particular function in the code. The main purpose of the question is to enhance users' understanding of the software in greater detail or to help users utilize the software more effectively.

   Sample statements:
   - "I searched all examples for a simple feature … Is there such a feature and how do i use it?"
   - "Can someone provide detailed instructions on how to …?"
   - "What should I do to …?"
   - "How could one go about replacing the XXX to XXX?"
   - "I'm sure this exists somewhere in the code but I couldn't find it. Is there a method to convert XXX to XXX?"
   - "What would be the best technique to perform this operation on image? Some links with sample code would be appreciated."
   - "This seems to be a simple problem, but I have no idea how to … I have been working around this by … Could someone post a few short lines of code that will show me how to …?"
   - "Would you suggest where I could look to find out how to turn … into …?"
   - "I want to use XXX for creating jbig2 files. What do I have to do with the chunk data to get a normal jbig2 image file?"

4. **Feature suggestion with no details on how to implement it.** It represents a feature desired by the user but **has not been implemented** in the existing code. The main purpose of this request is to enhance the functionality of the software so that it caters to the needs of a larger user base.

   Sample statements:
   - "It'd be nice if XXX could support …"
   - "It would be nice if I could add an entity and … just for this entity, but do it manually for others. Is this a planned feature?"
   - "Is there a way to synchronize two separate XXX databases that each contains some different data as well as some of the same data? This would be some type of compare and synchronize function."
   - "One feature you could include in the next version of XXX is to …"
   - "What about adding XXX to the XXX page to allow …?
   - "I would really like to have a feature in XXX: When playing a video, I feel that … would be wonderful"

5. **Problem report with no suggestions on how to fix it.** The problem is caused either by users' lack of understanding of the software or by bugs in the code.

   Sample statements:
   - "I'm experiencing problems with XXX. Sometimes the changes do not take effect (for example …). Have you any clue about the cause of these behavior?"
   - "I have found this problem in XXX versions under Windows XP … "
   - "I can't seem to get the XXX to print in any decent size. Is this a bug or is it a problem with my system?"
   - "I've been having an irritating problem with XXX crashing that I believe is directly related to the XXX plug-in, but I'm at a loss as to why. Hopefully somebody here might be able to tell me what the solution is..."

6. **Code contribution.** It can be a problem report together with how to fix it, a feature request together with implementation details, and code of plug-ins or other software components.

   Sample statements:
   - "XXX is VERY slow with memory allocations. So I looked in the code and changed the following code … "
   - "My theme is very slow. I think is better to do a new function in XXX that … I have made a proof of concept test modifying the code and work great! I will post the diff in this forum when I clean up the code. I hope to see this patch on the next version of XXX."
   - "Here is a proposal for using … determine database character storage lengths. This can be implemented using the proposed code change below. It applies to both the 1.3 and 1.4 versions of XXX."
   - "I've found that I need this function regularly. I'm not sure it is optimized for all situations but it works for me. If you agree, you could add this function to the XXX namespace. Here is its code: …"

7. **Questions about extending or modifying the existing code.** The individual has the intent to modify the code or write his own code to enhance or extend the software.

   Sample statements:
   - "I had been searching for docs on how to make and apply patches for our XXX System. Kindly give me link on where I can find that doc, so I could make my own patch"
   - "I've made a Spanish translation for XXX. Where or how can I publish on the official site?"
   - "From what I read on the web and from the XXX documentation there seems to be neither a standard way to output to multiple files nor an extension for it.

Before I try and write an extension can someone point me to the existing way of doing it please?"

- "Hi, I am trying to find a way to modify a file to … I tried the following script: … but nothing happened. Do you have any suggestions?"
- "Hi, I'm going to write an XXX which can do the following: … Are you interested in this? Would you add this to your next release? You can contact me at …"
- "I am attempting to demonstrate a simple example of a C# client using XXX to communicate with XXX. When I attempt to run the client, it throws the following exception: … The exception is thrown by the following line of code: … "
- "I need to use XXX to take advantage of the alpha channel. It might be a nice addition to add some "alpha" support. I'm thinking something like: … This function would just call … for each pixel. If you agree that this would be worthwhile, I'll write the function."

8. **Messages that do not fall into any of the categories listed above.** They could be announcement, seeking advice not particularly related to the software being developed, responses to questions raised earlier, sharing project-related information, etc.

Appendix D. Robustness Check for Essay 2

We recognize that OSSD participants are encouraged to post a message containing only one subject or question to improve the message clarify and to facilitate the community in understanding and responding to it. Hence, when some members have multiple requests that they need answers for, they tend to post multiple messages to initiate a discussion for each of their concerns within a very short period of time. To distinguish between postings representing continued participation and posting representing almost simultaneous postings of multiple questions, we adopted a cut-off value of 1 hour between two postings by the same individual. In other words, if he posts two messages within 1 hour, we treat his participation as two simultaneous postings rather than continued participation.

As shown in Table D.1, overall, when alternative threshold values such as 10 minutes, 20 minutes, 40 minutes, 2 hours and 3hours are used, the results are similar to those when the threshold value is 1 hour. As we increased the cut-off value for the interval between consecutive postings, *Response* became a significant and positive predictor when the cut-off value was 20 minutes or higher. Also, as the threshold increased the magnitude of the impact of Response also increased.

Table D.1 Survival Analysis with Different Cut-Off Values of Time Interval Between Two Consecutive Postings

Variable	**10 minutes**	**20 minutes**	**40 minutes**	**1 hour**	**2 hours**	**3 hours**
Response	0.003	0.048	0.098*	0.116**	0.145***	0.160***
Preusage	-0.297**	-0.344**	-0.332**	-0.333**	-0.433***	-0.410**
Modification	0.092	0.106	0.106	0.090	0.057	0.023
Response×*Preusage*	0.018	0.068	0.057	0.061	0.162	0.148
Response×*Modification*	-0.098	-0.121	-0.133	-0.117	-0.082	-0.050
PriorReturn	0.069***	0.070***	0.070***	0.070***	0.071***	0.071***
MessageLength	-0.003	-0.006	-0.000	0.003	0.007	0.007
PriorReturn	-0.144	-0.147	-0.165	-0.176	-0.167	-0.154
OtherProjectMembership	0.008	0.014	0.019	0.018	0.017	0.018
Downloads	0.038***	0.040***	0.042***	0.042***	0.045***	0.046***
ProjectAge	-0.002	-0.002	-0.001	-0.001	-0.001	-0.001
TargetDeveloper	-0.409***	-0.403***	-0.400***	-0.396***	-0.408***	-0.410***
RestrictiveLicense	-0.605***	-0.609***	-0.606***	-0.607***	-0.617***	-0.620***
ProjectStage	-0.106***	-0.104***	-0.105***	-0.107***	-0.100***	-0.103***
Number of Censored Observations	3289	3321	3354	3367	3404	3427
χ^2	706.162***	712.580***	714.660***	713.656***	727.585***	726.944***