ABSTRACT

Title of Dissertation: ABSORPTIVE CAPACITY AND OPEN SOURCE SOFTWARE PROJECT PERFORMANCE

Sherae Lee Daniel, Doctor of Philosophy 2007

Dissertation directed by: Ritu Agarwal
Decision and Information Technologies

Katherine Stewart
Decision and Information Technologies

The open source phenomenon is an exciting movement that is transforming traditional forms of software development. Some open source software (OSS) projects, such as Linux and Apache, are performing extremely well and rapidly replacing proprietary software in major corporations and governments. In addition to these highly publicized examples, there are legions of OSS projects that have not experienced a similar uptake.

The purpose of this dissertation is to understand how and why some OSS projects are able to perform better than others. It explores antecedents of OSS project performance from a knowledge-focused perspective because software development is a knowledge-intensive activity. In particular, it examines the development and effects of absorptive capacity for an OSS project. Absorptive capacity captures the degree to which an organization is able to acquire and assimilate knowledge. In describing how OSS absorptive capacity is developed, this dissertation identifies characteristics and behaviors of project participants that indicate an OSS project's absorptive capacity. I underscore the importance of the characteristics and behaviors of two different sets of project participants in an OSS project: those in the Internet-based user community and those in the development group. To the extent that absorptive capacity influences OSS project performance, I argue that these characteristics and behaviors are critical for OSS project performance. Archival data about OSS projects that use the SourceForge platform are used to empirically test the model developed.

This dissertation makes several contributions to theory and practice. The research informs project managers regarding the participants to target and behaviors to encourage that will lead to superior performance for their OSS project. In exploring the effect of absorptive capacity in an OSS project, this dissertation adds to the absorptive capacity literature by examining the interaction of two dimensions of this construct: knowledge acquisition and knowledge transfer. Finally, this dissertation extends the OSS literature by specifically exploring the effect of the Internet-based user community on OSS project performance.

ABSORPTIVE CAPACITY AND OPEN SOURCE SOFTWARE PROJECT PERFORMANCE

by

Sherae L. Daniel

Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy
2007

Advisory Committee:

Professor Ritu Agarwal, Co-Chair
Professor Katherine Stewart, Co-Chair
Professor Anand Gopal
Professor Gordon Gao
Professor Jennifer Preece

ACKNOWLEDGEMENTS

I am grateful to many people. First I would like to thank my advisers, Dr. Ritu Agarwal and Dr. Katherine Stewart. They are both strong and incredibly intelligent women. They posses many characteristics that I would feel privileged to acquire and it has been my honor to be their student and friend.

Second, I would like to thank those who provided me with a strong spiritual and emotional foundation. First are my parents who made phenomenal choices and sacrifices that make all I do possible. Thanks also to them for telling me I could achieve anything I wanted because I was as intelligent as anyone else, and if it didn't seem that way the others were cheating! Similarly, my grandparents have consistently gone beyond the call of duty to support me in my pursuits.

More broadly I thank my friends, mentors, extended and church families. Whether by listening when I called to rant about my newest hurdle, by reminding me of my beauty and intelligence or by offering a hug I appreciate each of them. I especially want to acknowledge the members of Mount Rise Baptist Church, Second Baptist Church, and especially my new friends on the Reid Temple Usher Board.

The third set of people I'd like to thank is those who led by example and achieved greatness or over came difficult situations through hard work and determination. In particular I want to acknowledge those who share my race or sex; it is because of them that I believed I could do it.

Finally, I'd like to thank those who offered help as I developed my dissertation research. For writing expertise I would like to thank Omari Daniel, Elizabeth Murphy and Susan Grodsky. For statistical expertise I would like to thank Aref Dajani and Eric Slud. For technical support I'd like to thank Chang-Han Jong. For constructive feedback I would like to thank my dissertation committee. A better team is hard to find!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 OSS Development

The open source phenomenon is an exciting movement that is transforming traditional software development practices. Open source software (OSS) is typically characterized as software that is developed by volunteers, available for modification by anyone and available to users for free. Large software developers, such as Netscape, have begun to experiment with open source development practices, and IBM has opened some of its code for public development. The actions of these large computing organizations are partially a result of the accelerating and widespread adoption of OSS. Both industry and government are beginning to rely on OSS, especially the flagship examples, such as Linux and Apache, for critical infrastructures (Perkins 1999). A recent study reported that approximately 45% of all mid-sized U.S. businesses had adopted or were experimenting with Linux, and it was predicted to be the fastest growing desktop operating system through the end of 2007 (Mauri 2004). Several national governments have adopted OSS platforms for computing, including the United States of America, Brazil and China (http://news.com.com/2100-1001-272299.html?legacy=cnet). Because of this recent surge of interest in OSS, identification of the antecedents of OSS performance is gaining practical importance.

In contrast to the flagship examples described above, many OSS projects exhibit comparatively little sign of performance (Krishnamurthy 2002). This dissertation examines a sample of OSS projects in order to understand what enables an OSS project to perform better than others. To facilitate an understanding of OSS project performance, a theoretical perspective grounded in the concept of knowledge is utilized because software development is a knowledge-intensive activity (Robillard 1999). Software development requires developers to have high levels of multiple types of knowledge. Developers must have rich technical knowledge about programming languages and software engineering procedures, combined with an awareness of how the code is implemented in a given development project. Furthermore, they must have high levels of domain knowledge or knowledge related to the context where the application will be used.

Because of the importance of knowledge in OSS projects, I seek to illuminate the effects of absorptive capacity, the ability to acquire and use knowledge, in OSS projects. Absorptive capacity, in particular, is leveraged in this dissertation because prior research has suggested that it yields superior performance for organizations that innovate in dynamic environments (Zahra et al. 2002). OSS development is fundamentally an innovation process (Krogh et al. 2003; Von Hipple et al. 2003), and rapid technology changes make the software industry a dynamic environment (Iansiti et al. 1997). Software developed by an OSS project may be quickly outdated by a newer technology forcing the OSS development group to solve new software problems rapidly to remain useful. To the degree that performance outcomes in a dynamic environment require an ability to rapidly acquire and assimilate knowledge, an OSS project's performance depends on its absorptive capacity. This dissertation seeks to answer the two related questions:

How can absorptive capacity be developed in an OSS project?

How does an OSS project's absorptive capacity affect an OSS project's performance?

Chapter 1 continues by defining the OSS project and discussing the dimension of OSS project performance that is the focus of this research. It concludes with a summary of the goals and expected contributions of this research.

1.2 Delineation of Research Phenomenon: The OSS Project

The subject of this dissertation is the OSS project. I define the OSS project as the development of a single software application under the restrictions of an OSI-approved license. OSI licenses require that the source code be available at little or no charge, redistribution of the compiled application must be allowed at no fee, distributions of modified versions of the source code must be available without discrimination toward different types of users and the distributions of modified versions must be allowed on the same terms as the original source code. Each OSS project has at least one developer associated with it, but may have an unlimited number of other developers and users associated with it. This definition does not require the developers to be volunteers, and some OSS developers are paid by corporations. For a detailed description of an OSS project see Fitzgerald's description (2006).

1.3 OSS Project Performance

This section briefly reviews commonly adopted measures of OSS project performance and then discusses why this research focuses on OSS project survival and development group activity intensity. It concludes with a description of how this dissertation conceptualizes development group activity.

There are many ways to measure OSS project performance. Some OSS project performance measures are drawn from traditional software performance and include software satisfaction, use and adoption (DeLone et al. 1992). An example of research using this kind of measure for OSS project performance is the paper by Grewal et al. (2006). Grewal et al. (2006) examine a performance measure directly related to use as they explore the effect of OSS project network embeddedness on commercial success measured as the number of times an OSS application is downloaded. Similarly, Crowston and Scozzi (2002) explore the effect of available competencies on downloads and page views. In addition to these kinds of measures that parallel traditional software outcomes, some unique measures of performance have been suggested for an OSS project that focus on the development process and, in particular, development group activity (Crowston et al. 2003). Conceptually development group activity represents the degree to which the OSS project is able to garner developer resources. Measures of development group activity studied in past research include development team size (Stewart et al. 2006d), number of releases (Stewart et al. 2006b), the extent to which teams accomplish work tasks (Stewart et al. 2006d) and the number of concurrent versions system (CVS) commits (Grewal et al. 2006).

Development group activity is a particularly interesting outcome because it is a differentiator across OSS projects and is a critical antecedent of other measures of OSS project performance. OSS projects may not be funded, making it difficult to attract development group activity. Without development group activity it is likely that the software will not continue to meet changing user needs. Specifically, development group activity is important for OSS projects because it can lead to other dimensions of performance such as usefulness and popularity. As user software needs change, development group activity can keep the software up to date and therefore lead to its usefulness and popularity. Essentially, most other OSS performance measures depend on development group activity.

Common development group activities such as fixing bugs and adding features are acts of knowledge creation in the form of software solutions. This is because the development group draws on its knowledge of a programming language and a particular problem to create new knowledge that is represented in the software solution. In this research development group activity is therefore conceptualized as a recreation of knowledge. The successful use of prior knowledge to create new knowledge is a commonly used measure of performance, especially as an intermediate outcome between resources and competitive advantage in the organizational literature (Cohen et al. 1990; Lane et al. 2006). This dissertation uses the organizational innovation literature, specifically the concept of absorptive capacity, to identify antecedents of two different outcomes based on development group activity; OSS project survival and OSS project development group activity intensity.

1.4 Research Goals And Contributions

This dissertation seeks to inform managerial practice and broaden the scholarly research regarding the impact of absorptive capacity in an OSS project. In exploring the development of absorptive capacity in an OSS project, I highlight the distinct effects of different types of knowledge and the participants who acquire and transfer that knowledge. Specifically, I contribute to the OSS literature by explaining how both the characteristics of the development group and an Internet-based user community (IBUC) affect OSS project performance through the type of knowledge they acquire and transfer. Prior literature has focused almost exclusively on the effect of the development group on development group activity (e.g. Stewart and Gosain 2006). This dissertation expands on prior literature by detailing how the IBUC has different yet complementary effects compared to the development group on development group activity.

In using absorptive capacity to understand the antecedents of OSS project performance I add to the body of literature that seeks to understand the development and effects of absorptive capacity. Several dimensions of absorptive capacity have been identified in prior work (Zahra et al. 2002). I extend this literature by exploring the interaction of two dimensions of absorptive capacity: knowledge acquisition capability and knowledge transfer capability. I also illuminate how acquiring knowledge with certain characteristics can have different effects, depending on the measure of performance observed.

Finally, because I argue that the characteristics and behaviors of OSS participants indicate absorptive capacity, I am able to inform managers about the most desirable types of participants. While this study is most applicable for the manager of an OSS project as he or she seeks to attract participants, it may be beneficial for other types of managers. An IBUC can be related to proprietary software and other kinds of product development, so this research can be used to inform managers of a variety of product development efforts concerning participants to target that increase the benefits of IBUCs. In particular this dissertation highlights the role of the IBUC as a knowledge resource and introduces an IBUC as a mechanism through which an organization can increase its absorptive capacity.

The remainder of this dissertation is organized as follows: Chapter 2 describes why absorptive capacity is important for understanding the antecedents of OSS project performance and presents the major constructs of the models developed. Chapter 3 then develops a research model that seeks to understand the antecedents of the survival of an OSS project. Chapter 4 presents a complementary research model that seeks to identify the antecedents of development group activity intensity in an OSS project. Chapter 5 details the empirical analysis of the models presented in Chapters 3 and 4. Chapter 6 concludes this dissertation with a discussion of the results, limitations and future research opportunities.

CHAPTER 2: CONCEPTUAL MODEL - ABSORPTIVE CAPACITY AND OSS PROJECTS

Chapter 2 discusses why a knowledge-focused perspective is useful and appropriate for understanding the antecedents of OSS project performance. I present the rationale for focusing specifically on absorptive capacity and discuss how an OSS project can build it. Specifically, in addition to developing a framework to understand how the development group can build absorptive capacity, I highlight the key role the IBUC plays in enhancing an OSS project's absorptive capacity. Finally, I describe the major constructs of my conceptual model and introduce the research models developed in Chapter 3 and 4.

2.1 A Knowledge Focused Perspective

Knowledge is an important resource for an organization that seeks to innovate and has been argued to be the most strategically significant resource of many organizations (Grant 1996; Kogut et al. 1992). Specifically, the knowledge base within a firm is a key determinant of superior performance (DeCarolis et al. 1999). Knowledge is important for an organization that seeks to innovate because knowledge facilitates the acquisition and application of new knowledge that can lead to innovation (Cohen et al. 1990).

As in other innovative organizations, knowledge is important for an OSS project. Knowledge about software problems and solutions is necessary for innovation in the OSS context. To continually meet user needs through innovation OSS developers must have knowledge of the software problems in business contexts and how the business contexts are changing. The developers in an OSS project must also have programming knowledge to be able to implement software solutions.

There are many constructs that represent a knowledge-focused perspective. Section 2.2 discusses why this research focuses on absorptive capacity for an OSS project.

2.2 Absorptive Capacity: Key OSS Capabilities

In order for an organization seeking to innovate in a dynamic environment to be responsive to the changing needs of users, it must be able to quickly acquire knowledge from users and incorporate this knowledge in the development of products. Absorptive capacity is critical for this kind of organization because it captures the degree to which an organization is receptive to knowledge and able to implement and exploit it.

2.3 The Development Of Absorptive Capacity For An OSS Project

An OSS project's absorptive capacity encompasses both its knowledge acquisition capability and its knowledge transfer capability. The knowledge acquisition capability enables the OSS project to receive knowledge necessary to innovate in a dynamic environment, such as knowledge about user needs and new techniques for addressing user needs. The knowledge acquisition capability specifically focuses on the OSS project's ability to acquire knowledge from outside its boundaries. The knowledge transfer capability facilitates the use of knowledge within the OSS project. Consistent with Argote and Ingram (2000), I define knowledge transfer as the process through which one unit is affected by the experience of another. The knowledge necessary for the OSS project to develop software may be distributed across many participants and for the OSS project to assimilate knowledge into software, it has to be transferred between participants. Specifically, the OSS project must be able to transfer knowledge to the participants, the development group, who can use it in the development process. I continue by detailing the development of knowledge acquisition capability and then the knowledge transfer capability in an OSS project.

2.3.1 Knowledge Acquisition And Preexisting Related Knowledge

The ability to acquire knowledge is enhanced when an organization has preexisting related knowledge (Cohen et al. 1990; Fichman et al. 1999). Preexisting related knowledge is the extent of abstract knowledge, know-how, and skills within the organization in areas related to the focal innovation (Fichman et al. 1999). Preexisting related knowledge enables knowledge acquisition by providing the foundation necessary for understanding new knowledge and how the new knowledge is relevant to innovations. For example, in an OSS project, knowledge about HTML facilitates the acquisition of knowledge about XML and the ability to understand why XML is relevant to the OSS project.

Different types of organizations have different ways of developing and maintaining their preexisting related knowledge. In firms the development of preexisting related knowledge is done through research and development departments (Cohen et al. 1990) or managerial practices (Lenox et al. 2004). In contrast, Malhotra et al. (2005) describe how enterprises in supply chain partnerships configure their processes and IT infrastructures to develop and maintain their preexisting related knowledge. In particular, they propose that joint decision-making and memory information systems build preexisting related knowledge bases for supply chain partnerships.

An OSS project also has unique methods for developing and maintaining preexisting related knowledge. An OSS project's preexisting related knowledge base is developed and maintained primarily by its participants. Thus, OSS participants embody the knowledge available to a project, and their characteristics and roles indicate the kind of knowledge that is available in an OSS project.

Participants in an OSS project can be categorized in several ways. In their study of the GIMP project, an OSS project that processes images in Linux, Ye and Kishida (2003) describe eight types of participants. These include the project leader, the core members, the active developers, the peripheral developers, bug fixers, bug reporters, readers and the passive users (Ye et al. 2003). Likewise, Sturmer (2005) presents a set of participant types that include core developers, initiators or owners, developers or leaders, active users and inactive users.

The detailed classifications described above, developed from exploration of the largest OSS development projects provide useful details about these organizations, but cannot be generalized for application in a broader set of OSS projects. As Ye and Kishida (2003) note, not all OSS projects have participants to fulfill the multiple roles that their detailed membership classification describes. Many OSS projects are small or early in their lifecycle and therefore do not have participants to fill the roles described. It is useful to observe an OSS project that is early in its lifecycle in order to understand the factors that *lead* to performance.

I argue that there are two core types of participant sets relevant in most OSS projects, regardless of the stage, but that these sets have distinct types of preexisting knowledge related to the OSS project. The two sets are the IBUC and the development group. Drawing from the growing literature on OSS participants, I define an IBUC as the participants who engage in some computer-mediated communication related to an OSS project, but do not actively develop source code (Hertel et al. 2003; Ye et al. 2003). An IBUC member could report bugs, submit feature requests, provide feedback about usability and provide technical support (Krogh et al. 2003; Lakhani et al. 2003; Ye et al. 2003; Zhao et al. 2003). The IBUC would include Ye and

Kishida's peripheral developers, bug fixers, bug reporters, readers and the passive users. The developers then are those who actively develop source code and would include Ye and Kishida's project leader, core member and active developer roles.

The developers have been the focus of most OSS research, as will be discussed below. Research focused on the development group provides a firm foundation for further OSS research, but it emphasizes only one of the two types of knowledge necessary for innovation. The two types of knowledge necessary for innovation are awareness knowledge and how-to knowledge (Rogers 1995; Tornatzky et al. 1990). Development group research stresses the availability of how-to knowledge or knowledge concerning how to develop software. My dissertation builds on the research focused on the development group by specifically highlighting the distinct influence of the awareness knowledge that the IBUC can provide in addition to the development group knowledge. First, I discuss the development group as a provider of how-to knowledge.

2.3.1.1 The Role Of The Development Group: How-To Knowledge

This section reviews the literature concerning the OSS development group and then describes how this dissertation extends this literature by adopting a knowledge-focused perspective. In particular, I explain how considering the how-to knowledge that the development group provides can distinguish a high performing OSS project. I further describe how-to knowledge as *one* type of knowledge necessary for an innovative organization, and thereby introduce the importance of the awareness knowledge provided by the IBUC.

A large body of research addresses the motivation of a developer to participate in OSS development and finds that key motivations are the desire to use the software developed, enjoyment of the software development process and opportunities to refine development skills

(Hars et al. 2002; Roberts et al. 2006). This research provides insight into the OSS community at large, but does not explain why an OSS project can perform better than another OSS project.

Multiple perspectives have been leveraged to understand the factors that allow an OSS project to perform better than another OSS project. From a technical perspective, research has sought to understand the ease with which developers can participate on a project by considering aspects of the code structure (Baldwin et al. 2006; MacCormack et al. 2006). The primary argument is that if the code is organized in a way that allows a developer to understand it easily and add features or fix bugs then that project will perform better than another OSS project where the code is not well organized. Stewart and Gosain (2006) take a more social psychological perspective and provide evidence of the effect of development group characteristics, such as adherence to OSS ideology, on performance. They find that adherence to some ideological components enables the project to attract and retain input. For a more thorough review of the literature focused on OSS development groups, see Table 1.

While this literature explains what motivates developers and facilitates their participation it assumes that the developers have the requisite knowledge to adequately develop software. A focus on the available knowledge in the development group is important because, as argued earlier, knowledge is a key resource, along with motivation and a cooperative environment that is necessary for software development. The current work expands this body of research by considering the degree to which the knowledge base within the development group leads to software solutions of relevant problems. However, as discussed above, innovation requires both how-to knowledge and awareness knowledge. The next section discusses the awareness knowledge.

2.3.1.2 The Role Of The IBUC: Awareness Knowledge

Awareness knowledge is knowledge required to use the innovation in a work context. An IBUC complements the development group because an IBUC is likely to have a different knowledge base from the development group. The IBUC can provide awareness knowledge, which is focused on problems that the software can solve. While the OSS development group often has members who both develop and use the software (Zhao et al. 2003), I argue that in general the IBUC is positioned to have awareness knowledge that complements the development group. The IBUC's complementary awareness knowledge is derived from the IBUC's access to a broader set of knowledge related to the software, compared to the development group. This access is expected because the IBUC may have different types of employment and software experiences. This section continues by further describing the IBUC and the knowledge base it possesses that serves to complement the development group.

Because the IBUC is not required to be actively involved in writing source code, the IBUC is more likely than the development group to accommodate members who are involved in professions other than software development. In a sample of participants drawn from SourceForge, Zhao and Deek (2003) find 60% of participants, that according to this research would be classified as IBUC members, have more than five years of development experience (Zhao et al. 2003). This implies that a substantial proportion of the IBUC, approximately 40%, have limited or no software development background. Having limited software development experience enables those in the IBUC to have a different perspective about the software and how it can be made easier to use. In summary, because there are fewer restrictions for participation in the IBUC, and the focus is on usage, an IBUC can have a broader knowledge base that has a stronger focus on usage compared to the development group.

Knowledge from participants who are not software developers can be important. Some software improvements can be identified only by those entrenched in the context where the software is used. For instance, the request below, from an IBUC member, draws on a context specific situation and identifies additional important functionality for the software. This kind of idea is not easily generated from within the development group.

"Let say the customer only want to payment only after their have finish their meal. Then i can't print the information to kitchen. Because the Printer.Ticket will only print when customer done their payment."

http://sourceforge.net/forum/forum.php?max_rows=25&style=flat&offset=25&forum_id=434920

A knowledge base that comes from software users in the business context is also important because the software may not be used in the way the developers designed it to be used (Orlikowski 1992). The IBUC may interpret, appropriate and manipulate the software to fit their geographical, technological or cultural setting, transforming the conceptualization of the software application from that which the developers intended. Mackey documents how users of electronic mail systems employed different usage patterns based on their individual preferences (Mackay 1988). This transformation in the identity of the software when used by the IBUC can lead to knowledge acquisition in the form of unique ideas for further development that complement the knowledge acquisition capability of the development group.

Another important characteristic of an IBUC that allows it to complement the development group is the speed at which the composition of the IBUC can change. The ability of the IBUC to change rapidly is important because technology changes quickly and a change in the IBUC membership could enable the project to acquire relevant knowledge necessary to meet changing user needs. The size, make-up and pattern of interactions that characterize the IBUC can swiftly change over time for at least two reasons. First, the Internet supports IBUCs and allows people from around the world to join and leave at any time. Second, since the IBUC is not actively involved in software development, its composition is inherently flexible because the coordination requirements for participation in the IBUC are lower than for the development group.

There are many empirical examples from the literature that illustrate the flexibility of the IBUC. Von Krogh et al. (2003) note that for the OSS development project Freenet, 356 individuals participated on the discussion list, implying there are 356 participants in the IBUC. Likewise, in a study of Apache, Mockus et al. note that 3,000 people contributed by reporting problems with the software (Mockus et al. 2000). However, there are several projects on SourceForge where there is no IBUC or there is an IBUC with one member. This observed variance in the size of the IBUC demonstrates that the IBUC is inherently flexible in nature.

Whether because of the focus on usage or the flexibility of the IBUC, it is well positioned to provide awareness knowledge to complement the how-to knowledge provided by the development group. In providing this alternative perspective the IBUC also functions to disturb the development group's tendency to focus on areas where the development group has had prior success. An organization often focuses its search for knowledge in a limited space where the organization has had prior success or is familiar with the knowledge. This tendency toward a local search inhibits the innovative process (Rosenkopf et al. 2001). An IBUC may expand an OSS project's search space. For example, based on the different interests and professions of the IBUC members, the IBUC is likely to know of different OSS projects and proprietary software applications from which knowledge can be acquired when compared to the development group.

Specifically, because the IBUC could have affiliations based on professions other than software development, the IBUC can be aware of a broader set of OSS projects from which code or ideas can be borrowed compared to the development group.

So far, this discussion has focused on characteristics of an IBUC that may be associated with any type of software. However, an IBUC associated with an OSS project is positioned to have an even more influential role in developing knowledge acquisition capability for an OSS project as compared to a proprietary software development project. While the IBUC for both types of software projects can enhance the knowledge acquisition capability, there are some distinct ways that an OSS IBUC facilitates an OSS project's knowledge acquisition capability. For instance, in contrast to IBUCs related to proprietary projects, an OSS project IBUC has access to the source code, and is therefore in a position to acquire knowledge about needed features and ways to implement those features. An OSS IBUC can further complement the knowledge of the development group by providing knowledge that an organizational sponsor typically supplies: current user needs and a plan to meet those needs. An OSS IBUC, through its knowledge in the form of feature requests and bug reports, can influence timelines and requirements defining the expected outcome of the project. By providing knowledge in the form of specific suggestions, the OSS IBUC participants give the development group opportunities for development that will be valuable to users. In particular, the IBUC aids the development group by acquiring knowledge that the development group can use to create software that meets changing user needs.

In summary**,** the OSS development group and the OSS IBUC represent the two sets of participants that are expected to have both complementary and unique effects on the OSS project's absorptive capacity. The OSS development group is composed of those OSS development project participants who actively write code and therefore have preexisting related knowledge about the software solution, or source code. This type of knowledge enables developers to acquire new knowledge about ways to enhance the software from a coding perspective. The IBUC is composed of those participants who exhibit some connection to the project via participation in computer-mediated communication associated with the project, but do not develop source code. IBUC participants associated with OSS projects are more likely than those in the development group to have backgrounds not related to software development. Diversity in the background of IBUC members coupled with the lower coordination costs of participation in the IBUC yields a broader knowledge acquisition capability that is more likely to be focused on user needs.

2.3.2 Knowledge Transfer And Overlapping Knowledge

A strong knowledge transfer capability is critical for an OSS project to allow the IBUC knowledge acquisition to consistently lead to development group activity. The IBUC possesses and can acquire knowledge that can enable development group activity, but that knowledge must be transferred to the development group in order to enable activity. Therefore, the relationship between the IBUC knowledge acquisition capability and development group activity is influenced by the knowledge transfer capability of the project.

The knowledge transfer capability is facilitated by overlapping knowledge held by units within an organization (Clark 1996; Cramton 2001; Fussell et al. 1992). At a minimum the two subunits must share a common language to transfer knowledge (Dearborn et al. 1958; Katz et al. 1966), but as the similarity of the shared cognitive structures increase the transfer of knowledge is easier (e.g., (Bower et al. 1981; Ellis 1965)). Thus, the extent of knowledge overlap between the IBUC and the development group is expected to alter the ease with which knowledge is transferred.

I explore two indicators of the degree of overlapping knowledge between the IBUC and the development group. One indicator is the software application type that is being developed. If the software is designed for use by developers, both the users and the developers share knowledge about software development. In this case the knowledge overlap between the IBUC and development group is higher than if they did not share this software development knowledge. The second indicator of knowledge transfer capability for an OSS project is the amount of communication between the development group and IBUC. As the IBUC and development group communicate they exchange knowledge, thereby amplifying their shared, overlapping knowledge.

2.4 Major Constructs And Relationships

A depiction of the overarching conceptual model describing the relationships presented in section 2.3 and guiding this research is shown in Figure 1. This model draws on Zahra and George's concept of absorptive capacity and its importance for organizations that seek to innovate in a dynamic environment. In particular, it focuses on the knowledge acquisition and knowledge transfer capabilities, which I argue are dimensions of absorptive capacity. The development group and IBUC have complementary critical knowledge acquisition capabilities and so I assess the effect of their knowledge acquisition capabilities on performance separately. Further, since the development group is responsible for development, I assert that the project knowledge transfer capability influences the relationship between IBUC knowledge acquisition capability and project performance, measured as development group activity.

Insert Figure 1: Conceptual Model Here

2.5 Research Models Overview

I define the performance outcome, OSS development group activity, as work performed by the development group that facilitates the evolution of the OSS. There are two dimensions of development group activity that are of interest to potential adopters and are likely to have different antecedents. Potential adopters may be interested in OSS project survival and development group activity intensity and they are expected to have different antecedents, therefore this dissertation develops two separate models for these outcomes. The first research model developed seeks to understand the antecedents of OSS project survival, where survival is defined as the OSS project having development group activity in a time period after the initial release of the OSS application. The second research model seeks to understand the factors that lead to development group activity intensity. This is a measure of the degree to which the developers are able to accomplish work that evolves the OSS.

The model focused on survival is important because for some users to be comfortable adopting OSS, the user must expect the OSS project to survive by exhibiting development group activity long after its initial release. Without development group activity over the long-term a user who expects to use the software long-term may have concerns about maintenance of the software and the ability of the software to evolve to meet their changing needs. These concerns lead users to decide to choose some other application (Norris et al. 2004).

I draw on March (1991) to understand the factors that enable an OSS project to survive. March (1991) suggests that for organizations that seek to learn to be able thrive in the long-term, the exploration of new knowledge is critical for success. It is critical for OSS projects to continue to learn because the software industry is dynamic and characterized by changing user needs. The survival of an OSS project is expected to be dependent on the explorative capabilities of the project because without new ideas the project will not meet the changing needs of users and the changing interests of developers. Stated otherwise, without exploration capabilities the OSS project is likely not to survive because developers and users will lose interest. For this reason this model focuses on the exploration of new knowledge as a key construct determining OSS project survival.

In contrast to the adopter interested in survival, another potential adopter may only plan to use the software for the next year to complete a single task. Her concerns are focused on the degree to which the development group will evolve the software intensely over the short-term. In this case it is important for developers to do mundane tasks frequently and consistently to meet the needs of users. The degree to which the project is evolving the OSS, which I term development group activity intensity, is therefore an important indicator of performance.

As in other organizations, in order to consistently produce exploration and exploitation are necessary (March 1991). Therefore, this model focuses on both the exploration for new knowledge and the exploitation of existing knowledge in developing an understanding of the antecedents of development group activity intensity. With only a strong explorative capability a project may survive, even though only the interesting work is completed at the leisure of the developers. Drawing on the concept of exploitation as performing tasks that draw on old certainties developed by March (1991), my conceptualization of exploitation in OSS projects captures the ability of the project to leverage resources to consistently complete even the mundane tasks.

Because the survival model focuses only on knowledge that enables exploration and the development group activity intensity model examines knowledge that facilitates exploration balanced with knowledge that leads to exploitation, some antecedents are likely to have different affects on the two outcomes. For instance, diversity leads to exploration that can lead to profitable innovations (Cohen et al. 1990), but it may also lead to conflict that can hinder performance (Dreu et al. 2003). Therefore, in an OSS project, diverse knowledge should increase exploration and thus lead to survival by offering the development group new exciting ideas. However, diversity is expected to have a more complex relationship with development group activity intensity which requires agreement within the development group. In developing two separate models, this dissertation illuminates the complex affects of diverse knowledge on performance.

When considering these research models, two control variables are necessary because these variables are positively associated with development group activity and are outside the scope of this research. Time influences development group activity in many ways. As teams work together over time they develop evolved routines and procedures to facilitate the conversion of inputs to outcomes and therefore time may lead to higher levels of development group activity (Ethiraj et al. 2005). Project management capabilities also accumulate in an additive fashion over time and may lead to higher levels of development group activity over time (Whang 1995). Second, I control for the number of project members because it is has been shown to affect the development group activity (Ren et al. 2006). Further, each member represents knowledge and should therefore enhance the project's knowledge acquisition capability. However, the focus of this research is not in understanding how the number of members leads to OSS project performance.

2.6 Summary

This chapter discussed why a knowledge-focused perspective is used and why I focus on the development of absorptive capacity in an OSS project. I presented the two focal dimensions of absorptive capacity that I examine; knowledge acquisition capability and knowledge transfer capability. Then this chapter discussed the development group and the IBUC as the two OSS project participant sets that have distinct effects on the development of OSS project knowledge acquisition capability. I argued that this dissertation adds to the OSS literature by moving beyond the previous focus on the development group and in particular exploring the knowledge from the IBUC that may be broader than the development group and more comprised of awareness knowledge. The importance of knowledge transfer capability was explained and the conceptual model with major constructs and their relationships were presented. Finally, the two research models and controls for models were introduced. Research models based on this conceptual model are developed in Chapters 3 and 4. They detail how absorptive capacity leads to the survival of an OSS project and development group activity intensity respectively.

CHAPTER 3: HOW DOES ABSORPTIVE CAPACITY LEAD TO AN OSS PROJECT'S SURVIVIAL?

3.1 Overview

Many OSS projects release software, but do not survive long after the initial release. As discussed in Chapter 2, I define survival as the OSS project having development group activity in a time period after the initial release of the OSS application. Stewart et al. (2006) document the short lifespan of many OSS projects by showing that many of them stop releasing new versions of the software after approximately one year. Some OSS projects survive, and the survival of an OSS project is a positive mark of distinction because it demonstrates that the project is useful and that someone is interested in developing it further. Specifically, developers and or users are using their resources to identify ways to enhance the software and implement those enhancements.

What factors contribute to the survival of an OSS project? To answer this question, I use an absorptive capacity lens because it "makes the firm receptive to acquiring and assimilating knowledge (Zahra et al. 2002)." An OSS project must be able to acquire and assimilate knowledge in order to continue developing the kind of software that interests developers and users. The motivations underlying OSS developer participation in OSS development includes their interest in refining development skills and potential career opportunities (Ghosh 2002; Hann et al. 2002; Lerner et al. 2002b), implying that developers are more likely to be motivated to participate in projects where they have opportunities to acquire knowledge about new and popular software solutions. Therefore, to continually attract developers and motivate them to participate, the OSS project needs to be able to acquire knowledge about the current trends in software applications and development.

I examine two characteristics of knowledge that enable the OSS project to have and acquire creative ideas for software solutions that reflect the current trends in software applications and development styles. I focus on the diversity and freshness of knowledge because I expect these characteristics to enable the OSS project to derive creative and current software solutions for the long term. The diversity of an OSS project's knowledge base is important because it provides the opportunity for an organization that seeks to innovate to connect ideas that were previously thought to be unrelated and thereby provide interesting development opportunities. Innovation often occurs by applying knowledge in a new context or connecting knowledge that previously had been considered unrelated (Cohen et al. 1990).

The freshness of knowledge is the degree to which the knowledge represents current changes in the environment outside the OSS project. Because an OSS project operates in a dynamic environment where user requirements change quickly, their survival depends on their ability to acquire fresh knowledge. The importance of fresh knowledge for organizational innovation is well documented in studies that focus on the positive effects of voluntary employee turnover (Dalton et al. 1979). Voluntary employee turnover can enhance performance by revitalization; it increases workforce innovation, flexibility and adaptability (Abelson et al. 1984). In contrast organizations characterized by low turnover may experience workforce skill stagnation, closed-mindedness, and "trained incapacity" (Dalton et al. 1979). In an IBUC, new members can inject the project with fresh ideas that lead to performance. They can do this without decreasing productivity when they leave, because the project does not depend on them to transfer knowledge to the future developers.

An OSS project's ability to acquire fresh and diverse knowledge is dependent on the characteristics of the knowledge base already held by the organization. Cohen and Levinthal argue an organization acquires knowledge in the same way that an individual does (Cohen et al. 1990). An individual's existing knowledge base facilitates the acquisition of some types of knowledge, but not others (Goldstein 1991). As an example, knowledge of algebra may facilitate the acquisition of calculus knowledge, but not knowledge about historical events (Ellis 1965).

For an OSS project, I consider characteristics of the OSS project's knowledge base in order to understand its ability to acquire diverse and fresh knowledge. For participants to have a diverse knowledge base, they should have a mix of long and short tenures with an OSS development platform. An OSS development platform is a bundle of resources that enable OSS development. It may include communication tools such as public forums or mailing lists, a bug tracking system and a current version system. The platform tenure diversity represented by the individuals in the development group and IBUC is a reflection of the diversity of the knowledge base. Platform tenure diversity is a characteristic of the IBUC or development group that indicates the diversity of individuals with respect to the time each individual became engaged in an OSS development platform.

The IBUC dynamism is a characteristic of the IBUC that indicates the degree to which the membership of the IBUC changes, and is a reflection of the freshness of knowledge available to the OSS project. Fresh knowledge may also be available to the project through developers. However, while theoretically developers can join and leave an OSS project at any time, there is evidence that this is not the case in practice. Developers may not change projects often because it is time intensive for a developer to understand source code enough to contribute to a project. Also, there is a limited supply of OSS developers for each project and so a developer's time is probably consumed by a single OSS project. Evidence of this phenomenon is presented by

Madey, Freeh et al. (2002) who analyze 39,000 OSS projects hosted on SourceForge over a 14 month period. They find 70% of developers belong to only one OSS project during the 14 month period. For these reasons development group dynamism is not a focus of this study. This chapter continues with detailed hypotheses explaining how IBUC and development group platform tenure diversity and IBUC dynamism lead to the survival of an OSS project.

Insert Figure 2: Research Model-Survival Here

3.2 Platform Tenure Diversity

The time a person joins an organization indicates a lot about their knowledge related to the organization. There are two main reasons tenure indicates the knowledge a participant may have related to the technology. First, organizations change, and the time a person joins indicates the specific instance of the organization, with which they were introduced and experienced over time (Ryder 1965). Similarly, tenure with an OSS development platform represents the knowledge participants have because joining at a particular time determines specific events that participants experienced related to the platform. Further, platforms evolve, and so some features of the platform may have been more salient when a person joined and influence the type of knowledge that the person has concerning the platform. Also, the time a person starts using a platform is often motivated by an external stimulus, such as a software development class or a corporate initiative. In this case, when a set of people join at a similar time they have a common university or corporate background and associated knowledge. In contrast, when the participants in an OSS project have joined the development platform at different times, they are more likely to have different knowledge related to the OSS project.

Second, time with the platform can alter the way a person thinks about the OSS project. Over time, participants can build a common understanding for how to interact with the other people associated with the platform. Knowledge is typically gained over time through socialization into the community and repeated interactions. Participant interaction establishes standard practices, methods of development, coordination, ways of thinking (Giddens 1976) and an organizational language (March et al. 1958). So, long tenure participants are likely to have been a part of the development of the standards and thus understand them more.

Over time participants can also be constrained by the technology. Tenure can determine knowledge because interaction with a technology constrains the way in which tasks are done, the associated performance, and the thoughts of the person in relation to the task (Orlikowski 1992). A development platform uses a limited number of technologies and long-tenure participants possess knowledge defined by use of those technologies. Long-tenure participants are therefore constrained by the use of those technologies. Tenure with an OSS development platform in particular, reflects the participant's experience with OSS development and applications and can constrain the way they accomplish tasks, their performance and thoughts. In particular, participants with longer tenures are likely to have a knowledge base characterized by experiences with OSS applications. In contrast, short-tenure participants may have a different set of knowledge that is based on experience with proprietary or limited software use. Therefore, expectations about how software should work can be related to the time a participant has been associated with an OSS development platform.

In summary, tenure with a particular development platform partly determines the type of knowledge germane to the OSS project held by the project participant. For participants to have a diverse knowledge base, they should have a mix of long and short tenures with an OSS development platform. Having multiple knowledge bases, some based on long histories with an OSS development platform and others based on shorter tenures with it, is expected to lead to a larger set of unique knowledge. This larger set of knowledge should enhance the knowledge acquisition capability and the OSS project's survival.

3.2.1 IBUC Platform Tenure Diversity

As discussed in Chapter 2, the IBUC is expected to have awareness knowledge such as user desired functionality and ease of use. This is because IBUC members are actively involved in the business context where the application is used, and as the context changes, the IBUC is well positioned to acquire knowledge about useful software innovations to meet their changing needs. An IBUC characterized by diversity is then likely to lead to a larger variety of contributions related to desired functionality and ease of use.

IBUC platform tenure diversity is reflected in the IBUC platform tenure diversity because long-tenure platform IBUC members are likely to have different experiences and backgrounds compared to short-tenure platform IBUC members. IBUC members who have engaged with the development platform longer are likely to be experienced in the specific development and distribution process. Therefore, experienced IBUC members are more likely than inexperienced IBUC members to understand the type of features the OSS development group is willing to develop and the kind of detail necessary in the feature request or bug report. However, short-term IBUC members might possess other kinds of useful knowledge. Because they are new to the OSS development platform they could have more experience with proprietary software, and thus, be better positioned to acquire knowledge that comes from the use of proprietary software.

An IBUC that has members with different experiences has more diverse knowledge that enables it to acquire a larger variety of knowledge compared to an IBUC with homogenous membership. The broader knowledge base leads to greater exploration in the form of development options. Because there are more options there is a greater likelihood that they will arrive at innovative and creative suggestions that will be of interest to developers. But, in addition to more ideas there are opportunities for creative combinations from the diverse knowledge.

One specific example of the synergy between the knowledge held and acquired by short and long-term IBUC members is if a short-term IBUC member brings an innovative idea and the long-term IBUC member helps turn that idea into an actionable request. This happens because the short-term IBUC member may be familiar with other software and the long-term IBUC member is familiar with the development platform process. The example below, drawn from the forums of the Azureus OSS project, shows how a short-term IBUC member offers an idea for a feature, and then a long-term IBUC member provides details that help the development group add the feature.

Short-Tenure Platform IBUC Member *"Nobody is talking about the one important missing on Azureus!?...This is strange…Why this professional soft is not including on load (torrent) menu, a mathemtatical calculation of drive space needed for all files or for selected files only, when not entire content will be downloaded….What do you say about this extremely important missing?"*

Long-Tenure Platform IBUC Member *"I'd say, I agree with you. I'm so used to do that mentally that I forgot it is a obvious feature to ask. You have my vote. Trying to be specific and offer some ground for discussion, the place where I think that could be, are in the torrents detail/Files, probably in the file pieces, a bottom indicator with one pair of values, the "reserved space" and the "needed space to finish", both for the selected, and another pair with the same type of values but this one for the all the torrent. The second pair will take into account the DND* *flag of the files, the first pair will not…The second pair would be red if the aggregated needed space to finish for the torrent are less than that value, the first pair will take that same color if the needed space to finish for the DND selected files are less than that spare space…Last, the values don't need to be in Bytes, we could use blocks as a more readable unit measure. Other obvious place are the open torrent dialog. After that I'm afraid of the feature impact in the performance of the application"*

As illustrated in the example above, platform tenure diversity allows for synergy between development platform long-term and development platform short-term IBUC members that can lead to innovative, detailed feature requests and bug reports. Further, as argued previously, IBUC platform tenure diversity leads to a larger base of knowledge that facilitates knowledge acquisition in the form of feature requests and bug reports.

Further, as time passes and different types of software become more popular, when the IBUC is diverse, the project will be more likely to have IBUC members familiar with the newly popular software. When the IBUC has members who are familiar with popular software, the OSS project is better prepared to thrive in the new software environment. Long-term IBUC members are likely content with using OSS applications and unaware of advancements in proprietary software. So, if a new proprietary application becomes popular, but the OSS project has all long-term members, the OSS project may not have the knowledge necessary to adapt to the new trend.

In summary, because an IBUC with platform tenure diversity leads to more exploration in the form of opportunities for development that are innovative and current, the development group will be excited about development opportunities on that project. Therefore developers will continue or start developing on that project and so I propose:

Hypothesis 1: IBUC platform tenure diversity is positively associated with the survival of an OSS project.

3.2.2 Development Group Platform Tenure Diversity

While the IBUC's influence is primarily through knowledge acquisition concerning desired functionality and ease of use, the development group platform tenure diversity indicates knowledge acquisition related to techniques for software development.

Tenure with a particular development platform and community can provide many benefits for the development group, but may also have limitations and impose constraints. Developers with long tenures are expected to be experienced and that experience has many benefits. Experienced developers are likely to be better able to work with others in a group and have refined development skills. The knowledge of an experienced developer is likely to allow him or her to acquire knowledge of complex or elegant programming solutions, which can lead to development group activity. However, experience as a developer, as signaled by platform tenure, can also be limiting; it could constrain the developers' ability to acquire knowledge. Like the IBUC, experience with a particular platform constrains a developer's ability to acquire knowledge because it limits the way the developer thinks about solving a task (Orlikowski 1992). A tenured developer is likely to be biased toward the development styles easily available in a particular platform and have trouble identifying the relevance of new development styles. Therefore, it is often difficult for developers with longer tenures to acquire knowledge about new potentially useful development styles, languages or programming techniques. Just as in other organizations the OSS development group is dependent on short term participants to be the "bearers of technological innovations and new values." (Reed 1978)

Shorter tenure with a particular platform indicates some knowledge acquisition opportunities that complement the knowledge acquisition of longer tenure developers and thus offer opportunities for exploration. Shorter tenures may be a sign of less experience as a programmer. Less experienced programmers may be familiar with a different and newer set of programming languages and techniques that allow them to acquire knowledge about these techniques. As an example, a more experienced programmer with a strong foundation in a language such as C++, that was popular earlier, may not be as prepared to acquire knowledge about new program languages such as Java. A shorter tenure developer can complement the long tenure developers by accessing newer technologies.

In summary, development group platform tenure diversity is expected to signal a more diverse source code development related knowledge base. A diverse knowledge base should generate more options for how to find and fix bugs and add features; thus, development groups with more platform tenure diversity are likely to have a better chance at exploration and survival. Further, as in the case of the IBUC, there may be a productive collaboration between the development expertise of the long-term platform tenure developers and the innovative ideas of the short-term platform tenure developers. The short-term platform tenure developers can offer ideas for development that are new and relevant and the long-term platform tenure developers can help implement them. I therefore propose:

Hypothesis 2: Development group platform tenure diversity is positively associated with the survival of an OSS project.

3.3 IBUC Dynamism

A key component of an organization's ability to create knowledge is having new knowledge (Cohen et al. 1990). One way that an organization gets new knowledge is by gaining new individuals who can access new knowledge. For example, organizational turnover can be seen as providing "fresh blood" through which to gain a new outlook (Dalton et al. 1979; Staw 1980). Each new individual brings both some new knowledge and the ability to acquire additional knowledge based on the unique knowledge that they bring. As new users come they can use their unique perspective to provide innovative opportunities for development activity. Even if the user suggests development opportunities that have been considered, they bring new knowledge in that the developers learn an additional person is interested in the feature. For an OSS project to continue to evolve software to meet changing user needs, it is critical that the OSS project increase its knowledge base by continually acquiring more users. Each new request from a user offers developers an occasion to explore novel and relevant development opportunities. As an example of how IBUC dynamism can enable survival, consider how this new IBUC member requests that the software developed by RapidMiner (YALE) be updated to be compatible with advancements in another OSS project.

New IBUC Member *"Hey all, I'm relatively new to RapidMiner, and so far I'm exceedingly impressed with not only its available features but it's workflow model for setting up experiments. The only catch: I'm currently a Weka user, and I've come to rely upon some of its extensible modules, namely BioWeka and FuzzyWeka*."

Developer *"I never tried to integrate a Weka module into RM and there is no documentation about that (at least none I am aware of). But I also think that this should be possible (with more or less work)"*

Based on these kinds of suggestions an OSS project will be able to continue to evolve to meet user needs. For these reasons I propose:

Hypothesis 3: IBUC dynamism is positively associated with the survival of an OSS project.

3.4 Summary

In Chapter 3 I developed a research model that focuses on the effect of an OSS project's absorptive capacity on its survival. This model focuses on the project's ability to acquire knowledge that leads to exploration in the form of innovative development opportunities. Specifically, I explored the platform tenure diversity in the IBUC and the development group and the IBUC dynamism as indicators of the project's ability to acquire knowledge that allows for innovative development opportunities. The next chapter presents a research model that investigates the effect of OSS absorptive capacity on development group activity intensity. This model examines the ability to acquire the kind of knowledge that leads to exploration and initial development group interest, and also capabilities that enable the development group to exploit knowledge.

CHAPTER 4: HOW DOES ABSORPTIVE CAPACITY LEAD TO OSS DEVELOPMENT GROUP ACTIVITY INTENSITY?

4.1 Overview

The model developed in Chapter 3 depicts how absorptive capacity facilitates OSS project survival and, in particular, examined the organization's exploration of current development options. The model developed in this chapter is focused on understanding how absorptive capacity enables the development groups of some projects to complete tasks related to designing and developing a piece of software consistently. This model investigates an OSS projects' ability to have new ideas, exploration, and also the ability to consistently address the mundane tasks, exploitation. Performance for this model then is development group activity intensity.

This model includes the knowledge acquisition capabilities of the model developed in Chapter 3 and incorporates additional constructs focused on exploitation. In particular, the knowledge transfer capability is included and an additional characteristic of the project participants that is expected to facilitate knowledge acquisition. The additional constructs are important dimensions of absorptive capacity focused on exploitation that are expected to facilitate development group activity intensity.

As discussed in Chapter 3 diverse and fresh knowledge is necessary to provide the development group with ideas for needed features or bugs to be fixed, and methods to implement the ideas. However, in addition to the ability to acquire fresh and diverse knowledge, to exhibit development group activity intensity the project must be skilled at assimilating and exploiting knowledge. To assimilate knowledge so that it can be implemented, it must be transferred to the development group. I therefore argue that the project's knowledge transfer capability moderates the relationship between the IBUC knowledge acquisition and the development group activity intensity.

4.1.1 Knowledge Acquisition And Development Group Activity

I consider the acquisition of knowledge types that affect the OSS project's exploration and exploitation in order to understand the effects of knowledge acquisition on development group activity intensity. Specifically, I explore the types of knowledge that lead to the identification of current and relevant software solutions, and facilitate the exploitation of development skills needed to implement those solutions. As discussed in the previous chapter, the freshness and diversity of the knowledge is important because it enables exploration, especially in the form of the creation of current software ideas necessary for software development in a dynamic environment.

In this model in addition to considering the diversity and freshness of knowledge, I consider the applicability of knowledge as it affects development group activity intensity. I consider the applicability of the knowledge because it provides the foundation necessary for the ideas to be understood and implemented by the development group, which specifically facilitates exploitation in the form of development group activity intensity.

As argued in the previous chapter, an OSS project's ability to acquire diverse, fresh and applicable knowledge is dependent on the characteristics of the knowledge base it possesses. Therefore, I focus on characteristics of the participants in the OSS project that indicate the characteristics of the knowledge base the project possesses. The development group and IBUC platform tenure diversity represents the diversity of the knowledge base and the IBUC dynamism indicates the freshness of the knowledge base. The IBUC and development group relationships with other OSS projects indicate the applicability of the knowledge base in the OSS project.

4.1.2 Knowledge Transfer And Development Group Activity Intensity

The knowledge transfer capability is vital in determining the degree to which the knowledge acquired by the IBUC influences the ability of the development group to exploit knowledge and lead to development group activity intensity. When the knowledge transfer capability is strong, the IBUC will more easily be able to pass ideas to the development group.

The amount of overlapping knowledge held by the development group and IBUC is used to understand the knowledge transfer capability of the OSS project because overlapping knowledge facilitates knowledge transfer. As indicated in Chapter 2, the application type that the project develops and the amount of communication between the development group and IBUC signify the degree of overlapping knowledge between the development group and IBUC. Figure 3 displays the research model investigated in this chapter. Each of the constructs and relationships are elaborated upon below.

Insert Figure 3: Research Model-Level of Development Group Activity Intensity Here

4.2 Platform Tenure Diversity

While a diverse knowledge base is expected to have positive effects on the survival of an OSS project, the effects on development group activity intensity are expected to be more complex. This section details the type of effect diversity should have on development group activity intensity.

4.2.1 IBUC Platform Tenure Diversity

IBUC members who have used the OSS development platform for a long period of time may encourage the project to be more like other OSS applications using the platform with which they are familiar. Short-term IBUC members, who may be familiar with using proprietary software, are likely to offer suggestions that would move the project to be more similar to other proprietary applications. Thus, an IBUC that is diverse based on platform tenure is likely to have a variety of distinct ideas concerning how the application should evolve to meet user needs. This diversity can reveal itself when IBUC members submit feature requests. When an IBUC offers inconsistent suggestions to the development group, the development group has the potential to be conflicted concerning which path to follow. Conflict can lead to developers leaving the project or not developing as much as they would if there was a consistent vision. In this way IBUC tenure diversity limits the development group activity and so I propose:

Hypothesis 4: IBUC platform tenure diversity is negatively associated with development group activity intensity.

4.2.2 Development Group Platform Tenure Diversity

Diverse ideas in a group can lead to conflict (Dreu et al. 2003). If an OSS development group is diverse based on tenure they are likely to have access to a wide scope of knowledge concerning development techniques and styles. When SourceForge started, developers familiar with many programming techniques are likely to have joined. Some developers may have been new to programming when SourceForge started and joined, but many more developers were likely experienced developers and joined SourceForge. The experienced developers may have been familiar with programming styles popular previously such as cobalt and linear programming. In contrast, the set of developers that join the platform more recently, are likely dominated by new developers, and are more likely to be familiar with newer techniques including using classes and Java. The ideas from the developers who joined SourceForge earlier could conflict with the ideas of the newer developers and take extensive amounts of time for the development group to resolve or lead to developers quitting the project. Consider the text below where a new developer on a project attempts to start a new project because the developers who have been on the project longer are not responsive to his desire to rework the project architecture.

New Developer: *"the [] leadership of Thormod have been very unreceptive to a variety of proposals to rework the architecture over the past year."[…] I have concluded that the best hope for Thormod is to create a new project that starts with the existing code base but is completely independent of the current Thormod project leadership.[…]"*(Divitini et al. 2003)

Elliott and Scacchi (2003) document how conflicting opinions concerning the integration of proprietary supporting software for the GNU Enterprise project result in online discussions that last a day or more. Further, these discussions may not be handled effectively because the computer-mediated technology used by the OSS development group reduces the ability of the development group to resolve conflict (Dreu et al. 2003). With the limited ability to resolve conflict, multiple ideas about the options for development styles and techniques can easily lead to developers leaving the group (Jackson 1992; McCain et al. 1983). In an OSS environment where participation is voluntary, developers can start a different version of the project for their personal use, and if they do, the activity should decrease because the developers are no longer making official contributions to the OSS project.

Diversity in development related knowledge might make it difficult to agree on an overarching plan because the developer's different backgrounds lead to different plans for development. An overarching plan enables the development group to understand the importance of development tasks that are mundane. If developers have different plans for development, although some tasks may get done, it will be difficult to get some of the more mundane tasks completed because there is no larger vision that explains the importance of those tasks. This should lead to low levels of development group activity intensity.

Many diversity studies have examined linear effects of diversity, but recent research suggests that the relationship between diversity and performance is non-linear (Earley et al. 2000; Gibson et al. 2003; Vegt et al. 2005). Extreme levels of platform tenure diversity are likely associated with special characteristics. OSS research points toward the fact that learning is important in OSS communities, and new developers seek to learn from experienced OSS developers (Ye and Kishida 2003). New developers bring innovative ideas about new programming techniques. At times these ideas are appropriate and offer ways for the project to create improved applications and therefore lead to activity. But, other times they are not appropriate and may cause the project to split because of disagreement, for example. In these cases if the long tenure developers disagree about the appropriate nature of the technique for the project they can hinder the project from going in that direction. In this case, the short tenure developers are likely hesitant to abandon the project because, if learning is important as prior research suggests (Ye and Kishida 2003) they desire to learn from the long tenure developer(s). So, if there are developers with long tenures on a project and very short tenure developers there may be a hierarchy of control that minimizes the conflict resulting from platform tenure diversity. In these cases the diversity can be effectively managed by allowing the innovation from new developers when appropriate and offering learning opportunities to new developers.

This logic suggests that high levels of platform tenure diversity should lead to high levels of development group activity intensity.

Several OSS projects have a single developer; Capiluppi, Lago and Morisio (2003) find that 49% of SourceForge projects in their random sample have a single developer. In these projects conflict is not likely to slow development group activity. Other OSS projects represent teams that work together off line and join the platform at the same time to develop an application. These teams are not limited by the computer mediated communication available on SourceForge as they seek to resolve conflict. In this case they are able to employ richer forms of communication, such as face to face, that make it easier to resolve conflict (Poole et al. 1991). Hence, when there is very low platform tenure diversity conflict should be very low or able to be minimized and the development group activity intensity is expected to be high.

Based on the logic derived above I expect high diversity to yield high development group activity intensity because of the ability to manage conflict and the importance of learning in the OSS community. Medium values result in low development group activity intensity because of high conflict and constraints on the ability to resolve conflict. Finally, low levels of platform tenure diversity should lead to high development group activity intensity because of the limited conflict. Taken together I expect a U-shaped pattern where extremely low and high levels of platform tenure diversity are associated with high development group activity intensity and medium values have low development group activity intensity. Therefore, I propose:

Hypothesis 5: Development group platform tenure diversity has a quadratic association with development group activity intensity.

4.3 Relationship Density

Relationships with other organizations represent channels through which an organization can acquire knowledge (Cohen et al. 1990; Hippel 1994). Through relationships with other organizations developing similar products, an organization can gather important knowledge about success and failure in the industry. From such relationships, an OSS project can learn about successful programming techniques or opportunities to add features. After learning the programming technique, the development group can add more features and fix more bugs, which is an increase in development group activity. Thus, the knowledge acquisition capability reflected in relationships with other OSS projects should lead to development group activity intensity.

4.3.1 IBUC Relationship Density

An IBUC member is able to participate in more than one OSS project. Participation with an OSS project indicates an IBUC relationship with the project. IBUC relationships with other OSS projects suggest increased IBUC familiarity with OSS applications. Familiarity with OSS projects enables the IBUC to acquire knowledge about potential features that are easier for developers to develop, when compared to features that the IBUC experiences using proprietary software. When the IBUC requests a feature that has been implemented in another OSS application, it is easier for the development group to develop because the development group can re-use the code from the original project. For instance, an IBUC member suggested that the development group on the project ZK on SourceForge write "not just to Dhtml but to Flash" and look at code from OpenLaszlo, another project on SourceForge to find an example of how to write to flash. Thus, IBUC knowledge from relationships with other OSS projects facilitates the addition of features, i.e., it amplifies development group activity.

In addition to feature suggestions that are easy to develop, IBUC relationships with OSS projects are likely to lead to increased development group activity because of opportunities to collaborate between two OSS projects. These opportunities are likely to be different from those opportunities that the development group knows about. The set of OSS projects that an IBUC is related to is apt to be driven by business context and so the IBUC relationships can provide knowledge about potential collaborations between OSS projects that develop applications for use by those in a particular business. For instance, instead of being related to projects that all use the java programming language, as the development group may be, the IBUC could be related to projects that provide software applications used by researchers. In this case, the IBUC is better able to suggest features that enable software compatibility between software that researchers use. Thus, IBUC relationships indicate a knowledge acquisition capability that provides opportunities for activity that are likely to be different from the knowledge acquired by the development group.

In summary IBUC relationships can facilitate opportunities for development that other OSS projects can be leveraged to implement, and are distinct from ideas the development group has considered. Based on this logic I propose:

Hypothesis 6: IBUC relationship density is positively associated with development group activity intensity.

4.3.2 Development Group Relationship Density

Those who perform the same job across shifts in a manufacturing facility can increase performance by having relationships with each other (Epple et al. 1991). The benefits of relationships between workers on different shifts are derived because the workers exchange knowledge about how to improve performance and I assert that there are similar benefits for developers who have relationships across OSS projects. Development group relationships indicate the ability to acquire knowledge focused on software development from other projects. Grewal et al. (2006) document how OSS developers participating on multiple projects, because they represent channels of knowledge flow between projects, can lead to a higher number of CVS commits.

OSS development group relationships with other OSS projects can easily lead to the acquisition of knowledge about software development. Specifically, the development group can acquire knowledge related to success and failure using programming techniques, resources and developer skills. Knowledge from other OSS projects is likely to be especially relevant because the source code is accessible, which may offer insights into how to add features and correct bugs. In addition, development group relationships with other projects may identify the skills possessed by developers on other projects. Such knowledge is critical for identifying new developers with specifically needed development skills. These developers may be recruited, thus leading to increased development group activity.

The acquisition of knowledge related to software development styles, developer skills and resources, facilitated by project relationships, enables the development group to add features and fix bugs. Thus, development group relationships reflect a knowledge acquisition capability that is likely to lead to development group activity. So I propose:

Hypothesis 7: Development group relationship density is positively associated with development group activity intensity.

4.4 IBUC Dynamism

The dynamism of an IBUC is a reflection of the degree of change that occurs in its membership. Over time after the initial release, the development group will implement all of their ideas for the development of the software application. At this point the project could become stagnant or unsure of the correct improvements or adaptations to make to the software to keep it useful in the future or to other users. As the IBUC offers suggestions, the IBUC gives the project new ideas for tasks to perform. These suggestions are drawn from different perspectives compared to those already in the OSS project, and so there is an improved chance that these tasks will represent new opportunities for development. This gives the development group an opportunity to attempt development that is both new and useful, so they are likely to respond to the requests of a dynamic IBUC. As the IBUC dynamism increases, the IBUC should be better positioned to acquire knowledge that leads to higher levels of development group activity because there will be new options for the development group to exploit their knowledge. Hence, I propose:

Hypothesis 8: IBUC dynamism is positively associated with development group activity intensity.

4.5 Application Type

A project that develops an application targeted to developers has an IBUC that is composed of software developers. An example of an application targeted to developers is Pred. Pred is a Java based graphic utility to find and edit Java property files. When the IBUC is composed of software developers the IBUC members have computer based technical knowledge. Subsequently, the common knowledge and shared cognitive structures of the development group and IBUC is greater for projects that develop applications targeted to developers than when the project develops a software application that is not targeted to developers. When the IBUC and development group have more overlapping knowledge, such as if the application is targeted to developers, they have a shared "language" and therefore it should be easier for them to communicate. Because the IBUC and the development group for more technical software have shared language and cognitive structures, the transfer capability of these projects is expected to be stronger. Thus it should be easier for an IBUC composed of software developers to make use of their knowledge acquisition by transferring their knowledge to the development group. The software application developed by the project therefore should affect the strength of the relationship between IBUC knowledge acquisition capability and development group activity.

The IBUC is expected to have a broader set of knowledge than the development group because the coordination costs for participation in the IBUC are lower than in the development group. As a result the IBUC may have ideas for software enhancements that are different from those that are in the development group. However, when the application type is for developers, the development group should understand the requests from the IBUC better and therefore be better prepared to address those requests. Therefore the IBUC and development group should be able to work together easily to identify a broad set of development group activities that match the skills and interests of the development group. In fact, the interaction of software developers in the IBUC with the development group can explain the success of the most popular OSS projects, such as Linux and Apache, because these are projects where the users are software developers. Both projects benefit from the suggestions of IBUC members who, because of the nature of the software developed, are technical. For these reasons I suggest:

Hypothesis 9: The positive relationship between IBUC relationship density and development group activity intensity will be stronger for application types that target developers.

Hypothesis 10: The positive relationship between IBUC dynamism and development group activity intensity will be stronger for application types that target developers.

4.6. IBUC-Development Group Communication

Communication between the IBUC and the development group represents the project knowledge transfer capability and may strengthen the relationship between IBUC knowledge acquisition and development group activity. Such communication can alter the relationship between IBUC knowledge acquisition and development group activity by allowing more IBUC knowledge to reach the development group, and enabling the emergence of ideas that are the result of the combination of IBUC and development group knowledge.

Higher levels of communication between the development group and IBUC facilitate knowledge transfer – in essence, they enable higher levels of knowledge acquisition. Some technical ideas are difficult to transfer and require repeated communication. For these kinds of ideas more frequent IBUC and development group communication will increase the likelihood of knowledge transfer to the development group and thus lead to development group activity.

Higher levels of communication between the development group and IBUC also allows for the emergence of new knowledge that benefits from the unique knowledge in the two participant sets. Consistent with Nambisan et al. (1999), I argue that innovation for OSS projects occurs at the “confluence of business expertise and technical mastery.” In this way the relationship between IBUC knowledge acquisition and development group activity will be strengthened for projects that have higher levels of communication. I therefore propose:

Hypothesis 11: IBUC-development group communication will positively moderate the positive relationship between IBUC relationship density and development group activity intensity.

Hypothesis 12: IBUC-development group communication will positively moderate the positive relationship between IBUC dynamism and development group activity intensity.

In contrast to its effects on the relationship between IBUC dynamism and development group activity, an increase in knowledge overlap between an IBUC with high platform tenure diversity and a development group may intensify the negative effects of IBUC platform tenure diversity. Knowledge transfer can have a negative effect on performance when the knowledge transferred is inappropriate for and cannot be applied in the context (Baum et al. 1998). When the knowledge is diverse and the transfer is strong it may be difficult for the development group to ignore potential divergent paths introduced by the IBUC. In this case, the IBUC knowledge acquisition could result in a forking of the project and the departure of some developers, thereby suppressing activity. I therefore propose:

Hypothesis 13: The negative relationship between IBUC platform tenure diversity and development group activity intensity will be more intense when there is strong IBUC-development group communication.

Hypothesis 14: The negative relationship between IBUC platform tenure diversity and development group activity intensity will be more intense for applications targeted to developers.

4.7 Summary

In Chapter 4 I developed a research model that describes the effect of an OSS project's absorptive capacity on its development group activity intensity. In this model, there are two dimensions of absorptive capacity that were argued to influence development group activity intensity: knowledge acquisition capability and knowledge transfer capability. Three characteristics of participant sets were proposed as indicators of knowledge acquisition capability because they are reflections of the knowledge base within the project. They are platform tenure diversity, relationship density and IBUC dynamism. Hypotheses were developed relating these participant characteristics to development group activity intensity. Two indicators of an OSS project's knowledge transfer capability were discussed as moderators of the relationship between the IBUC enabled knowledge acquisition capability and development group activity intensity. Application type and development group communication with the IBUC indicate the knowledge transfer capability. The next chapter describes the research methodology used for testing the research models presented in Chapters 3 and 4, and includes results from the empirical study.

CHAPTER 5: METHODOLOGY

This chapter presents the research methodology used to empirically test the research model developed and hypotheses proposed in Chapters 3 and 4. It also presents the results of the analysis. Section 5.1 outlines the methods for the study including details of models used, sample selected, variable measurement and sensitivity analysis conducted. Section 5.2 discusses results for models that focus on tasks closed, and Section 5.3 presents results based on change in lines of code as the measure of development group activity.

5.1 Methodology Overview

Development group activity can be measured in multiple ways. In order to capture a more complete picture of development group activity, the methods described in this section explore two measurements of it. Development group activity is considered in terms of the tasks closed and change in lines of code. Tasks closed and the changes in lines of code are used to measure the development group activity because they capture different dimensions of it. Tasks closed captures development group activity from the user's perspective, but does not capture the amount of resources or time used to complete the task. Closing a task could mean fixing a bug by changing the spelling of a word in the code or it could mean creating many new lines of code. Because it is not clear how much developer time is required for a task, closing tasks is not always a direct representation of developer effort. While the change in lines of code is captures more about the effort required from the developers, change in lines of code may not capture evolution of the software from the perspective of the IBUC.

A pilot study was conducted to asses the viability of measuring the variables of interest and to guide the final study. In particular, the pilot study was used to get an initial understanding of the relationships between the variables of interest and also to confirm the availability of data. The pilot study included a sample of OSS projects from SourceForge and it was verified that all measurements could be gathered. Most variables were examined in the pilot study empirical analysis; the changes in lines of code and application type were not. During the pilot study, logistic regression was used to examine the survival model developed in Chapter 3. Moderated and linear regression were used for the development group activity intensity model developed in Chapter 4.

Based on feedback received about the pilot study a Cox proportional hazard rate model was used for the OSS project Survival model instead of logistic regression. The Cox proportional hazard rate model allows an understanding of the richness in the continuous outcome variable compared to using logistic regression that restricts the outcome variable to being binary. The Cox model provides the conditional probability of OSS project death being later than a specified time given the independent variable values. The Cox proportional hazard rate model is used for both the tasks closed and the change in lines of code. The R Statistical Package Version 2.2.1 is used for this analysis.

Consistent with the pilot study, to examine the development group activity intensity model developed in Chapter 4, linear and moderated regressions are used for both the percentage of tasks closed and change in lines of code. SPSS Version 14 is used for these regressions.

5.1.1 Sample Selection And Data Collection

The sample of OSS projects is selected in order to minimize the effect of factors outside the focus of this research. I minimized the effect of the following factors: OSS development platform, time since the OSS project registered on SourceForge, programming language, and type of communication. This section describes each of these factors.

Different OSS development platforms may facilitate development group activity more than others based on the resources that the platform offers to developers and the ease of use of those resources. For instance, a platform that facilitates communication between developers and users by offering public forums may increase OSS project performance compared to a platform that does not offer public forums. I include only projects registered on SourceForge in my sample to control for the effect that the OSS development platform has on OSS project performance. SourceForge is one of the most popular OSS development platforms. SourceForge allows developers and users to observe other developer and user activities, send bug reports, feature requests and patches, post to forums, launch new OSS projects and join existing ones, coordinate and work jointly in specific OSS projects, and integrate the software they produce into a larger piece of software.

The effect on OSS project performance that may be related to the time the OSS project registered on SourceForge is minimized by selecting a sample of OSS projects registered on SourceForge between June 2004 and August 2004. It was important to select OSS projects that registered in a discrete time period in order to avoid the confounding effects due to a particular event. For example, there may have been a surge in OSS interest related to the release of an updated programming language package or proprietary releases such as when Netscape opened some of its source code.

Some programming languages require more code to accomplish the same task and since I consider change in lines of code as a measure of development group activity it is important to control for the programming language. I select only projects that use the Java programming language.

Finally, I minimized the effects that may be due to the type of communication used between the development group and IBUC. Only OSS projects that exhibited observable knowledge transfer between the development group and IBUC on public forums are included in the sample. A lack of observable transfer between the development group and IBUC in public forums suggest that the OSS project primarily uses some other form of computer-mediated communication. Because this study did not observe other forms of computer mediated communication, such OSS projects would confound the results.

5.1.2 Observation Periods

Archival data for each project is observed for the first 810 days after each project is registered on SourceForge. Because OSS projects start on different days, the 810 days do not correspond to the same calendar days for each project. In order to test for causation, it is beneficial to observe the independent variables in a time period that precedes the time period in which the dependent variables are observed. Therefore, there are two observation periods used in this research. Period one is 180 days and starts the day the OSS project registers on SourceForge. Otherwise stated, period one includes the first 180 days after the OSS project is registered on SourceForge. Period one precedes period two and the independent variables are observed during period one. Period two is 630 days. It starts on the 181st day after the OSS project is registered on SourceForge and ends on the 810th day after it is registered on SourceForge. The development group activity intensity variables are calculated based on events that occur during period two. A graphical depiction of these periods is found in Figure 4.

5.1.3 Dependent Variable

This section details the measurement of the dependent variables that represent development group activity considered in this research. For the survival model developed in Chapter 3 the outcome is time to OSS project death, and it is measured in terms of both tasks closed and change in lines of code. For the model developed in Chapter 4 the outcome is development group activity intensity. Development group activity intensity is measured in terms of the percentage of tasks closed and the amount of change in lines of code. The details of each of the measures of performance are discussed below.

As discussed above development group activity is considered in terms of the tasks closed and change in lines of code. Tasks can be opened by users or developers on the project, and include primarily features requests, support requests and bug reports, but there are occasionally user or developer defined categories. After opened, tasks can be closed by developers. In order to assess the change in lines of code all releases for all projects in the sample were downloaded and analyzed. The lines of code are assessed using the software package Understand. For each release of the project the Count Line Code field is used as a measure of the lines of code for that release of that project. Change in lines of code is the difference in lines of code between two releases.

Time To OSS Project Death. The time to OSS project death is parameterized by the survival function which is the probability that the death of the OSS project is later than some specified time. Both the time over which the OSS project is observed and the time of death for the OSS project are needed to calculate the survival function.

An OSS project can fall into one of two categories that determine how the length of time over which the OSS project is observed is measured. If the OSS project does not die while it is observed, then the length of time over which the OSS project is observed is 810 days. 810 days represents the time the project that registered the latest was observed and so I observe 810 days for all projects to be consistent. If the OSS project dies, then the length of time over which the OSS project is observed is the number of days between the death of the project and the day the project registered on SourceForge. Time to OSS project death is measured in terms of tasks closed and changes in the lines of code. The use of each measure is elaborated below.

Determining the death of an OSS project is complex because, theoretically, at any time the original or new developers can begin developing on the OSS project, thus bringing the project back to life. Prior research has suggested that OSS project death can be determined based on the length of time that a project is inactive and this dissertation follows this line of research (Chengalur-Smith et al. 2003). The research assumes that a long inactive period indicates that a project will not be active again in a later time period and is dead. A project is inactive, likely because there is no developer or user interested in writing code or requesting features concerning the OSS project. If an OSS project is inactive longer than the average inactive period in my sample of projects plus two standard deviations, I assume the OSS project is not likely to garner interest from developers or users in the future and is dead.

Time To OSS Project Death – Tasks Closed. When considering tasks closed, death of an OSS project is defined as inactivity, no tasks closed in that OSS project, for 490 days. So, when an OSS project does not complete any tasks over a 490-day period, the time of death is set to the first day of that 490-day period. 490 days is used for a time period to determine death based on analysis of the data for the projects' typical inactivity periods. On average, based on tasks closed, the projects are inactive for 70 days at a time and the standard deviation is 210 days. Therefore, if a project is inactive for the average number of days plus two standard deviations, this research deems the OSS project dead.

Time To OSS Project Death - Change In Lines Of Code. For change in lines of code, death of an OSS project is defined as no change in lines of code in that OSS project for 350 days. This death cut-off of 350 days is determined in the same way that the death cut-off is determined for death using tasks closed. 350 is the average inactivity period based on changes in lines of code plus two standard deviations. If a project has multiple streams, such as a client and server all releases both from the server and client stream are included in this analysis to determine inactivity periods.

Development Group Activity Intensity. The outcome variable for the linear and moderated regression is development group activity intensity. It is a measure of how much developer activity there is on the project in observation period two. This outcome is measured in terms of both the percentage of tasks closed and change in lines of code as discussed below.

Development Group Activity Intensity – The Percentage Of Tasks Closed. The percentage of tasks closed is measured as the percentage of tasks closed out of those available to be closed during period two. This is similar to the measure that Stewart and Gosain (2006) use as a measure of OSS project performance that they name tasks completed. They explore the percentage of tasks closed over the life of the project, but this research considers only period two. This research uses the term task closed and not completed to acknowledge that closing tasks does not always indicate completion.

Development Group Activity Intensity – Change In Lines Of Code. The change in lines of code is consistent with the measure that Fershtman and Gandal (2004) use for OSS project performance. The number of lines of code in the release closest to, and before the start of period two is subtracted from the lines of code in the release closest to, and before, the end of period two to measure the change in lines of code. If there is no release before the start of period two, then the earliest release is used as the first release. If there are not two releases the change in lines of code is zero. If there is no release after the start of period two then the change in lines of code is zero. If an OSS project had releases that appeared to be in different streams, such as a client and server, only one of the streams was used to measure the change in lines of code. Using only one stream is consistent with the way Stewart et al. (2006) handle multiple development streams under one OSS project on SourceForge.

5.1.4 Independent And Control Variables

This section first discusses measurement of the independent variables that represent knowledge acquisition capability. The independent variables include development group and IBUC platform tenure diversity, development group and IBUC relationship density and IBUC dynamism. Then I discuss two variables that measure the knowledge transfer capability; IBUC-development group communication and application type. Finally I discuss the measurement of the control variable, the number of project members.

Platform Tenure Diversity. Like other technologies, while SourceForge enables many processes it also constrains the ways that these processes are accomplished. The SourceForge platform represents a community and technological platform that facilitates the development of norms for those who participate on SourceForge. The longer a person engages with SourceForge the more a person's thought patterns, languages and methods of software development will become consistent with the norms common on SourceForge. A person with a shorter SourceForge tenure is likely to have different ideas about software development norms compared to a person with longer SourceForge tenure, as an example. For these reasons a diverse tenure with SourceForge represents a diversity of knowledge.

To calculate a measure of development group and IBUC platform tenure diversity, each participant's tenure is measured by his or her time with the SourceForge platform. Time with SourceForge is measured by the number of years since each user registered on SourceForge. IBUC platform tenure diversity is the variance of the time on SourceForge for the members of the IBUC and development group platform tenure diversity is the variance of the time on SourceForge for the developers.

Relationship Density. The number of relationships with other OSS projects measures IBUC and development group relationship density. For the IBUC, it is the sum of the number of OSS projects with which the IBUC has relations. If two IBUC members are related to the same OSS project, the OSS project is counted once. A relationship between an IBUC member and an OSS project is defined by the IBUC member developing within an OSS project or interacting through SourceForge enabled computer mediation with an OSS project. This includes an IBUC member submitting a bug to a project or requesting a feature. Relationships that are based on computer-mediated communication with other OSS projects are observed in observation period one. All relationships defined by development are observed December 2004, which is in period one for all projects in the sample. The development group relationship density is calculated in the same way the IBUC relationship density is calculated, but for the developers on the project instead of the IBUC members.

IBUC Dynamism. IBUC dynamism is the number of members of the IBUC observed during the first half of period one (the day the project registered on SourceForge and the 90 days after the registration day) divided by the number of IBUC members during all of period one (the day the project registered on SourceForge and the 180 days after the registration day).

IBUC-Development Group Communication. Public forums for each OSS project are explored because they are one way that the development group and IBUC communicate (Long et al. 2005). A public forum is composed of one or more threads. A thread is composed of at least one post, but may also contain response posts from the person who posted the original post or a new person. A post is a message written by an OSS project participant that can then be seen by anyone. A thread that contains a post from a developer and an IBUC member suggests knowledge transfer between the development group and IBUC. All posts in threads that contain a post from both the development group and IBUC during period one are counted.

Application Type. To identify whether OSS projects are designed to be used by developers, I explore the topic field on SourceForge for each OSS project. If the topic is development then I conclude the project is designed to be used by developers.

Number Of Project Members. This is the number of participants associated with the project and includes both IBUC members and developers. SourceForge userids are used to identify those who are a part of the IBUC. The IBUC for each OSS project is identified using userids that reported bugs, requested features or support during observation period one. Those listed as developers or administrators during December 2004 for the OSS project are excluded from being a part of the IBUC and are identified as developers. Each OSS project in the sample registered on SourceForge between June, 2004 and August, 2004 and so exploring the number of developers in December 2004 suggests the number of developers after the OSS project has had time to establish itself.

5.1.5 Sensitivity Analysis

Sensitivity analysis is done to evaluate how robust the model is and in particular the impact of choices made in the measurement of the variables. The effect of the time period over which the variables are observed is considered by shifting time period two 1 month forward. No difference in the results is found. Also, I examine whether the number of developers changes over the observation period and find that the number of developers does not change for the majority of the OSS projects in the sample.

5.2 Tasks Closed Results

The base data for the OSS projects are drawn from a database provided by SourceForge to the University of Notre Dame. The sample restrictions above yield a sample of 92 projects and descriptive statistics for the sample of 92 OSS projects were calculated. The analysis of the distributions of variables identified several of the variables as skewed and 7 projects as outliers. When examining the measure for IBUC-development group communication 5 projects were outside of 2 standard deviations around the mean and were deleted from further analysis. Similarly, one project was outside of 2 standard deviations based on the development group relationship density and one was an outlier based on IBUC dynamism. These restrictions yield a sample of 85 projects that are the focus of the analysis where tasks closed is used to measure development group activity.

These projects represent the development of a broad range of application types (see Table 2 for detailed descriptions). Included in the sample are projects that develop applications to aid in software development, management, statistical analysis, education, online sales and entertainment. 16 (18.8%) of the projects develop applications where the software is designed to assist with software development. Some of the applications replicate more traditional applications, such as one that replicates Microsoft Office. Other applications are developed to work in conjunction with traditional applications. Examples include G4j, a product designed to work with Gmail, and Amtu, which is intended to work with Amazon.

Descriptive statistics of the variables of interest for the 85 projects are shown in Table 3 (Table 7 for 73 projects). Many of the variables measured in this study were developed specifically to explore the constructs in this model and have not been observed in prior studies. For this reason the descriptive statistics of most of the variables cannot be compared with what other researchers have found. Some variables are similar to those studied in prior research and I discus how the variables I observe in this study compare to the observation of similar variables in prior studies. I continue by comparing the number of developers and the percentage of tasks closed observed in my sample compared with prior studies.

The number of developers on OSS projects using the SourceForge development platform has been observed in prior studies. Crowston and Scozzi (2002), for example, examined a sample of 7,477 OSS projects from SourceForge and found the mean number of developers to be .9. Stewart and Gosain (2006) examined a much more restricted sample of 67 projects from SourceForge. They restricted their sample to projects in the categories of communications or multimedia, required the OSS projects to have at least 4 developers, and development group activity in the current week. These restrictions led to a mean of 8.2 developers. Similarly, Krisnamurthy (2002) observed a very restricted sample of 100 projects from SourceForge that were the most active and in the mature stage. Krisnamurthy found the mean number of developers to be 6.6. The different restrictions used to identify the sample of OSS projects leads to different means for the number of developers.

The mean number of developers for the 85 OSS projects in this research is 3.06. 44 of the projects in this sample have 1 developer. When considering the restrictions used to gather the current sample compared to restrictions used in prior studies the mean number of developers is consistent with prior studies. Specifically the mean is higher than Crowston and Scozzi who sampled all projects, but lower than the other two studies that used strict requirements to include only high activity projects in the sample.

With respect to the percentage of tasks closed, few studies have observed this exact measure, but as mentioned above Stewart and Gosain (2006) used a measure similar to the percentage of tasks closed. The difference between their measure and the measure used in this research is that Stewart and Gosain (2006) observed tasks closed across the OSS project's entire life span, while this work explored the percentage of tasks closed in period two. Stewart and Gosain (2006) found the mean percentage of tasks closed to be .42, while this work found the mean percentage of tasks closed to be .24 for 85 projects (.27 for 73 projects). The difference between this work and that of Stewart and Gosain (2006) is likely the result of the stricter sampling criteria Stewart and Gosain (2006) used to include active projects and also a result of the difference in observation periods. The distribution of the percentage of tasks closed variable is right skewed and so the variable is transformed. After an exponential transformation, the value representing the level of skewness for the distribution of the percentage of tasks closed is below the recommended value of 1.

The correlation analysis, shown in Table 3, suggests that IBUC platform tenure diversity, IBUC and development group relationship density and IBUC dynamism have strong positive association with the percentage of tasks closed. In addition, the correlation analysis suggests some interesting associations that were not expected. Development group relationship density is positively associated with IBUC platform tenure diversity, IBUC relationship density and IBUC dynamism. Perhaps a development group with strong relationships is able to consistently attract IBUC members with diverse backgrounds and strong relationships. Essentially, a developer relationship with another OSS project represents a channel through which a potential IBUC member can learn about the OSS project. Further, the number of project members is correlated with many of the other variables. This is consistent with a resource based view that would suggest more people provide more resources that can facilitate OSS project performance (Butler 2001). IBUC-development group communication is also positively correlated with both change in lines of code and percentage of tasks closed. Finally, of note is the low correlation between the two measures of development group activity; percentage of tasks closed and change in lines of code. This indicates that there are two distinct types of activity that should be explored separately.

5.2.1 Survival Analysis - Tasks Closed

In observing the sample of 85 projects over the course of 810 days, 64 of them die, or have a period of 490 days with inactivity. The other 21 projects never have a period of inactivity lasting 490 days and therefore do not die during the time I observe them. The Kaplan-Meier survival function is pictured in Figure 5. This figure displays that approximately 50% of the projects are active a little over one year. This is consistent with the Stewart et al. (2006) finding that most activity occurs before the 400th day after the OSS project registers on SourceForge.

The results of the Cox proportional hazards rate model are displayed in Table 5. Positive coefficients indicate greater hazard of OSS project death, and so coefficients that have a negative sign have positive effects on survival. This model provides evidence for a positive relationship between IBUC platform tenure diversity and OSS project survival (Support for H1). There is also evidence for a positive relationship between IBUC dynamism and OSS project survival (Support for H3). However, there is not evidence of the positive relationship between development group platform tenure diversity and survival based on the percentage of tasks closed.

5.2.2 Development Group Activity Intensity – The Percentage Of Tasks Closed

Four ordinary least square regressions were examined to empirically explore the development group activity intensity model developed in Chapter 4. The results of these regressions are shown in Table 6. The first regression (Model A) examines the main effects between the development group and IBUC knowledge acquisition capabilities and the development group activity intensity measured by the percentage of tasks closed. Specifically, Model A is used to test Hypotheses 4, 6 and 8 because they cannot be tested in the models that include interactions. The effect of an interaction between the knowledge transfer capability, measured by IBUC-development group communication, and the IBUC knowledge acquisition capability on development group activity intensity measured by the percentage of tasks closed is explored in the second regression (Model B). The third regression explores how application type changes the effect of IBUC knowledge acquisition capability on development group activity intensity measured by the percentage of tasks closed (Model C). Model B and C are useful to understand if both knowledge transfer variables are needed. Specifically, they are used to understand if adding the second knowledge transfer variable explains more variance in the outcome than using only one knowledge transfer variable. The fourth (Model D) and final regression includes all main and interaction effects. Assuming Model D explains more variance than either Model B or C, it will be used to test Hypotheses 5,7 and 9-11. The main effect relationships between development group characteristics and the outcome can be tested in the model with interactions because the development group characteristics are not included in interactions.

All four regressions satisfy the assumptions of the linear regression model. After transformation, the skewness value was less than the recommended level of 1 for the dependent variable, the percentage of tasks closed. The skewness value indicates that the transformed distribution of percentage of tasks closed is not significantly different from being normally distributed. I centered all variables to reduce the possibility of multi-collinearity resulting from the inclusion of interaction terms for testing the moderation hypotheses. The model variance inflation factors (all below 4) and the correlations between the independent variables were observed and they suggest that the model does not have a problem with multi-collinearity. The Durbin Watson value showed no evidence of serial correlation (under 3). Observation of the residuals and Cook and White's test suggest that the errors are normally distributed (Cook 1979; White 1980).

The first regression in Table 6, labeled Main Effects, is significant with a P-value .00 and R-square 30%. The results of the regression show that IBUC platform tenure diversity, development group relationship density, IBUC dynamism, application type and IBUC-development group communication have statistically significant positive relationships with the percentage of tasks closed (Significant – Wrong Direction H4, Support for H7 and H8). Development group platform tenure diversity, IBUC relationship density and the number of project members all have statistically insignificant relationships with the percentage of tasks closed.

The second regression in Table 6, labeled Moderated Effects IBUC-Development Group Communication includes interactions between the IBUC knowledge acquisition variables and the measure of knowledge transfer, IBUC-development group communication. According to the R squared, this regression is significant and explains 36% of the variance in the percentage of tasks closed (p<.01). The change in R squared compared to the main effects model is significant (p<.01). This model suggests that the relationship between IBUC relationship density and the percentage of tasks closed is altered by IBUC-development group communication (Significant – Wrong Direction H11). There is also evidence that IBUC-development group communication alters the relationship between IBUC dynamism and the percentage of tasks closed (Significant – Wrong Direction H12). There was not evidence that the relationship between the IBUC platform tenure diversity and the percentage of tasks closed was changed by the IBUC-development group communication.

A graphical depiction of the interaction between IBUC relationship density and IBUC-development group communication was developed to further investigate this relationship. The picture of this interaction in Figure 6 shows that for high IBUC-development group communication, IBUC relationship density has a more steeply negative association with the percentage of tasks closed compared to projects with lower IBUC-development group communication.

Figure 7 depicts the interaction between IBUC dynamism and IBUC-development group communication. IBUC dynamism has a positive association with the percentage of tasks closed for projects with high and low IBUC-development group communication. However, smaller increases in IBUC dynamism improve the percentage of tasks closed more in projects with low IBUC-development group communication compared to projects with high IBUC-development group communication.

The third regression in Table 6 includes interactions between the IBUC knowledge acquisition variables and the measure of knowledge transfer, application type. It is significant and explains 35% of the variance in the percentage of tasks closed according to the R squared. According to the third regression application type alters the relationship that both IBUC relationship density and IBUC platform tenure diversity have on the percentage of tasks closed. Both interactions are significant (Significant – Wrong Direction H9 and Support H14) and the change in R squared, compared to the Main Effects model is significant (p<.01). There was not evidence that the application type moderated the relationship between IBUC dynamism and the percentage of tasks closed.

A graphical depiction of the interaction between application type and IBUC platform tenure diversity is shown in Figure 8. For projects that develop applications designed for use by developers there is a negative relationship between IBUC platform tenure diversity and the percentage of tasks closed. However, for projects that develop applications for use by end-users IBUC platform tenure diversity has a slightly positive effect on the percentage of tasks closed.

Figure 9 displays the interaction between application type and IBUC relationship density. It shows that for projects that develop applications for use by developers there is a positive relationship between IBUC relationship density and the percentage of tasks closed. Yet for projects that develop applications for use by end users IBUC relationship density has a negative effect on the percentage of tasks closed.

The final column in Table 6 shows the results of a regression with the percentage of tasks closed as the outcome and all interactions and main effects. This model is significant and explains 36% of the variance in the percentage of tasks closed (p<.01). The change in R between this model and Model B and Model C is significant, suggesting Model D is a better fit than

Model B or C (p<.05). Model D is used to test all hypotheses. This model provides support for many of the same interactions as the previous models. The exception is the relationship the interaction between IBUC-development group communication and IBUC relationship density has on the percentage of tasks closed. It was significant when the application type interactions were not in the model, but it is not significant when application type interactions are included in the model.

5.2.3 Summary

Although OSS has established itself as a key player in the software industry the factors that allow these projects to perform well in terms of development group activity are unclear. By drawing on an absorptive capacity perspective this section sheds light on the factors that lead to performance in terms of the tasks closed for OSS projects. In summary, the analysis using the percentage of tasks closed as the measure of development group activity provides evidence to support the conceptual model and both research models developed in this dissertation. The first column in Table 11 provides a summary of the hypotheses, and documents those hypotheses that are supported by the analysis of the percentage of tasks closed as the measure of development group activity.

5.3 Change In Lines Of Code Results

5.3.1 Survival Analysis – Change In Lines Of Code

The sample of 85 projects, discussed above, includes projects in various stages of development. When projects are in some stages of the development process, although they may accomplish tasks such as develop plans or manuals, they typically do not change the lines of code. In the planning stage of development, projects are typically focused on project planning and design. In this case, because the goal of the project is not to produce source code, change in lines of code is not an appropriate measure of performance. Therefore, when change in lines of code is used as the measurement of development group activity intensity, projects that are in the planning stage are not included in the sample. There are 10 projects in the planning stage. There are 2 projects that do not have a stage listed and they are also eliminated, since their stage cannot be determined. Subtracting those 12 projects yields a sample of 73 projects; it is used for the analysis when change in lines of code is the measure of development group activity intensity. Descriptive statistics of the 73 projects are shown in Table 7 and the correlations are in Table 8.

In observing these 73 projects over the course of 810 days, 56 of them die, or have a period of 350 days with inactivity measured by a lack of change in lines of code. The other 17 projects never have a period of inactivity lasting 350 days and therefore do not die during the time observed. The Kaplan-Meier survival function is pictured in Figure 10. It displays that approximately 50% of the projects last a little less than 100 days. This is an artifact of some OSS projects making an initial release and then not releasing again. The flat portion of the function toward the right side of the plot reflects the 17 OSS projects that last the full 810 days of observation.

The results of the Cox proportional hazards rate model are displayed in Table 9. Positive coefficients indicate greater hazard of OSS project death. This model provides evidence for a positive relationship between IBUC dynamism and OSS project survival (Support for H3). This model also finds an unexpected significant relationship between the application type and OSS project survival. There is not a relationship between the other variables and OSS project survival measured by changes in lines of code.

5.3.2 Development Group Activity Intensity – Change In Lines Of Code

As in the exploration of development group activity measured by the percentage of tasks closed, four ordinary least square regressions were examined to empirically explore the development group activity intensity model measured by change in lines of code. The results of these regressions are shown in Table 10.

All four regressions satisfy the assumptions of the linear regression model. After transformation, the skewness value was less than the recommended level of 1 for the dependent variable, change in lines of code. As in the regression that explores development group activity as the percentage of tasks closed, I centered all variables to reduce the possibility of multi-collinearity resulting from the inclusion of interaction terms for testing the moderation hypotheses. Because the independent variables are the same as when exploring development group activity as the percentage of tasks closed, there is not a problem with multi-collinearity, serial correlation and the distribution of errors.

The first regression in Table 10, labeled Main Effects, is significant and explains 41% of the variance according the R squared (p<.01). The results of the regression show that, as hypothesized, there is a quadratic relationship between development group platform tenure diversity and the changes in lines of code (Support for H5). Also, development group relationship density has a positive association with changes in lines of code (Support for H7). IBUC relationship density has a negative relationship with change in lines of code (Significant – Wrong Direction H6). IBUC platform tenure diversity, IBUC dynamism and the knowledge transfer variables, IBUC-development group communication and application type, do not have a relationship with changes in lines of code that is significant. However, the number of project members has a positive and significant relationship with changes in lines of code.

The second regression in Table 10, labeled Moderated Effects IBUC-Development Group Communication includes interactions between the IBUC knowledge acquisition variables and the measure of knowledge transfer, IBUC-development group communication. According to the R squared this regression is significant and explains 58% of the variance in changes in lines of code (p<.01). This model is significantly different from the main effects model and suggests that the association of both IBUC relationship density (Significant – Wrong Direction for H11) and IBUC dynamism (Support for H12) are altered by IBUC-development group communication. There was not evidence that the relationship between the IBUC platform tenure diversity and changes in lines of code was changed by the IBUC-development group communication.

A graphical depiction of both of these interactions is developed to further investigate the interactions. Figure 11 depicts the interaction between IBUC relationship density and IBUC-development group communication. This graphic shows that for projects with both high and low IBUC-development group communication higher IBUC relationship density leads to lower changes in lines of code. When comparing high and low IBUC development group projects it is clear that each additional unit of IBUC relationship density is associated with a larger decrease in change in lines of code for projects with low IBUC-development group communication compared to projects with high IBUC-development group communication.

Figure 12 depicts how IBUC-development group communication alters the relationship between IBUC dynamism and changes in lines of code. This graphic shows that there is a positive relationship between IBUC dynamism and changes in lines of code for projects with high and low IBUC-development group communication. But, a single increase in IBUC dynamism increases changes in lines of code for projects with high IBUC-development group communication more than for projects with low IBUC-development group communication.

The third regression in Table 10 includes interactions between the IBUC knowledge acquisition variables and the measure of knowledge transfer, application type. The third regression is significant and explains 45% or the variance in changes in lines of code (p<.01). However, it does not suggest that application type alters any of the relationships between IBUC knowledge acquisition capability and changes in lines of code.

The fourth and final regression in Table 10 includes all main and interaction effects and explains 64% of the variance in changes in lines of code according to the R squared (p<.01). The change in R between this model and Model B and Model C is significant, suggesting Model D is a better fit than Model B or C (p<.05). Model D is used to test all hypotheses. Many of the relationships are consistent with what has been shown in the previous models, but there is one exception. The interaction between application type and IBUC platform tenure diversity is significant, and a graphical depiction of this relationship is developed. As above the main effects are not interpreted.

Figure 13 displays the interaction between application type and IBUC platform tenure diversity. It shows that for applications designed for use by developers there is a positive relationship between IBUC platform tenure diversity and changes in lines of code. However, for applications designed for use by end-users IBUC platform tenure diversity has a negative effect on changes in lines of code.

5.3.3 Summary

By using change in lines of code as the measure of development group activity this section sheds light on the effects of absorptive capacity in OSS project performance. The second column in Table 11 documents those hypotheses that are supported in the analysis of change in lines of code as the measure of development group activity. This analysis, in conjunction with the analysis using percentage of tasks closed, enables a stronger understanding of the complex relationships between OSS project absorptive capacity and these two measures of development group activity. Chapter 6 continues by considering the difference between the results for the model with change in lines of code as the measure for development group activity compared to the results when the percentage of tasks closed is the measure of development group activity. The difference between the two measures is used to frame the interpretation of the findings and provide final conclusions.

CHAPTER 6: DISCUSSION

In this chapter, I conclude with a discussion focused on interpreting the results of the empirical study using the theoretical frame developed in Chapter 3 and 4 as the lens. I then also discuss the limitations of the dissertation. Finally, the key implications for theory and practice, together with promising future research directions are highlighted.

6.1 Discussion Of Empirical Results

Despite the ubiquity of OSS development and the many kinds of roles people play in OSS projects, there is a lack of theory to understand the characteristics and behaviors of OSS project participants that lead to performance for OSS projects. The goal of this dissertation is to use a knowledge lens to investigate the characteristics and behaviors of participants that lead to OSS project performance. In doing so this dissertation seeks to extend the current understanding of absorptive capacity by exploring the effect of absorptive capacity on OSS projects and how OSS projects develop it. To achieve this goal, this dissertation develops a conceptual model that focuses on two key sub-constructs of absorptive capacity; knowledge acquisition and knowledge transfer. In addition to the development of the conceptual model, two research models are developed and each is empirically tested using two separate measures of development group activity. This section attempts to understand the empirical study using the theoretical models in Chapter 3 and 4.

The overall empirical results provide support for the underlying thesis of the dissertation that absorptive capacity is important in determining the level of development group activity in an OSS project. Some results related to how absorptive capacity affects OSS projects are unexpected. For instance, relationship density and knowledge transfer do not affect the percentage of tasks closed in the way the theory in Chapter 4 suggests. When the empirical results are not consistent with the theory in Chapter 3 and 4 alternative explanations are explored.

In order to develop alternative explanations for the inconsistent results several steps were taken. The patterns of the unexpected results were considered to aid in the development of alternative explanations. Further, additional data was analyzed for each project to facilitate a rich understanding of the OSS projects and absorptive capacity. All threads related to the sample of projects were downloaded and read. They were explored to decipher themes that enhance comprehension of the empirical results. Specifically, each thread was summarized and then threads that had similar summaries were grouped into themes. A similar process was undertaken to analyze notes related to tasks opened for each project. Illustrative examples of the themes are included in the discussion to foster a clear picture of the explanations.

I continue with a discussion that is organized as follows. First I interpret the empirical examination of the survival model in Chapter 3. The empirical analysis of the intensity model in Chapter 4 is presented next.

6.1.1 Interpretation Of The Survival Model

The theory in Chapter 3 suggests characteristics of the development group and IBUC representing a knowledge base that enables exploration are expected to lead to survival. I argue that after the initial release of the project, exploration facilitated by fresh and diverse knowledge acquisition lead to survival for OSS projects.

As expected, the empirical results show that fresh knowledge is important for exploration that leads to survival. IBUC dynamism has a positive effect on survival in terms of tasks closed and lines of code added (H3 Table 5 and 9). Development group dynamism is not observed because as discussed in Section 3.1, an empirical study by Madey et al. (2004) provides evidence that the members of the development group frequently do not change. This limits the fresh software development knowledge that could be acquired through a new developer. However, when the project is developing an application for use by developers, the IBUC includes developers. In this case the IBUC can bring fresh software development knowledge about use of the application *and* software development. This provides an explanation for why the empirical results reveal that application type has a positive effect on survival (Table 5 and 9).

The empirical results provide a less consistent story concerning the effect of diverse knowledge on survival. The empirical analysis indicates that development group platform tenure diversity is not related to OSS project survival (H2 Table 5 and 9). The effect of team diversity on performance can diminish over time (Pelled et al. 1999). Accordingly, one reason the development group platform tenure diversity does not affect survival could be that since the development group does not change membership, the diversity impacts the project soon after initial registration and does not affect outcomes in later periods. IBUC platform tenure diversity only affects survival when survival is measured by tasks closed (H1 Table 5 and Table 9). This can be explained by the fact that when there are old and new IBUC members in the community the long tenure IBUC members, who are immersed in the SourceForge development process, encourage the short tenure IBUC members to submit tasks. However, overtime open tasks are more likely not to require code compared to tasks opened soon after registration; for instance, in later periods the tasks may focus on requests for documentation or help with usage. In addition, tasks opened in later periods may not be interesting enough for developers to want to add code. Developers often abandon projects or lose interest in them overtime as indicated by the fact that so few projects survive, in terms of new releases, after one year (Stewart et al. 2006c). So, after significant time passes since the project registers, the developers may suggest some other mechanism to close the task. For example, if the task is about a perceived bug in the application, a developer may close the task by stating that it is not a bug, but simply the way the application functions.

In summary, the empirical analysis of the survival model reiterates the importance of fresh knowledge for innovation and establishes the importance of fresh knowledge for OSS projects. The effect of development group diversity may diminish over time and IBUC platform tenure diversity seems to have an effect because of more senior IBUC members guiding newer members to open tasks.

6.1.2 Interpretation Of The Intensity Model

A second model is developed in Chapter 4 to understand the antecedents of development group activity intensity. Two models are needed because the expectation is that intensity and survival have different antecedents. While the survival model focuses on knowledge acquisition that enables exploration, the intensity model in Chapter 4 focuses on knowledge acquisition that enables exploration and exploitation. It also includes knowledge transfer that facilitates exploitation. Like for the model in Chapter 3, the empirical analysis for the intensity model examines both change in lines of code and percentage of tasks closed as the outcome measure. The results offer evidence that supports the intensity model. First, I discuss the variables that represent exploration knowledge acquisition and then the variables that represent exploitation knowledge acquisition. Then I discuss knowledge transfer and conclude with a dialogue about unexpected results.

Two out of three variables indicating knowledge acquisition that enable exploration are associated with development group activity regardless of the outcome measure. The first exploration knowledge acquisition indicator, IBUC dynamism, positively influences both measures of development group activity intensity (H8 Table 6 Model A and Table 10 Model A).

But, the other variables representing knowledge acquisition exploration associated with development group activity differ based on the outcome measure. IBUC platform tenure diversity affects the percentage of tasks closed (H4 Table 6 Model A), while development group platform tenure diversity affects change in lines of code (H5 Table 10 Model D). This pattern of results imply user knowledge is more important for the percentage of tasks closed, since two variables affecting this outcome are related to the IBUC, and software development knowledge is more important for the change in lines of code.

The effect of exploitation knowledge acquisition on development group activity also differs based on the outcome measure. Knowledge acquisition that facilitates exploitation is represented by relationship density. The empirical analysis finds no evidence that relationship density has a main effect relationship with the percentage of tasks closed (H6 Table 6 Model A and H7 Table 6 Model D). In contrast, relationship density has a main effect relationships with change in lines of code (H6 Table 10 Model A and H7 Table 10 Model D). While source code from all OSS projects is available, a project is not possibly aware of the source code in all other OSS projects. Relationship density can indicate the level of awareness a project has about source code that is useful and available. This is because when a project is related to another OSS project, project participants have knowledge about access to code that enables a certain function because they use the other OSS application or develop it. Therefore, relationship density represents software development knowledge. These results provide further evidence of the importance of software development knowledge for change in lines of code and also suggest that the percentage of tasks closed is only affected by exploration facilitated by IBUC knowledge acquisition.

The strong impact of user knowledge is represented by 2 out of 3 IBUC characteristics having main effect relationships with the percentage of tasks closed. This can be explained by the direct influence the IBUC has on this measure by opening tasks. The IBUC members open 63.4% (1,404 /2,213) of the open tasks associated with projects in the sample. Further, the only IBUC characteristic that does not have a significant relationship with the percentage of tasks closed can be considered a source of software development knowledge. Although IBUC relationship density could indicate a source of user knowledge, as discussed above it also represents a channel though which software development knowledge can enter the project.

The critical influence of software development knowledge on change in lines of code is also evident. The empirical analysis shows that all of the variables indicating software development knowledge, including IBUC relationship density, have main effect relationships with the change in lines of code (Table 10 model D). This can be explained by the fact that each change in the number of lines of code requires some software development knowledge. Conversely, increases in the percentage of tasks closed do not always require software development knowledge. Sometimes a task is opened in error by a user. A note attached to a bug submitted by an IBUC member for the project Strustsbox exemplifies this phenomenon. It mentions "Please close the bug – user error ☺. " Another type of task that does not require software development knowledge is one that requests information about how to find or use the code. Consider the detail associated with an open task.

IBUC Member: *Where is It? From reading the posts in this forum, I gather that there are some people out there using CHADDB. However, I can't seem to find a place to download it. Could you send me a download link?*

The interaction of IBUC knowledge acquisition and IBUC-development group knowledge transfer also represents exploitation. In Chapter 4 I argue that the knowledge transfer capability intensifies the effect of IBUC knowledge acquisition on development group activity intensity. But, in general the empirical results suggest that high IBUC knowledge acquisition is related to a lower percentage of tasks closed for high knowledge transfer projects compared to low knowledge transfer projects (Figure 6, 7 and 8). In contrast, high IBUC knowledge acquisition is related to higher change in lines of code for high knowledge transfer projects compared to low knowledge transfer projects (Figure 12 and 13). Figure 11 reveals a similar pattern although the depiction of the interaction is not as dramatic. Together the empirical results point toward the fact that these interactions have different effects on development group activity according to the outcome measure. This discussion continues by presenting explanations for this difference.

Percentage of tasks closed includes a penalty for tasks that are not addressed. For example, consider the situation where project A and B each close 2 tasks, and project A had 2 open and project B had 10 open. Project A has percentage of tasks closed equal to 100% and project B has percentage of tasks closed equal to 20%. The difference between the percentage of tasks closed measure for project A and B represents the penalty project B faces because of tasks that are not addressed.

Because of the penalty the percentage of tasks closed is sensitive to too much knowledge acquisition and transfer, but change in lines of code is not as sensitive. In essence when a project has high knowledge acquisition and high knowledge transfer there are likely too many tasks open that are not interesting to the developers. For instance, if there is high IBUC dynamism there are likely many new users who submit inappropriate or uninteresting tasks. Consider the examples above where an IBUC member requested a bug be fixed that was not a bug, but user error. If there is also strong knowledge transfer, the new IBUC members may believe that the developers are interested in their opinions because they respond to their questions. Because they feel their opinion is valued they are likely to open an increased number of tasks. Also, there can be too much knowledge transfer that does not lead to task closure. In the case of high IBUC dynamism the questions in the threads could focus on questions to help new users begin using the application. This IBUC development group communication takes developer time away from task closure. An example of this kind of thread is shown below.

IBUC Member: *"I tried to run the Mugen.NET.exe file, but had an abnormal progran termination error, and nothing else. Is this just a bug in the program, or some configuration error? [If it's the config, what should I do to fix it?] Thanks and good luck with this."*

The only time knowledge transfer does not limit the effect of IBUC knowledge acquisition on the percentage of tasks closed is when the IBUC knowledge acquisition is represented by IBUC relationship density and knowledge transfer is measured by application type. In this case an IBUC member is expected to have experience as a developer and also experience using OSS applications. So, she is able to submit tasks that are interesting to the development group and she does not have many basic user questions. In essence there is consensus between the IBUC ideas for tasks to be completed and developer interest in completing them.

The unexpected positive main effect relationship between knowledge transfer and percentage of tasks closed further emphasizes the importance of consensus between the development group and IBUC for increasing the percentage of tasks closed (Table 6 Model A). The theory in Chapter 4 focuses on knowledge transfer as a way to intensify the effect of IBUC knowledge acquisition, but it can also alter the degree to which the IBUC members understand the development group. For instance, the development group can manage the development process through IBUC-development group communication. They can lead an IBUC member to open tasks that are likely to be closed. This can explain knowledge transfer having a positive relationship with the percentage of tasks closed. Consider the below illustration of this kind of IBUC-development group communication.

IBUC Member: *Currently it appears you can only chart 6 months, 1 year, 2 years, and all data.. .is it possible to change it to a user defined period? Also can the candle charts be daily, weekly etc? Great looking system overall !!!!*

Developer: *"A user-defined period for the charts is not currently possible, nor a different period for the candle charts. I can add them in a future release, please open a feature request …so I don't forget it."*

A final set of unexpected results relates to the direction of relationships. The original hypothesis suggests IBUC relationship density has a positive effect on development group activity, but the empirical analysis indicates IBUC relationship density has a negative effect on the change in lines of code (H6 Table 10 Model A). The original theoretical development does not consider the fact that the development group could be induced to develop on a linked project. An examination of the threads uncovers that sometimes IBUC members are show interest in attracting developers to other OSS projects. When developers join linked projects, the change in lines of code on the original project is apt to decrease because the developers divide their time between more projects. An example of an IBUC member attempting to recruit developers from the jcrawler project is shown below.

IBUC Member: *"I like your code, it's clean and simple. Please consider joining the Apache JMeter team and lending your help there. We're a small group and no one works on JMeter as their job, so we could really use help, and anyone who takes an active role is going to be able to get what they want done without too much interference from the developers who've been there."*

A second result that is contrary to expectation based on the direction of the relationship is the relationship between IBUC platform tenure diversity and percentage of tasks closed (H4 Table 6 Model A). The theory in Chapter 4 suggests IBUC platform tenure diversity decreases intensity because the diverse IBUC offers divergent ideas that make it difficult for developers to agree on a direction for development. However, a review of the tasks shows that many tasks that the IBUC members submit do not change the direction of the project; in contrast, they tend to be incremental adjustments. For example, bugs are an example of incremental adjustments that usually do not change the direction of the project. 66% of the tasks that the IBUC opens are bug reports. In this case a diverse IBUC can lead to an IBUC using the application in many ways that produce multiple kinds of bugs providing a variety of opportunities for the development group to sharpen their skills. The variety of open tasks is important so that developers at different levels of expertise have challenging yet achievable opportunities to close tasks.

6.1.3 Summary Of Discussion

The interpretation of the empirical results provides a nuanced story of the relationship between OSS project absorptive capacity and OSS development group activity. Survival for an OSS project whether based on change in lines of code or tasks closed have similar antecedents. IBUC dynamism and application type affect survival in terms of lines of code and tasks closed which indicates the importance of fresh knowledge for an OSS project. When considering the intensity model, the results suggest that the degree to which a certain type of knowledge acquisition matters and the effect of knowledge transfer depend on whether the outcome is a change in the number of lines of code or the percentage of tasks closed. Software development knowledge is more important for change in lines of code. The penalty for not responding to open tasks causes knowledge transfer to affect the percentage of tasks closed and change in lines of code differently. Knowledge transfer tends to lead to a more positive relationship between IBUC knowledge acquisition and change in lines of code and it leads to a less positive relationship between IBUC knowledge acquisition and the percentage of tasks closed.

6.2 Limitations

Some limitations of this dissertation that suggest fruitful extensions to this research are notable. Two limitations focus on the ability to generalize from the sample chosen and the measurement of the variables. A final limitation relates to further understanding the relationships that are not supported. Each limitation is discussed below.

Although SourceForge is one of the largest open source development platforms, there are other OSS development platforms. Some projects do not use an OSS development platform that other OSS projects use. The largest OSS projects, such as Linux and Apache, do not use SourceForge. OSS projects that use other platforms are likely to have different characteristics from those OSS projects that use SourceForge. For instance, OSS projects with corporate funding may use development platforms that require financial resources, while OSS projects with limited resources will use the free platform, SourceForge. If a project has financial sponsors, those sponsors have opinions concerning development that outweigh and are not necessarily in concert with IBUC opinions. For example, Google leverages OSS development processes, but this project may not be as affected by an IBUC as projects on SourceForge. If a project does not use a platform that connects many projects, the developers on that project are likely to have fewer relationships with other projects. For these reasons it is not clear how well the model in this research applies to those projects that do not use the SourceForge development platform or projects that have corporate sponsors. Future research should explore this model in contexts beyond SourceForge.

Archival data operationalize the variables in this research, and so the data may not exactly represent the theoretical constructs. Surveys allow for a stronger understanding of the theoretical constructs and their relationships. However, this analysis is done at the level of the OSS project, and it is difficult to acquire surveys from multiple participants in each OSS project for a large sample of projects, and single source respondents have challenges related to single response biases. A case study can shed additional light on the constructs and relationships this research proposes.

Finally, the empirical analysis reveals several results that are contrary to expectations. While I provide explanations for these results it is outside the scope of this work to test these arguments. Future research is needed to explore the role of additional variables.

6.3 Implications For Theory And Practice

This dissertation advances the research on absorptive capacity by highlighting sub-constructs that are important in an Internet based context marked by a dynamic environment. I present the knowledge acquisition and knowledge transfer constructs as important for organizations seeking to innovate in a dynamic context. Further, this dissertation demonstrates that these sub-constructs affect performance directly and interactively. Also, it shows how knowledge transfer can enhance or limit knowledge acquisition depending on the outcome of interest.

In addition, this research advances the line of research that seeks to understand the factors that lead to improved OSS project performance in terms of development group activity. Prior OSS research focuses almost exclusively on the development group when identifying factors that lead to development group activity (e.g. Roberts, et al. 2006; Stewart et al. 2006). This research identifies the IBUC as a key set of participants that increases development group activity. In particular, it identifies characteristics of the IBUC and development group and also interaction patterns between the two that improve development group activity. While using an IBUC in other development contexts is not specifically addressed in this research, this dissertation suggests that other development processes can benefit from an IBUC.

Lastly, this dissertation has implications for practice. As OSS development begins to blend with proprietary software development, managers need to know how the open source software development works (Fitzgerald 2006). In particular, when managers consider paying open source developers, they should be aware which developers are the most valuable. I am able to inform a manager, based on the outcome that is of interest to him, of the types of participants that he should target for participation and the behaviors he should encourage in an OSS project. For instance, a manager could be interested in adding a lot of code because it is a new project and many code intensive features need to be added. He should understand that the participants who affect this outcome are those that have access to software development knowledge. Access to development knowledge can obviously be attained through the development group and not so obviously though the IBUC members with many relationships to other OSS projects. He should also encourage the development of a strong knowledge transfer capability because it improves the positive effects of IBUC knowledge acquisition.

If I consider a manager interested in closing tasks because her concern is in ensuring users of the application feel their concerns are addressed, she should focus on attracting an IBUC that is dynamic and diverse. Specifically, this manager should continually recruit new IBUC members and also seek to maintain the long-term IBUC members. This manager should manage knowledge transfer capabilities and perhaps institute knowledge management procedures to minimize the limitations that knowledge transfer can have on the positive benefits of IBUC knowledge acquisition. Finally, a manger interested in survival should focus on creating a dynamic IBUC by constantly recruiting new users to join the community and if funding allows, the manager should try to recruit new developers.

6.4 Future Research Directions

OSS projects are on the forefront of leveraging the Internet for innovation, however there are other examples of Internet use in innovative processes. IBUCs have emerged around a variety of products beyond OSS, but the research to understand how an IBUC affects product development is limited. For instance, although there are communities associated with Microsoft (http://forums.microsoft.com) it is yet to be uncovered how an IBUC affects proprietary software development. The effects of an IBUC in a proprietary setting can be different from the open source setting given the hierarchy of control and the limited access to source code in most proprietary software development. Even though there is a rich literature on how users affect proprietary software development, an IBUC, as discussed above, is different from other user groups, and the effect of an IBUC still needs to be further identified in the proprietary setting.

In addition to product development, internet based communities affect other outcomes, such as sales and health management. Communities focused on media, including movies or books, are likely to have effects on sales based on discussion about the media. There is evidence that internet based community ratings of movies alter movie sales (Dellarocas 2003). This work can be extended by examining how the characteristics of the internet based community offering the recommendations alter sales. Similarly, there is evidence that the design of internet communities can enable knowledge exchange for those seeking to improve their health (Ma et al. 2007).

Exploration of the characteristics of internet support communities that improve the user’s experience can enable managers of such communities to improve user satisfaction and increase revenues from site development.

Table 1: OSS Development Literature Review

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects (Roberts et al. 2006)	Individual	Intrinsic and Extrinsic Motivation, Education, Use, Experience, Status and Participation (number of source code contributions accepted)	Rank within Apache	Survey	They find developer motivations are interdependent and that different motivations have alternative effects on different outcomes. Finally they find past success is an indicator of attracting developer interest.
An Empirical Analysis of Economic Returns to Open Source Participation (Hann, Roberts, Slaughter and Fielding 2005)	Individual	Volume of Software Code Submissions and Status in Apache	Increases in Wages	Survey and Archival Data	They use labor economics to understand why developers participate in OSS.
Motivation, Governance and the Viability of Hybrid Forms of Open Source Software Development (Shah 2006)	Individual	Governance	Developer Participation	Qualitative Interviews	A framework is derived to understand motivations to participate with a focus on different governance structures.

OSS Development Literature Review					
Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
Motivation of Software Developers in OS Projects: An Internet-based survey of contributors to the Linux kernel (Hertel et al. 2003)	Individual	Identification as a Linux Developer, Pragmatic Motives, Tolerance of Time Investment	Participant Engagement, Willingness to be Involved in Subsystem in the Future, Lines of Source Code and Number of Patches Accepted, Satisfaction	Survey	They draw on volunteer, collective action, small teams and OSS anecdotes to identify motivations for participation in OSS development. Only self efficacy lead to more SLOC for a developer.
Equilibrium Selection and Public Good Provision: The Development of Open Source Software (Myatt et al. 2002)	Individual	Degree of Relatedness of Projects, Optimism versus Pessimism,	Participation in OSS Development	Analytical Modeling	Less able programmers may free ride on more capable programmers. Asymmetries between programmers might be explained to generate integrated public good provision. Optimism or pessimism may influence success of integrated or separated projects.

OSS Development Literature Review

Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
Open Source Software User Communities: A Study of Participation in Linux User Groups (Bagozzi et al. 2006)	Individual	Individual attitudes and emotions and perceptions about control and LUG social identity(cognitive, affective and evaluative social identity) -used experience as moderator	User Group Participation	Survey	They use Theory of Planned Behavior and model of goal directed behavior to understand the factors that lead to user group participation.
The Architecture of Participation: Does Code Architecture Mitigate Free Riding in Open Source Development Model (Baldwin et al. 2006)	Individual	Code Modularity and Option Value	Developer's incentive to join and remain involved in OSS development effort and decrease the amount of free riding in equilibrium	Analytical Modeling	They base analysis on private provision of public goods.
Working for Free? Motivations for Participating in Open Source Projects (Hars et al. 2002)	Individual	Age, Sex, Motivations, Projects Worked on, Paid by Commercial Sponsor, College Degree, Professional Programming, Student		Survey	External rewards seem to be more salient than internal rewards.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Why Developers Participate in OSS Projects: An Empirical Investigation (Hann et al. 2004)	Individual	Exploratory Analysis		Survey	Establish 5 functional motivations.
The Impact of Ideology on Effectiveness in OSS Development Teams (Stewart et al. 2006b; Stewart et al. 2006d)	Team	OSS Team Adherence to OSS Ideology Quality of Team Communication and Trust in the OSS Team	Team Size, Team Effort, Task completion and Development stage	Survey	They find the norms and values of the larger OSS community must be fostered and consistent task communication maintained for projects to succeed.
Effective work practices for Software Engineering: Floss Development: A Model and Hypotheses (Crowston et al. 2005)	Team	Organizational context not explored (reward, education and information systems) and group design (task structure (coordination and collective mind), team composition and team norms)	Team Effectiveness (taskoutput - system quality, individual satisfaction, continued capability to work together) measured by level of activity, number of downloads, development status, user ratings	None completed	They use team effectiveness theory (Hackman), collective mind and coordination.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
The social structure of Free and OSS (Crowston and Howison 2004)	Team	Conceptual		Social Network Analysis	Large teams are less centralized.
Impacts of License Choice and Organizational Sponsorship on Success in OSS Development Projects (Stewart et al. 2006b)	Project	License Choice and Organizational Sponsorship	Developer Activity and User Interest	Survey	Users are most attracted to projects that are sponsored by non-market organizations and employ non-restrictive licenses. The influence of licensing on development activity depends on what kind of organizational sponsor a project has.

OSS Development Literature Review					
Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
An Exploratory Study of Factors Influencing the Level of Vitality and Popularity in OS Projects (Stewart et al. 2002)	Project	Organizational Sponsorship, Target Audience (developer vs. end user), License Choice and Development Status	User Attention and Developer Activity	Survey	Vitality has a significant impact on popularity over time. Otherwise stated, the more active a project is in terms of making releases and announcements, the greater the attention from the community. They found sponsored projects to have a better success rate than non-sponsored projects.
The Moderating Role of Development Stage in Affecting Free/Open Source Software Project Performance (Stewart et al. 2006a)	Project	Development Stage, Trust and Shared Ideology, [control Administrator Experience and Project Age]	Number of Developers, Task Completion and Perceived Effectiveness	Survey	They find the relationship between team climate variables and objective and subjective outcomes vary based on the stage of development.
From Planning to Mature: on the Determinants of Open Source Take Off (Comino, Manenti and Parisi 2005)	Project	License Choice, Target Audience is Administrator	Development Stage	Archival Data	They find projects that develop software targeted to more technical users are more successful.

OSS Development Literature Review					
Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems (Grewal et al. 2006)	Project	Network Embeddedness of Projects and Project Leaders	CVS Commits and Number of Commercial Downloads	Archival Data	There are different factors that lead to technical success compared to commercial success.
The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination (Fershtman et al. 2004)	Project	License Choice, Programming Language, Audience [control for OS, Multiple or Single OS, Age of Project, Application Type, Programming Language]	Output Per Contributor (sloc/contributor)	Survey	Projects that use a non restrictive license have higher mean SLOC, number of contributors and sloc/contributor. Projects that develop software for developers use less restrictive licenses.
The Influence of Networking Governance Factors on Success in OSS Development Projects (Sagers 2004)	Project	Restricted Access to Team Membership, Collective Sanctions and Reputation	Age of Project, Ratio of Total Bug Reports and Feature Requests to Open Bug Reports and Feature Requests, Perceptions about Value, Performance and Utility	Survey	He finds restricted access to the development team improves coordination within OSS projects and safeguards exchanges among project members. He also finds the importance of reputation minimizes conflicts in OSS projects.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Membership Dynamics and Network Stability in the Open-Source Community: The Ising Perspective (Oh et al. 2004)	Project	External Forces (IV) and Network Size(mediator) and Connectivity(mediator) Scale Free versus Random	Network Stability and Herding Behavior	Simulation	They use the Ising Perspective and Soft Systems Method to develop a model to understand network stability in OSS projects.
Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development (Crowston et al. 2002)	Project	Availability of Competencies (popularity of programming language, intended audience, developers) Competency Marshalling (number of administrators and developers and rank of administrator) higher activity[used project lifespan as control]	Number of Downloads, Page Views, Development Status and Intensity of Work by Developers	Archival Data	They find choosing a popular programming language and having a popular administrator can help a project be more successful.
Survival of Open-Source Projects: A Population Ecology Perspective (Chengalur-Smith et al. 2003)	Project	Organizational Age, Number of Core Developers, Reliability and Niche Strategy	Project Survival, Reliability Between Releases	(research in progress -no data)	This paper highlights key factors that should lead to OSS project survival by using an ecology perspective.

OSS Development Literature Review

Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
The Scope of Open Source Licensing (Lerner et al. 2002a)	Project	Number of Developers	Activity	SourceForge	Commercial operating systems and projects for developers use less restrictive licenses while projects for end users use more restrictive licenses.
From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development (Lee et al. 2003)	Project	Criticism and Critical Evaluation	Knowledge Creation	Case Study	They compare Linux to traditional forms of software development to identify a new knowledge creation model.
A case study of OSS development: The Apache Server (Mockus et al. 2000)	Project	Defect Density (defect per thousand lines of code; defect per thousand deltas) and Time to Repair Errors (depending on severity of error), number of developers and participants (bug reporters), number of people who add features, number of people who correct errors, number of features and bugs, number of developers who do most of developing, specialization (number of people who work on a single module)		Case Study	Apache developers have similar productivity levels to commercial developers. For example 15 developers do 80% of work.
Two Case Studies of OSS Development: Apache and Modzilla (Mockus et al. 2002)	Project	Same as Mockus et al. 2000 except specialization		Case Study	They study Modzilla and Apache for 3 years.

OSS Development Literature Review					
Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
The FreeBSD Project: A Replication Case Study of OSS Development (Dinh-Trong et al. 2005)	Project	Same as Mockus et al. 2000		Case Study	Focused on further supporting or contradicting Mockus et al.’s hypotheses. They find a larger number of developers did most of the work for FreeBSD compared to Apache.
Community joining and specialization in OSS innovation: a case study (vonKrogh et al. 2003)	Project	Joining Script that includes actions – e.g. Number of Emails Sent Before Users Become Developers, Time Between First Email and Being Given CVS access		Inductive Case Study	From their study of Freenet they find developers and non developers equally likely to be responded to and new releases brought new participants
Evolution in OSS: A Case Study (Godfrey et al. 2000)	Project	Lines of Code, Functions, Files, Variables		Case Study	The growth pattern contradicts Lehman's Laws.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
How open source software works: "Free" user-to-user assistance (Lakhani et al. 2003)	Project	Provision of field support -compared info providers and seekers and frequent providers to non frequent providers and the same for frequent seekers and non frequent seekers, considered time between question and answer, consider time for providers to prepare answer		Case Study	People do mundane tasks in OSS because they learn from doing them.
Essence of Distributed Work: The Case of the Linux Kernel (Moon et al. 2000)	Project	Modularity, Leadership, Incentives for Sharing, Parallel Release, Testing and Development Stream, Communities of Practice		Case Study	They focus only on code writers and take three perspectives. Then they present implications for virtual organizations.
The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary (Raymond 2000)	Project	NA		They Describe the development of the Fetchmail project.	

OSS Development Literature Review

Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
The Ecology of Open Source Software Development (Healy and Schussman unpub)http://www.kieranhealy.org/files/drafts/oss-activity.pdf	Project	Developers, Downloads, Site Views, Message Unique Author, CVS Commits		Archival Data	All variables are highly skewed and characterized by power law distributions. Most downloaded are applications designed for end users. Most cvs commits are system level applications. Downloads and site views are not correlated with developer activity. They suggest professional core developers, project leadership and hierarchy are key factors in OSS success.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Free/Libre Open Source Software: Survey and Study (Ghosh 2002)	Individual	Sex, Age, Education Level, Profession, Marital Status, Number of Children, Income, National Affiliation, Time Spent Developing OSS, Time Spent Developing Proprietary, Preferred Operating System, Number of OSS projects Participated On, Amount of Contact with other Developers, Perceptions of Purpose of OSS Development, Identification with Free or Open Community		Survey	They provide insight into the fundamental features of OSS. They provide information about 2,784 developers and also distinguish between Free and OSS. They find developers are mostly male and well educated.
What makes a Virtual Organization Work? (Markus et al. 2000)	Virtual OSS Organization	Motivations (reputation etc.), Self Governance (leadership, managed membership, monitors and sanctions, rules, license and voting procedures), Institution Reputation, Technology, Work Structures and Processes, Culture (social and self control)	Functioning set of virtual knowledge workers	Descriptive	They focus on understanding how to acquire, keep and motivate knowledge workers (especially through means other than money - suggest leading instead of managing)

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Open Source Software: Free Provision of Complex Public Goods (Bessen 2001)	OSS Community	Application Type	Firm Participation in OSS development	Analytical Modeling	Complexity makes contracting difficult and provides a reason for firms to contribute to OSS.
Cave or community? An empirical examination of 100 mature open source projects (Krishnamurthy 2002)	Project	NA		Archival Data	They find most projects are developed by one developer with little discussion and more developers lead to more downloads.
Defining OSS Success (Crowston et al. 2003)	Project	System and Information Quality, Use, User Satisfaction, Individual and Organizational Impact, Number of Developers, Level of Activity, Cycle Time, Employment Opportunities, Individual Reputations, Knowledge Creation		Literature Survey	They expand on Delone and Mclean 2003 to identify success measures for OSS projects.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science (Von Hippel et al. 2003)	Project	Conceptual	Knowledge Creation	Theoretical	They suggest motivations for participation can be explained by private investment and public goods theory.
The Power of Gifts: Organizing Social Relationships in Open Source Communities (Bergquist et al. 2001)	OSS Community	NA		Virtual Ethnography	They discuss how power is related to gift giving and how societies based on scarcity use exchanges and abundance as signs of power.
Striking a Balance between Trust and Control in a Virtual Organization: A Content Analysis of Open Source Software Case Studies (Gallivan 2001)	OSS Community	Cooperation, Reliable Performance		Content Analysis of Case Studies	They suggest social and self controls can be uses as a substitute for trust.
Open Source Software Development and Distributed Innovation (Kogut et al. 2001)	OSS Community	Conceptual			They discuss effect of licensing, modularity and geographical distribution on innovation.

OSS Development Literature Review

Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
OSS Development (Von Krogh 2003)	OSS Community	Conceptual			They give description of OSS and overviews when and why it occurs.
Innovation by User Communities: Learning from OSS (Hippel 2001)	User Community	Conceptual			He presents the idea of users acting as manufacturers in the development of products. He uses OSS as an example.
Open Source Software: A History (Bretthauer 2002)	OSS Community	Conceptual			They discuss GNU, Berkley Software Distribution and Linux.
Gift economies in the development of OSS: Anthropological reflections (Zeitlyn 2003)	OSS project	Conceptual			They suggest that there are better metaphors than Raymond's Cathedral and Bazaar -e.g. anthropological.

OSS Development Literature Review					
Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
Permanently Beta: Responsive Organizations in the Internet Era (Neff and Stark 2003)	Project process	User Involvement		Descriptive	They describe how the internet fosters innovation in a new way.
Free Software Needs Profit (Ousterhout 1999)	OSS Community	NA		Descriptive	OSS can be more successful when linked with a corporate sponsor.
Understanding the Requirements for Developing Open Source Software Systems (Scacchi 2002)	OSS project	NA		Case Study	They focus on requirements and how the collection and use of requirements for OSS may be different than traditional
A Framework Analysis of the OSS Development Paradigm (Feller et al. 2000)	OSS Phenomenon	NA		Descriptive of OSS	They develop a framework based on Zachman's IS architecture and Checkland's CATWOE framework from Soft Systems methodology to analyze the open source approach to development.

OSS Development Literature Review

Study	Level of Analysis	Independent Variables	Dependent Variables	Method	Study Highlights
The Boston Consulting Group/OSDN Hacker Survey (Lakhani, Wolf, Bates and DiBona 2002)	OSS Developer/OSS project	Motivations, Demographics and Developer Profession		Survey	They present a study of developer motivations, demographics and professions of developers. They find creativity lead to more activity per week.
Some Simple Economics of Open Source (Lerner et al. 2002b)	OSS project	Leadership, Developer Motivation, Organization, Problem Type		Case Study	They pose venture capital as possible outcome to OSS developer participation.
Coase's Penguin, or, Linux and the Nature of the Firm (Benkler 2002)	OSS Community	Modular, Small Modules, Low Cost Integration		Case Study	They suggest peer production has advantages over hierarchies and markets in matching human capital to tasks.
Who is an Open Source Developer (Dempsey et al. 2002)	Linux Contributors and Contributions	Diversity of Contributors, Number of Contributions Per Developer and License Type		Archival Data	Contributors come from around the world, and the rate of contributions is increasing.

OSS Development Literature Review					
Study	**Level of Analysis**	**Independent Variables**	**Dependent Variables**	**Method**	**Study Highlights**
OSS as Lead User’s Make or By Decision: A Study of Open and Closed Source Quality (Kuan 2002)	OSS project	OSS or Proprietary project	Bug Resolution Rates	Archival Data	They suggest OSS is of greater quality compared to proprietary software because developers are users.

Table 2: Project Descriptions

Project Name	Description of Application
Agentfactory	The AgentFactory Framework provides tools for the fabrication and deployment of intentional software agents.
Algobros3	AlgoBros3 is a platform independent MarioBros clone written in Java.
Amtu	The Amazon Merchant Transport Utility is a Java-based utility designed to make posting files and receiving reports from Amazon's Merchants@ Interface as simple as dragging and dropping files into a directory.
Annoty	Annoty is an annotation web service aimed at any educational environment.
Antbuildcreator	Build Creator is a plugin for Eclipse that adds a wizard for creating Ant build files.
Argaut-xdbaudit	Argaut XDBAudit 1.8.3 scans corporate networks for Oracle and Microsoft SQL Server databases, collecting usage statistics and license related inventory audit information, including infrastructural metadata relating to capacity and performance metrics.
Bbalc	BBalc provides chess-computer like functions such as analyzing the game and generating/proposing actions. The modular design allows users to extend existing analysis modules to fit their needs.
Bbweblog	BBWeblog is a simple, customizable Bulletin Board/Weblog web application.
Calendartag	The calendar tag library provides a simple and easy way to generate an interactive calendar in your Java enabled pages. The library provides quick implementation and fully customizable look and feel.
Chaddb	CHADDB,the CHurch ADministration Database, is a table-driven church management system that enables churches of any size or structure to manage members through relationships. Web-enabled or run as native application.
Churchinfo	ChurchInfo is a free church database network application. This membership and management database is designed specifically to help automate the operation of a church. It has features to track members, families, groups, donations, etc.
Conges	Conges is a vacation management software. It permits employees to set vacation period, to print customizable vacation paper and for the service supervisor to modify or delete the vacation period entered by employees.
Cookxml	CookXml is a powerful dynamic XML data binding tool for Java. It is designed to be easily extensible and yet simple to use, with advanced exception handling. CookSwing and CookSwt are tag libraries for CookXml to do XML->GUI (XUL) for Swing and SWT.
Css2xslfo	CSSToXSLFO is a conversion utility from CSS2 to XSL-FO, which can be converted to PDF, PostScript, etc. It has special support for XHTML.

Project Name	Description of Application
Cubnc	CuBnc is a multisite bnc for ftp servers. It allows you to have multiple slave-servers connected to one master. When you enter the master via normal ftp, you see the slaves as dirs. When you enter such a dir, you become logged in to the site.
Daffodilreplica	Daffodil Replicator is a powerful Open Source Java tool for data integration, data migration and data protection in real time. It allows bi-directional data replication and synchronization between homogeneous / heterogeneous databases.
Delicious-java	Delicious-java is a Java API for interacting with the del.icio.us social bookmarks service.
Donj7	It is a minimally multiplayer role-playing game that tries to reproduce an actual Pen&Paper game, through massive use of dialog between players and a game master.
Drinnovations	Drinnovations is an eclipse plugin for Tab Alerts. You will see "Tabs, Alert Me!" for files and "Tabs, Folder Files" for folder on right clicking. It can tell you each line you have tabs,remove, and refresh your files.
Epice	Web-based course management and communication tool for Instructors and Students to use outside traditional classroom and office-hours. Assignments, materials, notes, questions, submissions, and scores can be up or downloaded providing 24/7 availability.
Eremise	The purpose of the E-Remise system will be to facilitate the proper evaluation of large numbers of electronic assignments. Another purpose of E-Remise is the archiving of source code and other work done by students. It is useful for pedagogical reasons.
E-volve	An email client which focuses on sorting incoming emails for better email management using classification algorithms.
Fddtools	FDDTools is a multi-platform application supporting Feature Driven Development (FDD). The current version can be used to create/edit/display/print FDD-style progress tracking diagrams as described in the book "Java Modeling In Color With UML".
Fjep	Eclipse Plug-In for deploying a project into one "fat" executable jar file containing all referenced libraries. References are taken from the project settings, so no manual configuration is necessary.
Floranta	The Floranta libraries provide components for building Rich Client Wikis using AJAX or Java. Floranta clients can be embedded in any webpage (even plain html) and allow users to leave notes, images, cards, etc on the pages. E.g.: http://www.floranta.com
G4j	G4J is a set of API that communicate to GMail. Use this API to login, retrive/search/browse message, download attachment and do others action on GMail account! An email application (Gmailer for java) is included to demonstrate the usage of the API.
Gavamail	Gavamail is a POP3/IMAP server implemented in JAVA for reading your gmail e-mail with your favorite mail client. Currently only POP3 is implemented but I hope to implement IMAP too. Libraries provided by the g4j and libgmail projects are used.
Gmf	GMF aims to provide a general framework to manage contents going a step further of traditional CMS. GMF is oriented towards the management of an IT department. One of its main goals is the complete implementation of the ITIL specification.

Project Name	Description of Application
Heroarena	Hero Arena (HA) will be a turn based strategy game written as a Java Application for a Windows OS. Development will be open to developers of all ranges with hopes of improving skills and knowledge through working on a fun, laid-back project.
Hidrosig	HidroSIG is a GIS that supports raster and vectorial maps with modules oriented to the hydrological analysis, time series, remote sensing and more. It has been made 100% in java using VisAD for data visualization and using MySQL to store all data.
Hitman	Hitman is java based load testing tool for websites. It gives valuable information about the response time when the number of hits increase.
Ivtk	The InfoVis Toolkit is an Interactive Graphics Toolkit written in Java/Swing to ease the development of Information Visualization applications and components.
Jasterisk	A set of JNI classes providing direct access to Asterisk PBX functionality from Java. This is not a socket-level interface into the Asterisk manager app but a true Java<->Asterisk integration at the Thread level.
Javamatch	JavaMatch is an engine that can search inside a runtime Java data structures, and look for objects that best match the criteria that you specify. The extensive query mechanism allows for highly customizable tuning of your match queries.
Jcrawler	JCrawler is a perfect cralwing/load-testing tool which is cookie-enabled and follows human crawling pattern (hit/second).
Jicengine	Java Instance Configuration Engine (JICE) is an XML-based tool for constructing and configuring Java applications. The configuration of the application is described in an XML file, which is then processed by JICE, yielding the corresponding application.
Jmage	Jmage is a java based framework for dynamic image modification. It allows configurable image filters to be applied on popular binary formats such as JPG, GIF, PNG, and use the resulting images in your own application.
Jmathtools	Jmathtools, mathematical / scientific java stuff designed to be easily used and modified, not really designed for high performance calculation.
Jphotos	This project is a Java Online Photo Album which offers the maximum in flexibility, functionality, and ease of use. It aims to be an easily set up and easily maintained CGI photo album for users with large numbers of photos they want to display.
Jpo-eclipse	Eclipse plug-in for Matrix One Java Program Objects.
Jregexptester	JRegexpTester is a standalone Swing application that helps you test regular expressions with the Java standard API. The extracted data can be modified with formatters similar to those used by sprintf. Since release 0.3, it helps you manage CSV files too.
Junitrunner	Eclipse plugin to allow run/debug junit test method using context popup menu. Method to launch is defined by cursor position. If cursor between methods the whole junit class is launched. Till v1.2.0: Also allow to run/debug class with main() method.

Project Name	Description of Application
Kasai	Kasai is a 100% Java based authentication and authorization framework. It allows you to integrate into your application a granular, complete and manageable permission scheme.
Kontent	Kontent is a local content management system.
Lasersquad3d	Laser Squad 3D is a squad-based tactical pseudo-realtime strategy game. It can be played either single player or two players over a network. It includes different missions, weapons and equipment. Emphasis is on easy setup, quick start, and enjoyment.
Lavadora	A Plug-in for the Eclipse Platform which allows to easy access, create and deploy Web Services, and also to work with WSDL documents, browse and publish own Web Service information in UDDI registries.
Ldapeclipse	An Eclipse plugin which provides functionality for working with LDAP directories from within Eclipse. It is possible to view and edit data in multiple directories and view the directory schema. The plugin features context sensitive help.
Lpd	The Local Projects Database (LPD) is a web-based tool for managing records on development assistance activities financed and/or implemented by development partners. This java based application, allows inserting, maintaining and exchanging activity data.
Megatrack	MegaTrack is a Java application for hurricane and tropical storm tracking for Atlantic and eastern Pacific storms.
Micro-jabber	Jabber client libraries for light java devices such as mobile phones, it brings instant messaging capabilities to every java phone (with support to MIDP), but may be used for other applications! It uses a very tiny XML parser crafted for this project.
Minpo	MINPO is a minimal java web-app, free to use as a starting point for a larger, stand-alone web-app, providing a basic framework (using Struts), persistence mechanism (using Hibernate), and portal GUI providing simple user authentication and management.
Monsterjournal	The Monster Journal - An electronic journaling program to replace the paper notebook. The Monster Journal provides the user a centralized, easily accessible electronic notebook for storing thoughts, ideas, and answers to questions.
Montag	Montag is a Web Services based system for interacting with XML Databases that provide a Java implementation of the XML:DB API. So, it permits heterogeneous SOAP-enabled clients, written for different platforms and languages, to operate over XML Databases
Mscp	A server control panel like cpanel, but free for use and platform independent.
Mstor	Mstor is a JavaMail local store provider based on the mbox email storage format.
Mugen-net	Mugen.Java or JMugen is 2D fighting game engine like Mugen

Project Name	Description of Application
Mywl	An Eclipse 3.0 compatible plugin which allows BEA's WebLogic Server (6.1, 7.1 and 8.1) to be managed from within eclipse and thus easily debug deployed J2EE applications in BEA.
Nanodesigner	Nanodesigner is a software platform for research on molecular nanotechnology. It has a plug-in architecture and will include tools for molecular visualization & modelling, design of complex new molecules, molecular dynamics,…
Na-worksheet	The NA_WorkSheet is a set of algorithms coded in Java that implements various Numerical Analysis techniques. The tool may be used for graphing, root finding, differentiation, integration, interpolation, linear systems solving, and matrix operations.
Netpop	NetPop -- a Netscape / AOL Email Popper written in Java, much like YPOPs! for Yahoo mail and Mr Postman for Hotmail, that allows anyone to use a POP3 email client (Thunderbird, Eutora, Outlook, etc) with Netscape and AOL webmail.
Ooo-tools	OOo Tools is a collection of tools that complement OpenOffice. The tools include a Writer properties viewer, a modified word counter, and a bibliography converter.
Ootpj	This project provides a web application, a stand-alone client, and a public API for managing "Out of the Park" baseball simulation leagues. Each of the interfaces (web, stand-alone, and API) provides full access to a huge amount of statistical, ratings,
Openworkbench	Open Workbench is a desktop application for project management and scheduling in which you can define a work breakdown structure, set dependencies and resource constraints, assign resources to tasks, auto schedule and then monitor progress.
Opsi	Opsi (open pc server integration) is a software management / deployment system. It consists of: 1. automatic software deployment for win32 based PCs 2. depotserver which holds the software packages, configuration infos 3. automatic OS-Installation
Osrecruiter	Java toolkit for creating and manipulating hr-xml (www.hr-xml.org) compliant resumes. Users construct resumes as java objects and convert them to .xml, .XHTML, or .pdf. Comes bundled with stylesheets and schema; also allows users to define their own.
Passreminder	PassReminder is a free password manager. It KEEPS PASSWORDS SECURE. Then it is easy to remember login and password for forum, operating system, bank.
Php-java-bridge	An optimized, XML-based network protocol which can be used to connect a native script engine with a persistent Java or ECMA 335 virtual machine. It is more than 50 times faster than a SOAP-based protocol, faster and more reliable than comm. via JNI.
Pred	PrEd is a Java based graphical utility to find and edit Java property files in JAR, WAR, EAR and other kind of ZIP archives. It is the perfect tool for customizing Java and J2EE application archives. Enjoy, Daniel Palomo van Es.
Radioactive	The first and only open source RFID software suite.
Remotecvs	A unique client-server combination allowing users to select their own build settings and watch as the server produces their own CVS build.

Project Name	Description of Application
Schemeway	A set of Eclipse plugins for the Scheme programming language. Features a powerful, fully extensible S-expression-based editor. Integrates seamlessly with any Scheme interpreter.
Scotland-yard	This is a game of Scotland Yard, which has been coded in Java. The moves of Mr. X are made by the program, while the human player makes the detective moves. The game works fine on both Windows and Linux platforms. However more work still needs to be done
Sg-ng	The SoftwareGroupNG project is an interactive website, targeted at bringing to the foreground of the software industry, Nigerian developers who are capable or are ready to learn to use the latest software technologies to design, implement and deploy solutions.
Shelled	ShellEd is a superb shell script editor for Eclipse. The great benefit of this plugin is the integration of man page information for content assist and text hover.
Shop	SHOP (Simple Hierarchical Ordered Planner), JSHOP, and SHOP2 are domain-independent automated-planning systems based on ordered task decomposition, which is a type of Hierarchical Task Network (HTN) planning.
Sokofinity	This is the Sokofinity project. The goal of this project is to recreate the classic NES game DuckHunt, only this time in 3D with Virtual Reality. Using an Infinity Box and Flock Of Birds positioning sensors, the game gets a new dimension.
Sonia	SoNIA (Social Network Image Animator) is a Java-based package for visualizing dynamic or longitudinal "network" data.
Spontane	Spontane is a 2d fast-paced fighting game, for play on your local computer, LAN, or Internet, on any OS which runs Java. Similar to Nintendo's Super Smash Bros.
Spring-rich-c	Java Swing application framework built on the Spring Framework with the goal of simplifying whats required to build professional, enterprise-ready rich client applications.
Strutsbox	StrutsBox is a visual Eclipse plugin toolkit for developing applications with Jakarta Struts Framework, currently the most used J2EE/MVC framework.
Swinggestures	Swing Gestures is an extension for the Java Swing GUI's, that allows to add mouse gestures to these interfaces in a generic, easy, platform independent and easily extensible way.
Webrcp	WebRCP is a framework for launching Eclipse RCP-Applications with Sun's Java Web Start. WebRCP supports the most common platforms.
Wsabi4axis	WSABI For Axis: A Java-based web application to manage Apache Axis deployments. Allows users to monitor and configure CRUD SOAP-based Web Services, Handlers (JAX-RPC and Axis), Global Configuration, etc. using an intuitive GUI.
Xmlbeansplug	Plug-ins for a number of IDE's to support Apache XMLBeans. Currently support is in development for Eclipse V3.0.

Project Name	Description of Application
Yale	Yale is a data mining, machine learning, knowledge discovery, business intelligence, learning, preprocessing, validation,visualization tool.

Table 3: Research Variable Descriptive Statistics (N=85)

Variable	Minimum	Maximum	Mean	Standard Deviation
Percentage of Tasks Closed	0.00	1.00	0.24	0.37
Change in Lines of Code	0.00	15596.70	335.45	1866.59
IBUC Platform Tenure Diversity (in years)	0.00	7.99	0.88	1.51
Development Group Platform Tenure Diversity (in years)	0.00	20.23	1.19	3.26
IBUC Relationship Density	0.00	28.00	4.36	7.15
Development Group Relationship Density	0.00	18.00	2.54	3.54
IBUC Dynamism	0.00	3.67	1.48	0.72
IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)	-40.95	76.07	5.59	14.03
IBUC-Development Group Communication * IBUC Relationship Density (centered)	-84.33	707.27	37.71	106.40
IBUC-Development Group Communication * IBUC Dynamism (centered)	-8.76	32.44	1.56	5.27
Application Type * IBUC Platform Tenure Diversity (in years) (centered)	-1.35	2.37	-0.06	0.51
Application Type * IBUC Relationship Density (centered)	-4.49	13.48	0.24	3.01
Application Type * IBUC Dynamism (centered)	-0.42	0.75	-0.05	0.21
IBUC-Development Group Communication	2.00	55.00	7.76	10.06
Application Type	0.00	1.00	0.19	0.39
Number of Project Members	2.00	37.00	7.20	6.17

Notes:
N=85

Table 4: Research Variables (N=85)

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1 Percentage of Tasks Closed	Pearson Correlation	1.00															
	Sig. (2-tailed)																
2 Change in Lines of Code	Pearson Correlation	0.03	1.00														
	Sig. (2-tailed)	0.77															
3 IBUC Platform Tenure Diversity (in years)	Pearson Correlation	0.37	0.16	1.00													
	Sig. (2-tailed)	0.00	0.14														
4 Squared Development Group Platform Tenure Diversity (in years)	Pearson Correlation	-0.02	0.00	0.09	1.00												
	Sig. (2-tailed)	0.88	0.97	0.39													
5 IBUC Relationship Density	Pearson Correlation	0.20	0.25	0.44	0.05	1.00											
	Sig. (2-tailed)	0.07	0.02	0.00	0.63												
6 Development Group Relationship Density	Pearson Correlation	0.23	0.22	0.34	0.49	0.30	1.00										
	Sig. (2-tailed)	0.03	0.04	0.00	0.00	0.01											
7 IBUC Dynamism	Pearson Correlation	0.27	0.06	0.39	0.15	0.30	0.19	1.00									
	Sig. (2-tailed)	0.01	0.57	0.00	0.16	0.00	0.08										

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
8 IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)	Pearson Correlation	0.12	0.53	0.06	-0.04	0.33	0.08	0.00	1.00								
	Sig. (2-tailed)	0.29	0.00	0.58	0.74	0.00	0.45	0.97									
9 IBUC-Development Group Communication * IBUC Relationship Density (centered)	Pearson Correlation	-0.02	0.53	0.21	-0.03	0.60	0.09	0.10	0.50	1.00							
	Sig. (2-tailed)	0.82	0.00	0.06	0.77	0.00	0.40	0.35	0.00								
10 IBUC-Development Group Communication * IBUC Dynamism(centered)	Pearson Correlation	-0.09	0.49	0.01	0.05	0.21	0.05	0.06	0.49	0.37	1.00						
	Sig. (2-tailed)	0.39	0.00	0.94	0.67	0.05	0.67	0.59	0.00	0.00							
11 Application Type * IBUC Platform Tenure Diversity (in years) (centered)	Pearson Correlation	-0.20	-0.08	-0.18	0.12	0.12	0.25	-0.03	0.06	0.04	0.06	1.00					
	Sig. (2-tailed)	0.07	0.44	0.10	0.27	0.28	0.02	0.76	0.60	0.73	0.57						
12 Application Type* IBUC Relationship Density (centered)	Pearson Correlation	0.13	-0.14	0.10	0.08	0.09	0.24	-0.02	-0.04	-0.21	0.03	0.66	1.00				
	Sig. (2-tailed)	0.22	0.21	0.39	0.48	0.41	0.03	0.84	0.75	0.06	0.80	0.00					
13 Application Type * IBUC Dynamism(centered)	Pearson Correlation	-0.13	-0.02	-0.04	0.00	-0.03	0.05	-0.36	0.12	0.12	0.02	0.51	0.35	1.00			
	Sig. (2-tailed)	0.23	0.86	0.72	0.99	0.77	0.67	0.00	0.27	0.27	0.82	0.00	0.00				
14 IBUC-Development Group Communication	Pearson Correlation	0.24	0.53	0.37	0.04	0.53	0.22	0.22	0.61	0.70	0.39	0.02	-0.09	0.10	1.00		
	Sig. (2-tailed)	0.03	0.00	0.00	0.70	0.00	0.04	0.05	0.00	0.00	0.00	0.84	0.44	0.37			

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
15 Application Type	Pearson Correlation	0.19	-0.06	-0.10	-0.07	0.09	-0.01	-0.18	0.02	-0.06	0.10	-0.18	0.13	-0.37	-0.15	1.00	
	Sig. (2-tailed)	0.08	0.56	0.37	0.50	0.44	0.96	0.11	0.84	0.58	0.35	0.09	0.25	0.00	0.16		
16 Number of Project Members	Pearson Correlation	0.18	0.47	0.39	0.14	0.55	0.43	0.27	0.52	0.59	0.45	0.16	0.06	0.06	0.72	-0.08	1.00
	Sig. (2-tailed)	0.10	0.00	0.00	0.22	0.00	0.00	0.01	0.00	0.00	0.00	0.15	0.61	0.56	0.00	0.47	

Notes:
N=85

Table 5: Survival –Tasks Closed

Variable	Main Effects
IBUC Platform Tenure Diversity	-.21*
Development Group Platform Tenure Diversity	.01
IBUC Dynamism	-.6***
Application Type	-.68*
IBUC-Development Group Communication	-.03
Number of Project Members	-.02
R2	.24
Likelihood Ratio Test	23***

Notes:
N=85

* p<.1, **p<.05, ***p<.01

Table 6: Development Group Activity Intensity – Percentage of Tasks Closed

Variable	Main Effects (Model A)	Moderated Effects IBUC-Development Group Communication (Model B)	Moderated Effects Application Type (Model C)	Moderated Effects IBUC-Development Group Communication Application Type (Model D)
IBUC Platform Tenure Diversity (in years)	.23*	.18	.05	.02
Squared Development Group Platform Tenure Diversity (in years)	-.13	-.11	-.12	-.1
IBUC Relationship Density	-.12	.02	-.06	.01
Development Group Relationship Density	.22*	.15	.25*	.19
IBUC Dynamism	.26**	.24**	.3**	.31**
IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)		.12		.11
IBUC-Development Group Communication * IBUC Relationship Density (centered)		-.3*		-.23
IBUC-Development Group Communication * IBUC Dynamism (centered)		-.2*		-.22*
Application Type * IBUC Platform Tenure Diversity (in years) (centered)			-.4**	-.37**
Application Type * IBUC Relationship Density (centered)			.34**	.27*
Application Type * IBUC Dynamism (centered)			.09	.13
IBUC-Development Group Communication	.26*	.37**	.27*	.35**
Application Type	.3***	.31***	.21*	.26**
Number of Project Members	-.10	.01	-.06	.05
R^2	.3	.36	.35	.36
F	4***	3.77***	3.6***	3.77***

Notes:
N=85
* p<.1, **p<.05, ***p<.01

Table 7: Research Variable Descriptive Statistics (N=73)

Variable	Minimum	Maximum	Mean	Standard Deviation
Percentage of Tasks Closed	0.00	1.00	0.27	0.38
Change in Lines of Code	0.00	15596.70	295.69	1855.47
IBUC Platform Tenure Diversity (in years)	0.00	7.99	0.96	1.58
Development Group Platform Tenure Diversity (in years)	0.00	20.23	1.10	3.21
IBUC Relationship Density	0.00	28.00	4.58	7.31
Development Group Relationship Density	0.00	18.00	2.75	3.73
IBUC Dynamism	0.00	3.67	1.55	0.76
IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)	-41.33	72.10	5.58	14.46
IBUC-Development Group Communication * IBUC Relationship Density (centered)	-84.79	697.71	38.08	108.50
IBUC-Development Group Communication * IBUC Dynamism (centered)	-9.01	29.06	1.72	5.25
Application Type * IBUC Platform Tenure Diversity (in years) (centered)	-1.48	2.25	-0.07	0.55
Application Type * IBUC Relationship Density (centered)	-4.92	12.97	0.29	3.15
Application Type * IBUC Dynamism (centered)	-0.44	0.67	-0.06	0.23
IBUC-Development Group Communication	2.00	55.00	7.88	10.45
Application Type	0.00	1.00	0.21	0.41
Number of Project Members	2.00	37.00	7.15	6.14

Notes:
N=73

Table 8: Research Variable Correlations (N=73)

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1 Percentage of Tasks Closed	Pearson Correlation	1.00															
	Sig. (2-tailed)																
2 Change in Lines of Code	Pearson Correlation	0.06	1.00														
	Sig. (2-tailed)	0.62															
3 IBUC Platform Tenure Diversity (in years)	Pearson Correlation	0.34	0.13	1.00													
	Sig. (2-tailed)	0.00	0.28														
4 Squared Development Group Platform Tenure Diversity (in years)	Pearson Correlation	0.01	-0.05	0.11	1.00												
	Sig. (2-tailed)	0.93	0.67	0.35													
5 IBUC Relationship Density	Pearson Correlation	0.19	0.16	0.41	0.05	1.00											
	Sig. (2-tailed)	0.10	0.19	0.00	0.70												
6 Development Group Relationship Density	Pearson Correlation	0.23	0.24	0.34	0.56	0.31	1.00										
	Sig. (2-tailed)	0.05	0.04	0.00	0.00	0.01											
7 IBUC Dynamism	Pearson Correlation	0.23	0.09	0.37	0.20	0.31	0.17	1.00									
	Sig. (2-tailed)	0.05	0.46	0.00	0.09	0.01	0.14										

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
8 IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)	Pearson Correlation	0.09	0.51	0.00	-0.05	0.28	0.08	-0.02	1.00								
	Sig. (2-tailed)	0.43	0.00	0.98	0.69	0.02	0.50	0.88									
9 IBUC-Development Group Communication * IBUC Relationship Density (centered)	Pearson Correlation	-0.03	0.47	0.17	-0.06	0.56	0.09	0.10	0.46	1.00							
	Sig. (2-tailed)	0.79	0.00	0.15	0.59	0.00	0.46	0.38	0.00								
10 IBUC-Development Group Communication * IBUC Dynamism(centered)	Pearson Correlation	-0.15	0.60	-0.02	0.08	0.22	0.02	0.00	0.47	0.37	1.00						
	Sig. (2-tailed)	0.20	0.00	0.84	0.53	0.06	0.87	0.99	0.00	0.00							
11 Application Type * IBUC Platform Tenure Diversity (in years) (centered)	Pearson Correlation	-0.21	-0.07	-0.21	0.12	0.12	0.24	-0.03	0.07	0.05	0.05	1.00					
	Sig. (2-tailed)	0.07	0.53	0.07	0.33	0.31	0.04	0.83	0.53	0.66	0.66						
12 Application Type* IBUC Relationship Density (centered)	Pearson Correlation	0.12	-0.10	0.10	0.09	0.09	0.23	-0.05	-0.03	-0.21	0.00	0.63	1.00				
	Sig. (2-tailed)	0.30	0.39	0.41	0.45	0.46	0.05	0.70	0.81	0.07	0.97	0.00					
13 Application Type * IBUC Dynamism(centered)	Pearson Correlation	-0.14	-0.04	-0.03	-0.04	-0.07	0.05	-0.36	0.12	0.12	0.03	0.49	0.33	1.00			
	Sig. (2-tailed)	0.25	0.73	0.81	0.74	0.58	0.68	0.00	0.33	0.31	0.82	0.00	0.00				
14 IBUC-Development Group Communication	Pearson Correlation	0.22	0.51	0.34	0.03	0.51	0.22	0.22	0.57	0.69	0.35	0.03	-0.09	0.10	1.00		
	Sig. (2-tailed)	0.07	0.00	0.00	0.79	0.00	0.06	0.06	0.00	0.00	0.00	0.79	0.46	0.40			

Variable	Statistic	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
15 Application Type	Pearson Correlation	0.19	-0.06	-0.11	-0.06	0.10	-0.01	-0.21	0.04	-0.06	0.12	-0.19	0.13	-0.40	-0.15	1.00	
	Sig. (2-tailed)	0.10	0.64	0.34	0.62	0.40	0.92	0.07	0.76	0.60	0.32	0.11	0.26	0.00	0.19		
16 Number of Project Members	Pearson Correlation	0.16	0.56	0.40	0.16	0.62	0.46	0.30	0.52	0.66	0.42	0.15	0.03	0.04	0.78	-0.06	1.00
	Sig. (2-tailed)	0.18	0.00	0.00	0.17	0.00	0.00	0.01	0.00	0.00	0.00	0.20	0.82	0.74	0.00	0.60	

Notes:
N=73

Table 9: Survival - Change in Lines of Code

Variable	Main Effects
IBUC Platform Tenure Diversity	-.10
Development Group Platform Tenure Diversity	.02
IBUC Dynamism	-.44**
Application Type	-.63*
IBUC-Development Group Communication	-.04
Number of Project Members	.00
R2	.18
Likelihood Ratio Test	16.6***

Notes:
N=85

* p<.1, **p<.05, ***p<.01

Table 10: Development Group Activity Intensity – Change in Lines of Code

Variable	Main Effects (Model A)	Moderated Effects IBUC-Development Group Communication (Model B)	Moderated Effects Application Type (Model C)	Moderated Effects IBUC-Development Group Communication Application Type (Model D)
IBUC Platform Tenure Diversity (in years)	-.09	.02	-.21	-.09
Squared Development Group Platform Tenure Diversity (in years)	-.3**	-.38**	-.31***	-.38***
IBUC Relationship Density	-.37**	-.43***	-.31***	-.39***
Development Group Relationship Density	.29**	.47***	.35***	.53***
IBUC Dynamism	.04*	.11	-.02	.01
IBUC-Development Group Communication * IBUC Platform Tenure Diversity (in years) (centered)		.09		.1
IBUC-Development Group Communication * IBUC Relationship Density (centered)		.23*		.29**
IBUC-Development Group Communication * IBUC Dynamism (centered)		.43***		.42***
Application Type * IBUC Platform Tenure Diversity (in years) (centered)			-.26	-.25*
Application Type * IBUC Relationship Density (centered)			.1	.16
Application Type * IBUC Dynamism (centered)			-.1	.2
IBUC-Development Group Communication	.2	.07	.18	.04
Application Type	.03	-.02	-.1	-.2*
Number of Project Members	.52***	.15	.58***	.18
R^2	.41	.58	.45	.64
F	5.53***	7.67***	4.59***	7.2***

Notes:
N=73
* p<.1, **p<.05, ***p<.01

Table 11: Summary of Hypotheses

Hypothesis	Percentage of Tasks Closed	Change in Lines Of Code
H1: IBUC platform tenure diversity is positively associated with the survival of an OSS Project.	**Supported** **(Table 5)**	Not Supported (Table 9)
H2: Development group platform tenure diversity is positively associated with the survival of an OSS Project.	Not Supported (Table 5)	Not Supported (Table 9)
H3: IBUC dynamism is positively associated with the survival of an OSS Project.	**Supported** **(Table 5)**	**Supported** **(Table 9)**
H4: IBUC platform tenure diversity is negatively associated with development group activity intensity.	**Significant – Wrong Direction** **(Table 6 Model A)**	Not Supported (Table 10 Model A)
H5: Development group platform tenure diversity has a quadratic association with development group activity intensity.	Not Supported (Table 6 Model D)	**Supported** **(Table 10 Model D)**
H6: IBUC relationship density is positively associated with development group activity intensity.	Not Supported (Table 6 Model A)	**Significant – Wrong Direction** **(Table 10 Model A)**
H7: Development group relationship density is positively associated with development group activity intensity.	Not Supported (Table 6 Model D)	**Supported** **(Table 10 Model D)**
H8: IBUC dynamism is positively associated with development group activity intensity.	**Supported** **(Table 6 Model A)**	**Supported** **(Table 10 Model A)**
H9: The positive relationship between IBUC relationship density and development group activity intensity will be stronger for application types that target developers.	**Significant – Wrong Direction** **(Table 6 Model D)**	Not Supported (Table 10 Model D)
H10: The positive relationship between IBUC dynamism and development group activity intensity will be stronger for application types that target developers.	Not Supported (Table 6 Model D)	Not Supported (Table 10 Model D)
H11: The positive relationship between IBUC relationship density and development group activity intensity will be positively moderated by IBUC-development group communication.	Not Supported (Table 6 Model D)	**Significant – Wrong Direction** **(Table 10 Model D)**

Hypothesis	Percentage of Tasks Closed	Change in Lines Of Code
H12: The positive relationship between IBUC dynamism and development group activity intensity will be positively moderated by IBUC-development group communication.	**Significant – Wrong Direction** **(Table 6 Model D)**	**Supported** **(Table 10 Model D)**
H13: The negative relationship between IBUC platform tenure diversity and development group activity intensity will be more intense when there is strong IBUC-development group communication.	Not Supported (Table 6 Model D)	Not Supported (Table 10 Model D)
H14: The negative relationship between IBUC platform tenure diversity and development group activity intensity will be more intense for applications targeted to developers.	**Supported** **(Table 6 Model D)**	**Significant – Wrong Direction** **(Table 10 Model D)**

Figure 1: Conceptual Model

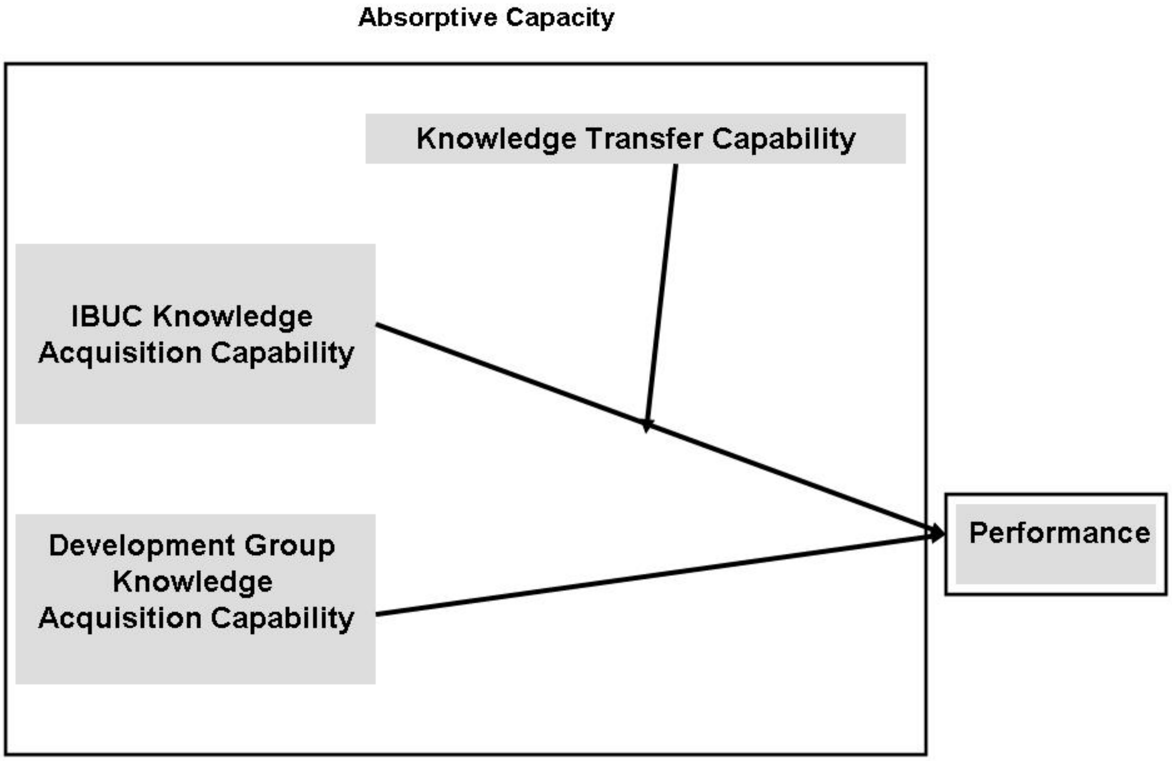

Absorptive Capacity
Knowledge Transfer Capability
IBUC Knowledge Acquisition Capability
Development Group Knowledge Acquisition Capability
Performance

Figure 2: Survival Research Model

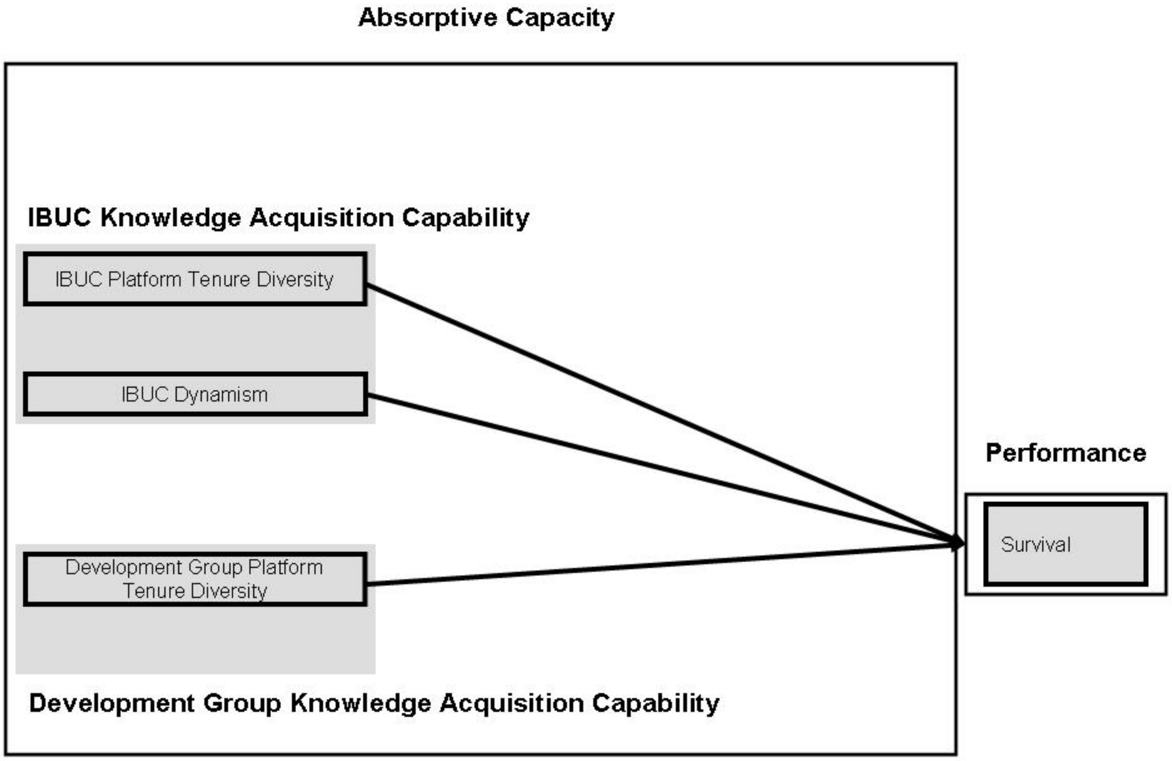

Absorptive Capacity
IBUC Knowledge Acquisition Capability
IBUC Platform Tenure Diversity
IBUC Dynamism
Development Group Platform Tenure Diversity
Development Group Knowledge Acquisition Capability
Performance
Survival

Figure 3: Development Group Activity Intensity Research Model

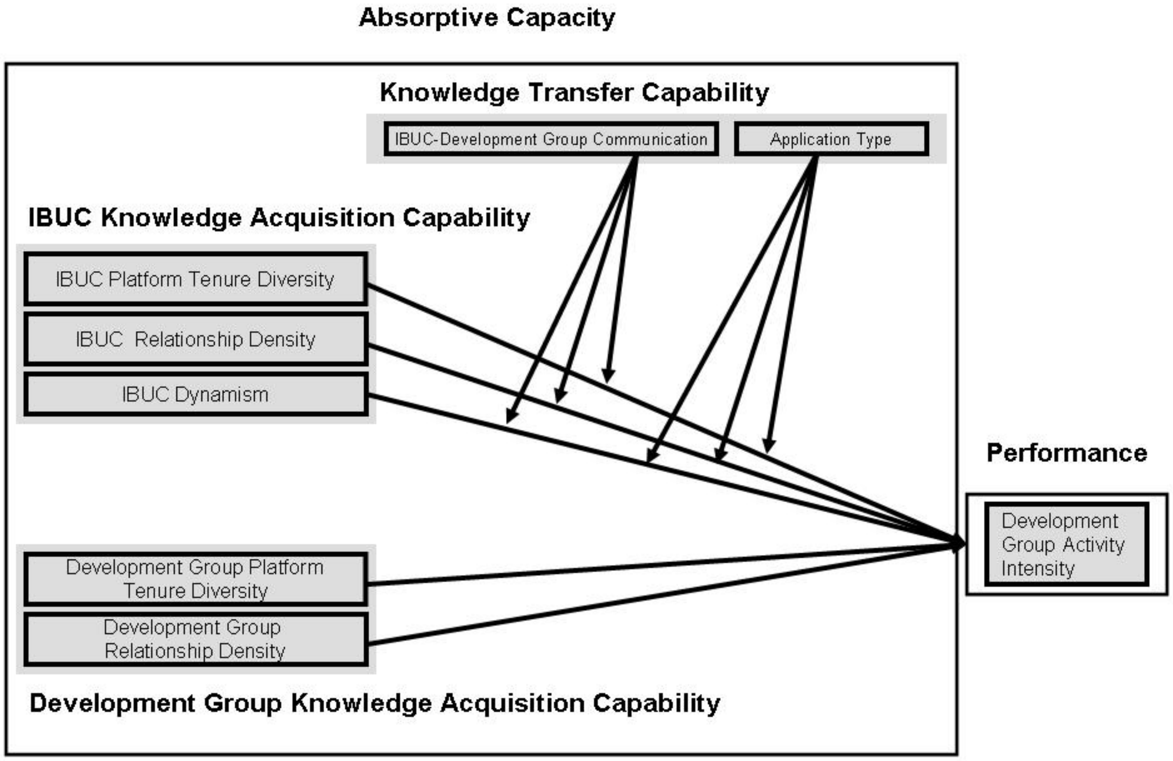
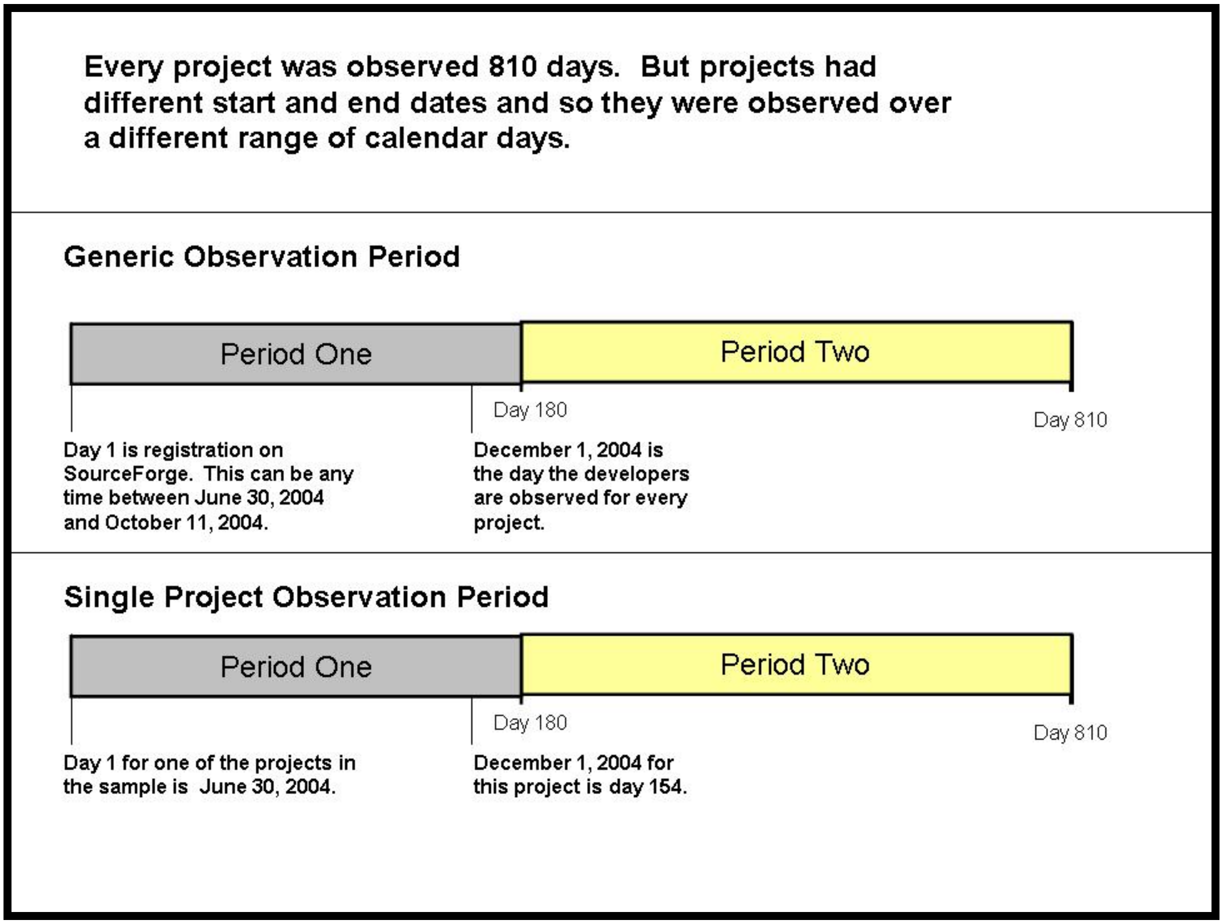

Absorptive Capacity
Knowledge Transfer Capability
IBUC-Development Group Communication
Application Type
IBUC Knowledge Acquisition Capability
IBUC Platform Tenure Diversity
IBUC Relationship Density
IBUC Dynamism
Performance
Development Group Activity Intensity
Development Group Platform Tenure Diversity
Development Group Relationship Density
Development Group Knowledge Acquisition Capability

Figure 4: Observation Periods



Every project was observed 810 days. But projects had different start and end dates and so they were observed over a different range of calendar days.

Generic Observation Period

Day 1 is registration on SourceForge. This can be any time between June 30, 2004 and October 11, 2004.

December 1, 2004 is the day the developers are observed for every project.

Single Project Observation Period

Day 1 for one of the projects in the sample is June 30, 2004.

December 1, 2004 for this project is day 154.

Figure 5: Percentage of Tasks Closed Kaplan-Meier Plot

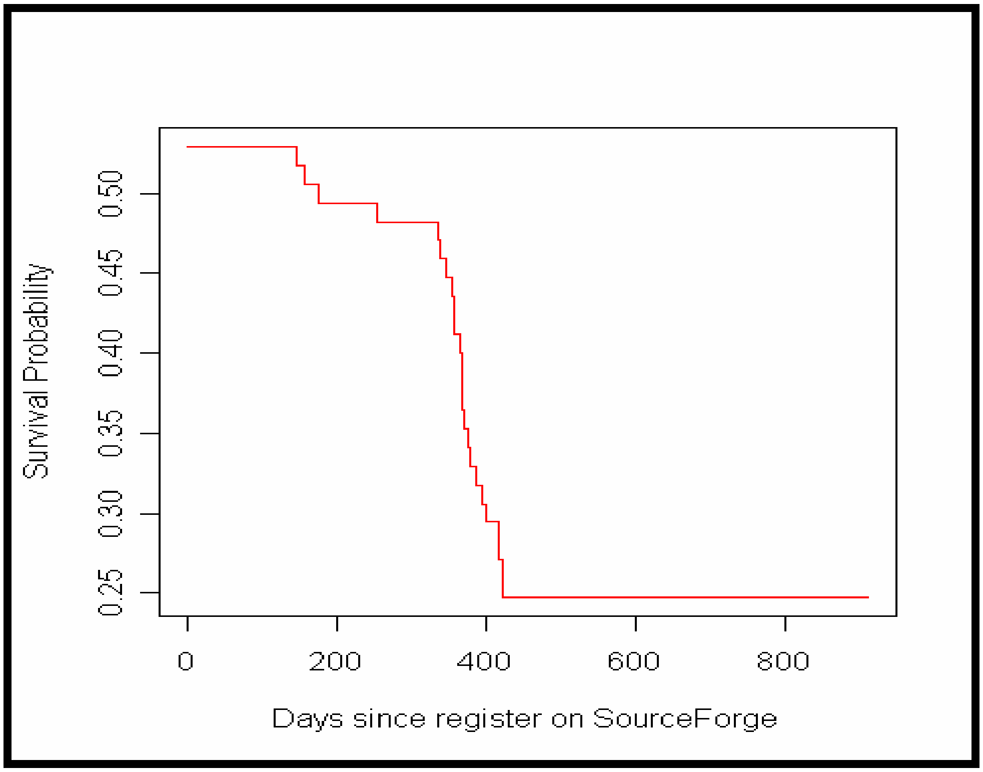


Figure 6: Percentage of Tasks Closed IBUC-Development Group Communication - IBUC Relationship Density Interaction

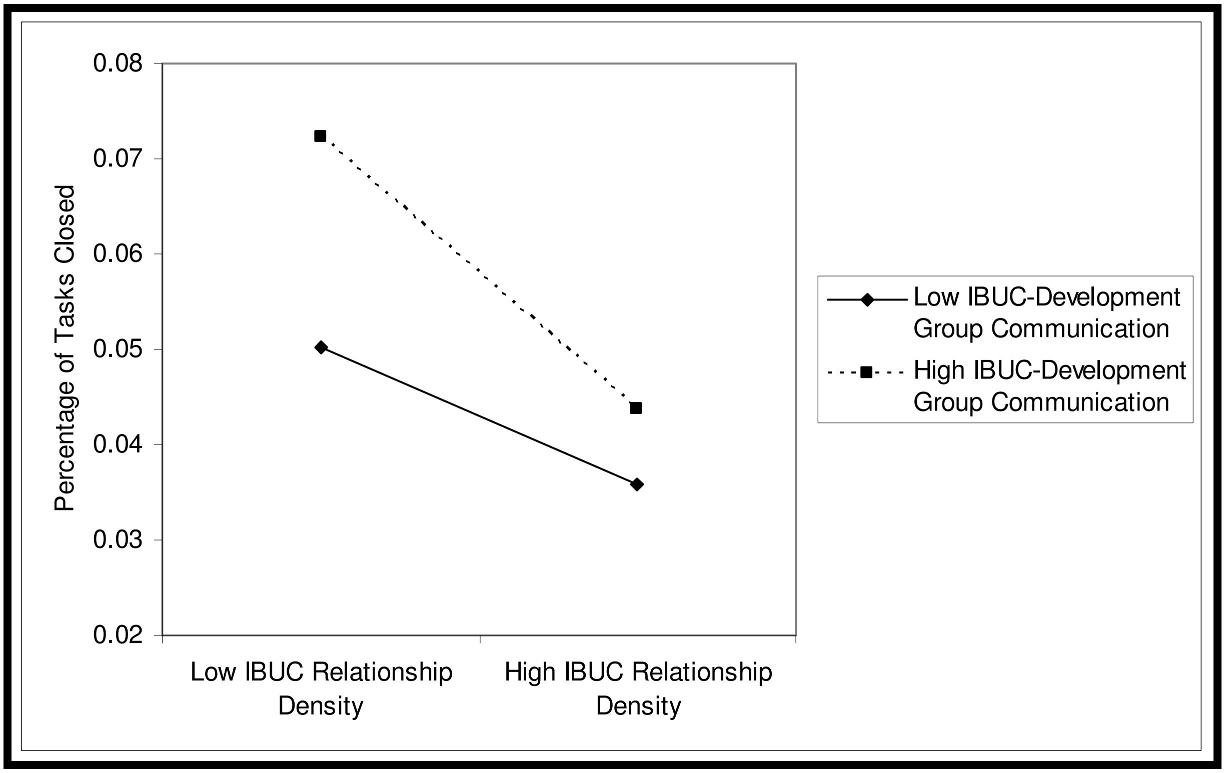


Figure 7: Percentage of Tasks Closed IBUC-Development Group Communication - IBUC Dynamism Interaction

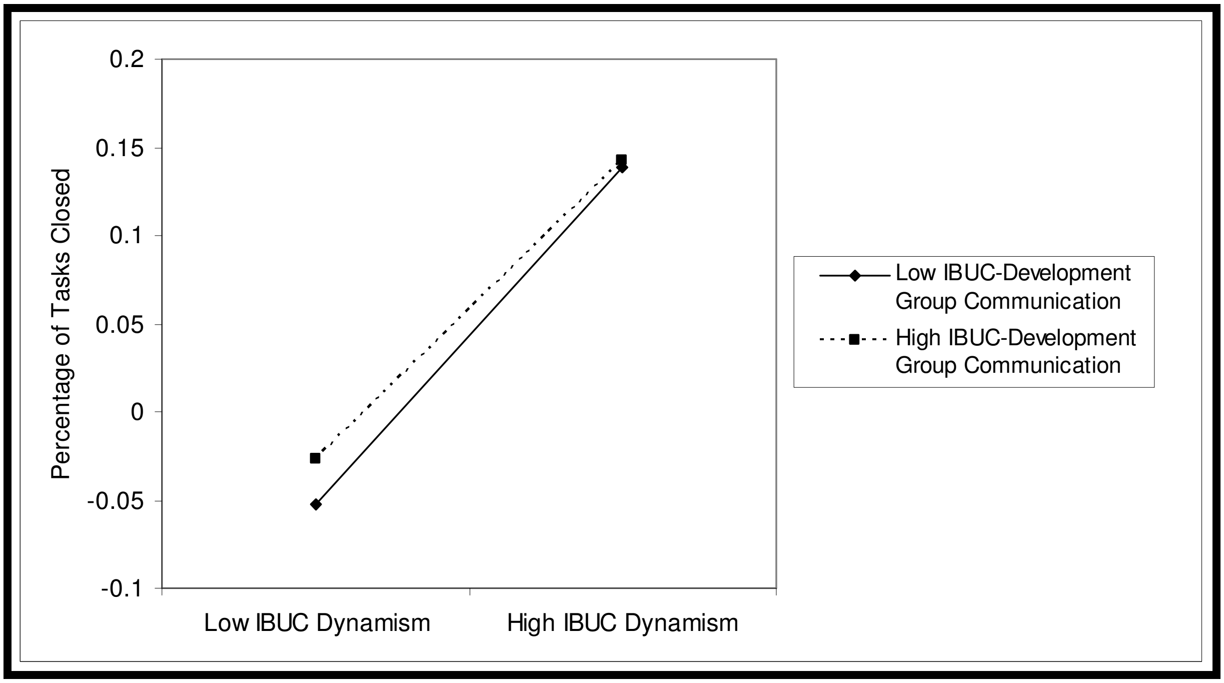


Figure 8: Percentage of Tasks Closed Application Type - IBUC Platform Tenure Diversity Interaction

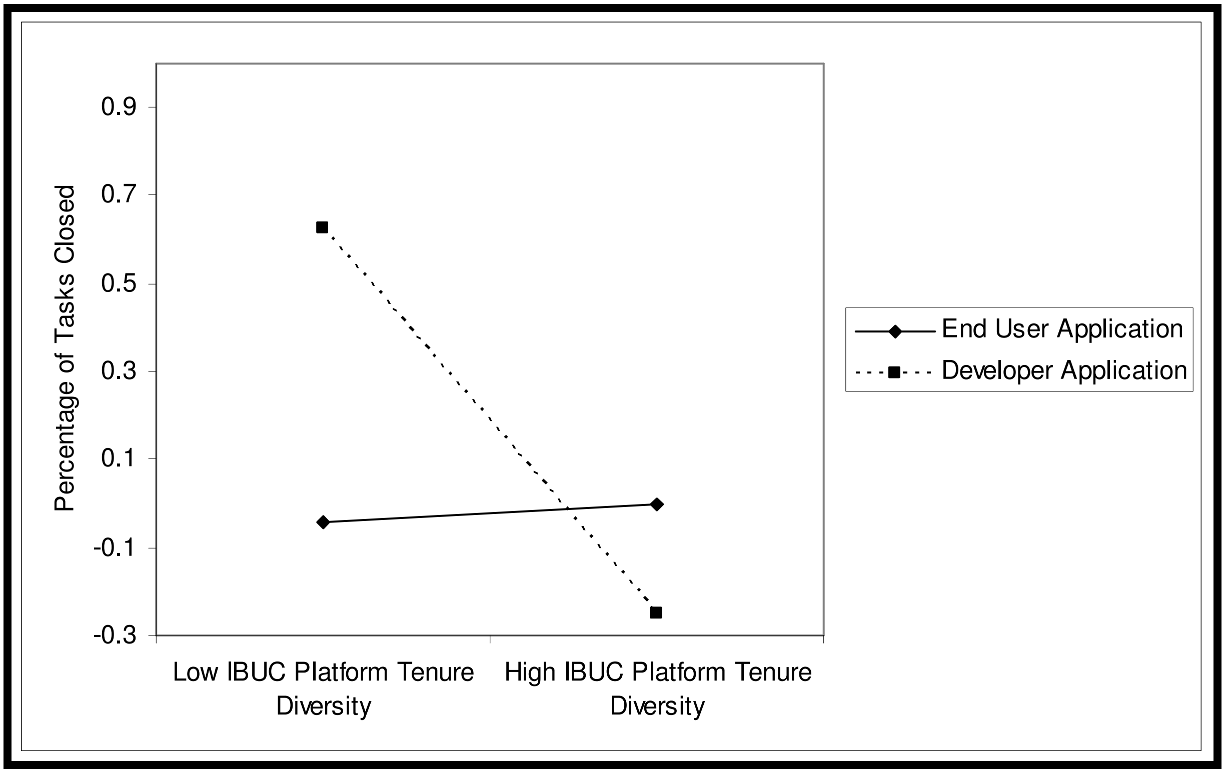


Figure 9: Percentage of Tasks Closed Application Type - IBUC Relationship Density Interaction

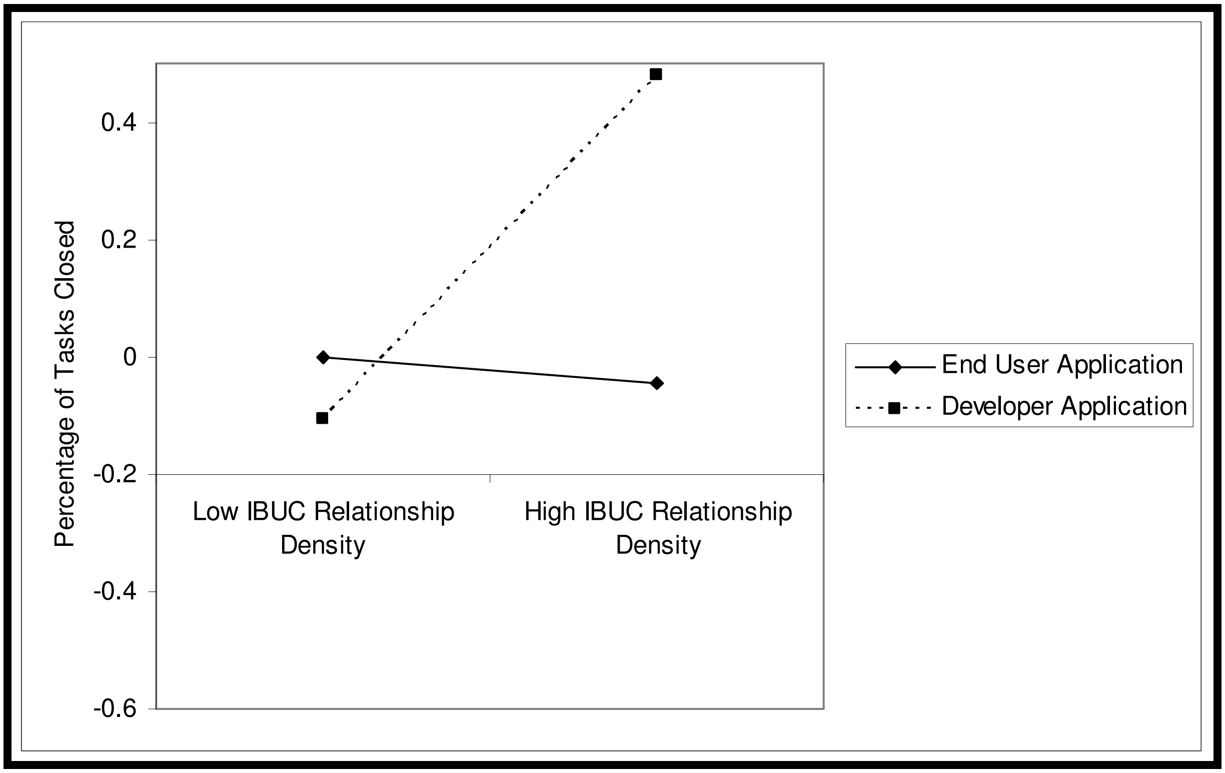


Figure 10: Change in Lines of Code Kaplan-Meier Plot

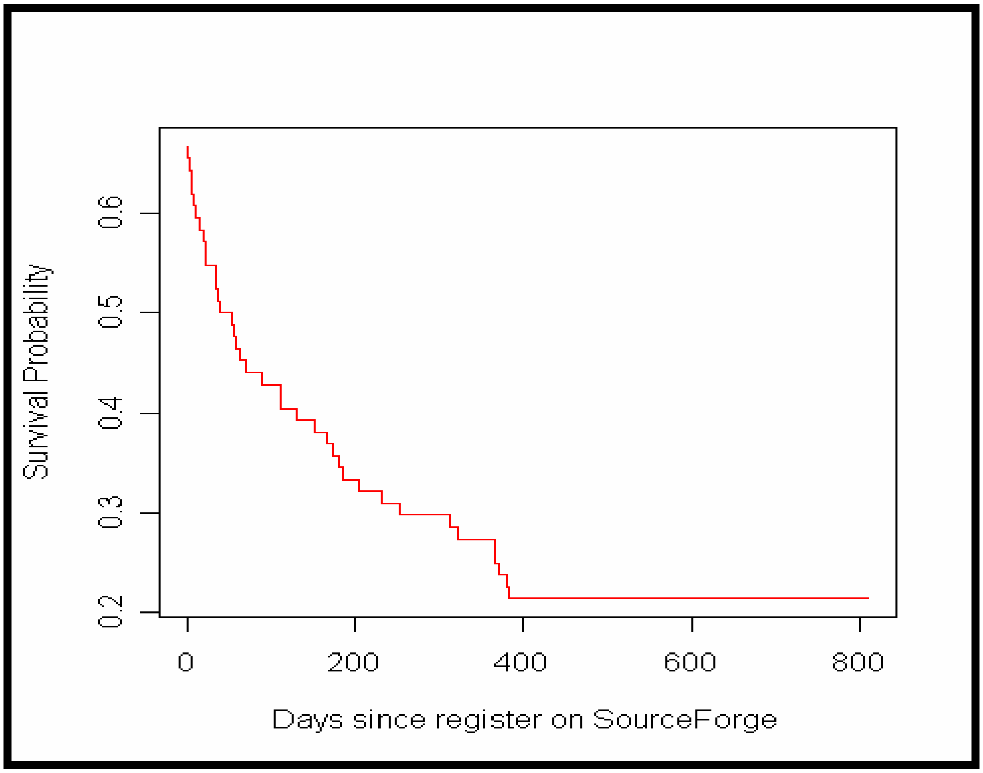


Figure 11: Change in Lines of Code IBUC-Development Communication - IBUC Relationship Density

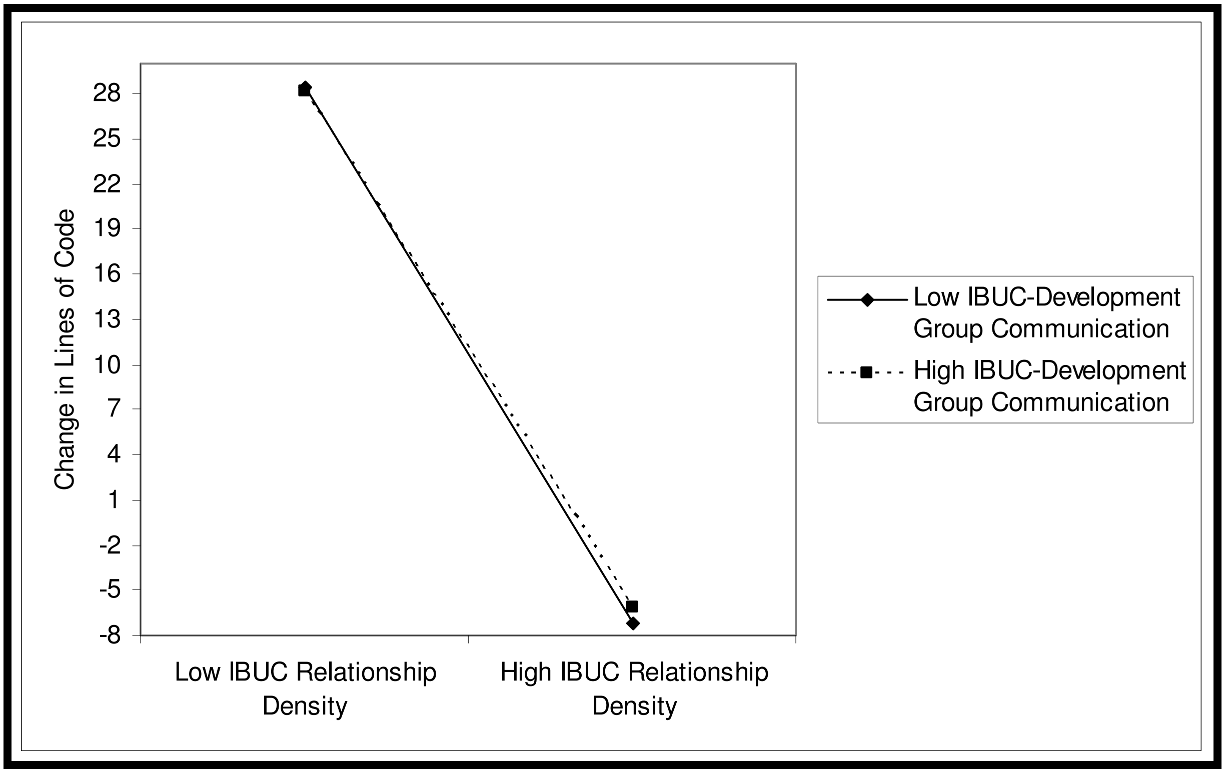


Figure 12: Change in Lines of Code IBUC - Development Group Communication - IBUC Dynamism

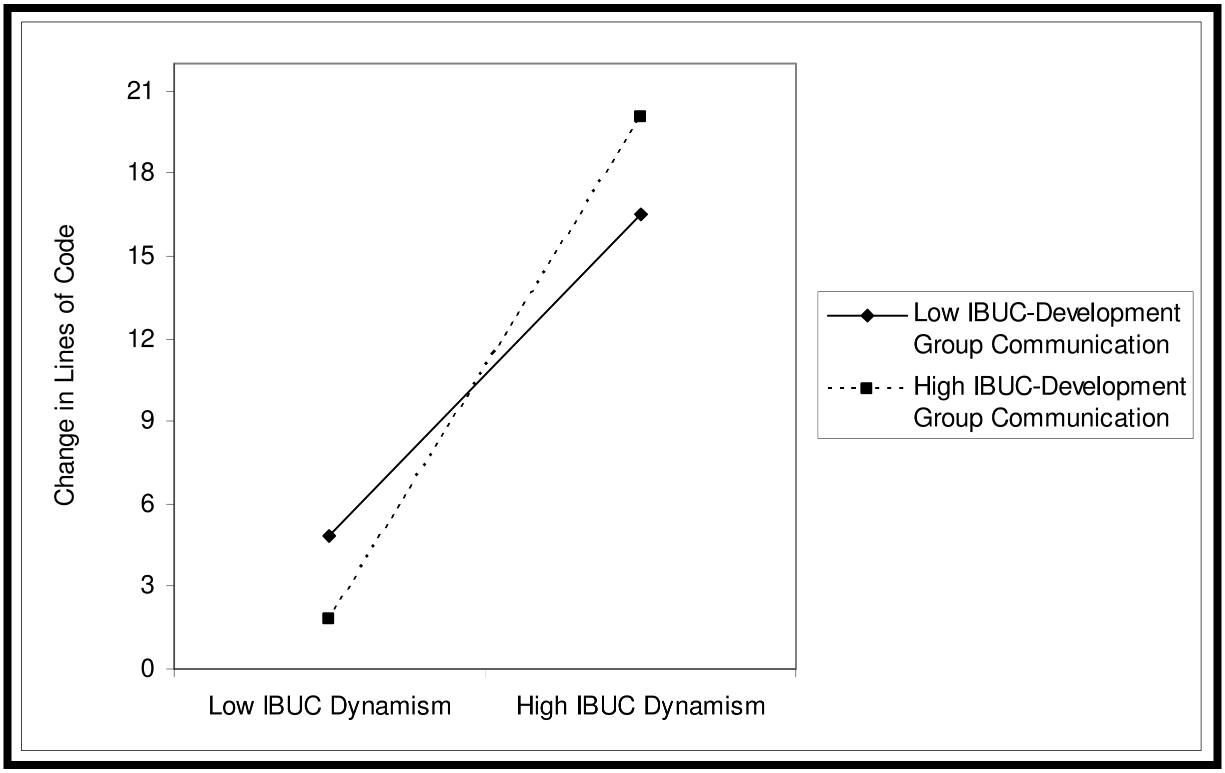
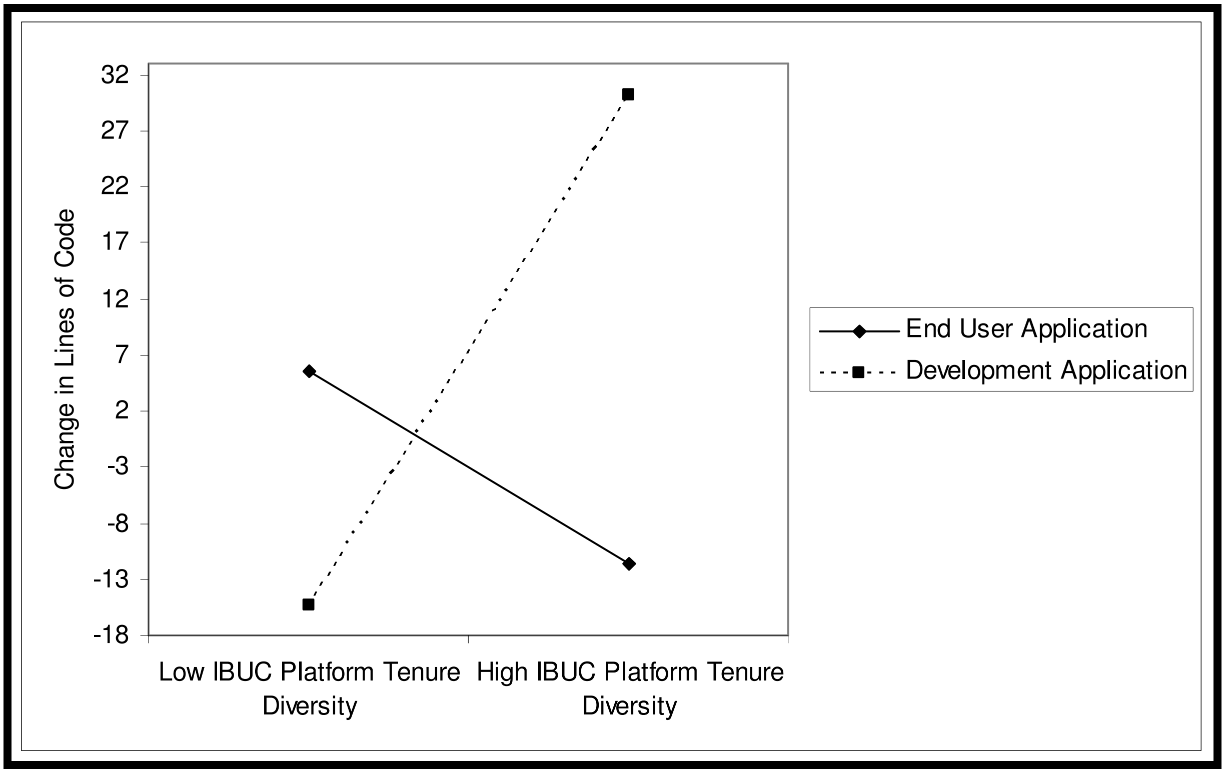


Figure 13: Change in Lines of Code Application Type - IBUC Platform Tenure Diversity



REFERENCES

Abelson, M., and Baysinger, B. "Optimal and Dysfunctional Turnover: Toward an Organizational Level Model," *Academy of Management Review* (9:2) 1984, pp 331-341.

Bagozzi, R., and Dholakia, U. "Open Source Software User Communities: A Study of Participation in Linux User Groups," *Management Science* (52:7) 2006, pp 1099-1115.

Baldwin, C., and Clark, K. "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model," *Management Science* (52:7) 2006, pp 1116-1127.

Baum, J.A.C., and Ingram, P. "Survival-enhancing learning in the Manhattan hotel industry, 1898-1980," *Management Science* (44) 1998, pp 996-1016.

Bergquist, M., and Ljungberg, J. "The power of gifts: organizing social relationships in open source communities," *Information Systems Journal* (11:4) 2001, pp 305-320.

Bessen, J. "Open Source Software: Free Provision of Complex Public Goods," Research on Innovation, pp. 1-27.

Bower, G., and Hligard, E.R. *Theories of learning* Prentice-Hall, Englewood Cliffs, 1981.

Bretthauer, D. "Open Source Software: A History," *Information Technology and Libraries* (21:1) 2002, pp 3-11.

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

Chengalur-Smith, S., and Sidorova, A. "Survival of open-source projects: A population ecology perspective," International Conference on Information Systems, Seattle Washington, 2003.

Clark, H. *Using Language* Cambridge University Press, New York, 1996, p. 432.

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

Cook, R.D. "Influential Observations in Linear Regression," *Journal of American Statistical Association* (74) 1979, pp 169-174.

Cramton, C.D. "The mutual knowledge problem and its consequences for dispersed collaboration," *Organization Science* (12:3) 2001, pp 346-371.

Crowston, K., Annabi, H., and Howison, J. "Defining Open Source Software Project Success," International Conference on Information Systems, Seattle, WA, 2003.

Crowston, K., Annabi, H., Howison, J., and Masango, C. "Effective work practices for Software Engineering: Floss Development: A Model and Hypotheses " Hawaii International Conference on System Science, Kona, Hawaii, 2005.

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

Dalton, D., and Todor, W. "Turnover turned over: An expanded and positive perspective," *Academy of Management. The Academy of Management Review* (4:000002) 1979, pp 225-235.

Dearborn, D., and Simon, H. "Selective Perception: A Note on the Departmental Identifications of Executives," *Sociometry* (21:2) 1958, pp 140-144.

DeCarolis, D., and Deeds, D. "The impact of stocks and flows of organizational knowledge on firm performance: An empirical investigation of the biotechnology industry," *Strategic Management Journal* (20:10) 1999, p 953.

Dellarocas, C. "The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms," *Management Science* (49:10) 2003, pp 1407-1424.

DeLone, W.H., and McLean, E.R. "Information Systems Success: The Quest for the Dependent variable," *Information Systems Research* (3:1) 1992, pp 60-94.

Dempsey, B.J., Weiss, D., Jones, P., and Greenberg, J. "Who is an Open Source Software Developer?," *Communication of the ACM* (45:2) 2002, pp 67-72.

Dinh-Trong, T., and Bieman, J.M. "The FreeBSD Project: A Replication Case Study of Open Source Development," *IEEE Transactions on Software Engineering* (31:6) 2005, pp 481-494.

Divitini, M., Jaccheri, L., Monteiro, E., and Traetteberg, H. "Open source processes: no place for politics?," in: *International Conference on Software Engineering*, Portland Oregon, 2003.

Dreu, C.D., and Weingart, L. "Task versus relationship conflict, team performance, and team member satisfaction: A meta-analysis," *Journal of Applied Psychology* (88) 2003, pp 741-749.

Ellis, H.C. *The Transfer of Learning* MacMillan, New York, 1965.

Epple, D., L. Argote, and Devadas, R. "Organizational learning curves: A method for investigating intra-plant transfer acquired through learning by doing," *Organization Science* (2) 1991, pp 58-70.

Ethiraj, S., Kale, P., Krishnan, M.S., and Singh, J. "Where do capabilities come from and how do they matter? A study in the software services industry," *Strategic Management Journal* (26:1) 2005, pp 25-46.

Feller, J., and Fitzgerald, B. "A framework analysis of the open source software development paradigm," ICIS, Association for Information Systems, Brisbane, Australia, 2000, pp. 58-69.

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

Fichman, R., and Kemerer, C. "The Illusory Diffusion of Innovation: An Examination of Assimilation Gaps," *Information Systems Research* (10:3) 1999, pp 255-275.

Fitzgerald, B. "The Transformation of Open Source Software," *Management Information Systems Quarterly* (30:3) 2006, pp 587-598.

Fussell, S., and Krauss, R. "Coordination of knowledge in communication: effects of speakers' assumptions about what others know," *Journal of personality and social psychology* (62:3) 1992, pp 378-391.

Gallivan, M.J. "Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies," *Information Systems Journal* (11:4) 2001, pp 277-304.

Ghosh, R.A. "Free/Libre Open Source Software: Survey and Study," in: *Workshop on Advancing the Research Agenda on Free/Open Source Software*, Brussels, Netherlands, 2002.

Giddens *New Rules of Sociological Method* Basic Books, New York, 1976.

Godfrey, M.W., and Tu, Q. "Evolution in Open Source Software: A Case Study," *working paper*) 2000, pp 131-142.

Goldstein, I.L. "Training in work organizations," in: *Handbook of industrial and organizational psychology,* M.D.D.a.L.M. Hough (ed.), Consulting Psychologists Press, Palo Alto, CA, 1991.

Grant, R. "Prospering in dynamically-competitive environments: Organizational capability as knowledge integration," *Organization Science* (7:4) 1996, pp 375-389.

Grewal, R., Lilien, G.L., and Mallapragada, G. "Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems," *Management Science* (52:7) 2006, p 1043.

Hann, I.-H., Roberts, J., and Slaughter, S. "Why Developers Participate in Open Source Software

Projects: An Empirical Investigation," International Conference on Information Systems, Washington DC, 2004.

Hann, I.-H., Roberts, J., Slaughter, S.A., and Fielding, R. "Economic Incentives for Participating in open Source Software Projects," 23rd International Conference on Information Systems, Barcelona, Spain, 2002, pp. 365-372.

Hars, A., and Ou, S. "Working for Free? Motivations for Participating in Open Source Projects," *International Journal of Electronic Commerce* (6:3) 2002, pp 25-39.

Hertel, G., Niedner, S., and Herrmann, S. "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel," *Research Policy* (32) 2003, pp 1159-1177.

Hippel, E.v. "Sticky Information and the Locus of Problem Solving: Implications for Innovation," *Management Science* (40:4) 1994, pp 429-439.

Hippel, E.V. "Innovation by User Communities: Learning from Open Source Software," *Sloan Management Review*) 2001.

Iansiti, M., and MacCormack, A. "Developing Products on Internet Time," *Harvard Business Review* (75:5) 1997, pp 108-117.

Katz, D., and Kahn, R. *The Social Psychology of Organizations* John Wiley and Sons, New York, 1966, pp. 14-29.

Kogut, B., and Metiu, A. "Open source software development and distributed innovation," *Oxford Review of Economic Policy* (17:2) 2001, pp 248-264.

Kogut, B., and Zander, U. "Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology," *Organization Science* (3:3) 1992, pp 383-397.

Krishnamurthy, S. "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects," in: *working paper*, 2002.

Krogh, G.v., Spaeth, S., and Lakhani, K. "Community, joining, and specialization in open source software innovation: a case study," *Research Policy* (32) 2003, pp 1217-1241.

Kuan, J. "OSS as Lead User's Make or By Decision: A Study of Open and Closed Source Quality " Open Source Software : Economics, Law and Policy, Toulouse, France, 2002.

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

Lane, P., Koka, B., and Pathak, S. "The Reification of Absorptive Capacity: A Critical Review and Rejuvination of the Construct," *Academy of Management Review* (31:4) 2006, pp 833-863.

Lee, G., and Cole, R. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development," *Organization Science* (14:6) 2003, pp 633-649.

Lenox, M., and King, A. "Prospects for Developing Absorptive Capacity through Internal Information Provision," *Strategic Management Journal* (25:4) 2004, pp 331-345.

Lerner, J., and Tirole, J. "The Scope of Open Source Licensing," in: *working paper*, 2002a.

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

Long, J., and Yuan, M. "Are All Open Source Projects Created Equal? Understanding the Sustainability of Open Source Software Development Model," Americas Conference on Information Systems, Omaha, Nebraska, 2005, pp. 980-990.

Ma, M., and Agarwal, R. "Through a Glass Darkly: Information Technology Design, Identity Verification, and Knowledge Contribution in Online Communities," *Information Systems Research* (18:1) 2007, pp 42-67.

MacCormack, A., Rusnak, J., and Baldwin, C.Y. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Management Science* (52:7) 2006, pp 1015-1031.

Mackay, W.E. "Diversity in the Use of Electronic Mail," *ACM Transactionson Office Information Systems* (6:4) 1988, pp 380-397.

March, J. "Exploration and Exploitation in Organizational Learning," *Organization Science* (2:1) 1991, pp 77-87.

March, J.G., and Simon, H.A. *Organizations* John Wiley and Sons, New York, 1958.

Markus, M.L., Manville, B., and Agres, C.E. "What makes a virtual organization work?," *Sloan Management Review*) 2000, pp 13-26.

Mauri, R.A. "Unstoppable Linux: A Computer Platform for the Whole World," *Vital Speeches of the Day* (70), March 15, 2004 2004, pp 340-344.

Mockus, A., Fielding, R., and Herbsleb, J. "A Case Study of Open Source Software Development: The Apache Server," Proceedings of the 22nd International Conference on Software Engineering, ACM Press, Limerick, Ireland, 2000.

Mockus, A., Fielding, R.T., and Herbsleb, J.D. "Two Case Studies of Open Source Software Development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology* (11:3) 2002, pp 309-346.

Moon, J.Y., and Sproull, L. "Essence of Distributed Work: The Case of the Linux Kernel," *First Monday* (5:11) 2000.

Myatt, D., and Wallace, C. "Equilibrium Selection and Public Good Provision: The Development of Open Source Software " *Oxford Review of Economic Policy* (18:4) 2002, pp 446-461.

Norris, J., and Kamp, P.-H. "Mission-Critical Development with Open Source Software : Lessons Learned," *IEEE Software* (21:1) 2004, pp 42-49.

Oh, W., and Jeon, S. "Membership Dynamics and Network Stability in the Open-Source Community: The Ising Perspective " International Conference on Information Systems, Washington, D.C., 2004.

Orlikowski, W. "The Duality of Technology: Rethinking the Concept of Technology in Organizations," *Organization Science* (3:3) 1992, pp 398-424.

Ousterhout, J. "Freesoftware needs profit," *Communications of the ACM* (42:4) 1999, pp 44-45.

Pelled, L.H., Eisenhardt, K., and Xin, K. "Exploring the Black Box: An Analysis of Work Group Diversity, Conflict, and Performance," *Administrative Science Quarterly* (44) 1999, pp 1-28.

Perkins, G. "Culture clash and the road to world domination," *Software IEEE* (16:1) 1999, pp 80-84.

Poole, M.S., Holmes, M., and Desanctis, G. "Conflict Management in a Computer-Supported Meeting Environment," *Management Science* (37:8) 1991, pp 926-953.

Raymond, E. "The Cathedral and the Bazaar," 2000.

Reed, T.L. "Organizational Change in the American foreign service, 1925-1965 The utility of cohort analysis," *American Sociological Review* (43) 1978, pp 404-421.

Ren, Y., Carley, K., and Argote, L. "The Contingent Effects of Transactive Memory: When is it More Beneficial to Know What Others Know?," *Management Science* (52:5) 2006, pp 671-682.

Roberts, J.A., Hann, I.-H., and Slaughter, S.A. "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Management Science* (52:7) 2006, pp 984-1000.

Robillard, P. "The role of knowledge in software development," *Communications of the ACM* (42:1) 1999, pp 87-93.

Rogers, E.M. *Diffusion of Innovations* The Free Press, New York, 1995.

Rosenkopf, L., and Nerkar, A. "Beyond local search: Boundary-spanning, exploration, and impact in the optical disk industry," *Strategic Management Journal* (22:4) 2001, p 287.

Ryder, N. "The cohort as concept in the study of social change," *American Sociological Review* (30:6) 1965, pp 843-861.

Sagers, G. "The Influence of Network Governance Factors on Success in Open Source Software Development Projects," Twenty-Fifth International Conference on Information systems, Washington D.C., 2004.

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

Shah, S. "Motivation, Governance and the Viability of Hybrid Forms in Open Source Software Development," *Management Science*) 2006.

Staw, B. "The consequences of turnover; SUMMARY," *Journal of Occupational Behavior* (1:4) 1980, pp 253-274.

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

Stewart, K.J., Ammeter, A.P., and Maruping, L.M. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," *Information Systems Research* (17:2) 2006b, pp 126-145.

Stewart, K.J., and Ammeter, T. "An exploratory study of factors influencing the level of vitality and popularity of open source projects," 23rd International Conference on Information Systems, Barcelona, Spain, 2002, pp. 853-857.

Stewart, K.J., Darcy, D.P., and Daniel, S.L. "Opportunities and Challenges Applying Functional Data Analysis to the Study of Open Source Software Evolution," *Statistical Science* (21:2) 2006c, pp 167-178.

Stewart, K.J., and Gosain, S. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *Management Information Systems Quarterly* (30:2) 2006d, p 291.

Tornatzky, L., and Fleischer, M. *The Processes of Technological Innovation* Lexington Books, Lexington, 1990.

Von Hippel, E., and Von Krogh, G. "Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science," *Organization Science* (14:2) 2003, pp 209-223.

Von Hipple, E., and Von Krogh, G. "Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science," *Organization Science* (14:2) 2003, pp 209-223.

vonKrogh, G., Spaeth, S., and Lakhani, K. "Community, joining, and specialization in open source software innovation: A case study," *Research Policy* (32:7) 2003, p 1217.

Whang, S. "Market provision of custom software: Learning effects and low balling," *Management Science* (41) 1995, pp 1343-1352.

White, H. "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity," *Econometrica* (48) 1980, pp 817-838.

Ye, Y., and Kishida, K. "Toward an Understanding of the Motivations of Open Source Software Developers," Twenty-Fifth International Conference On Software Engieneering, Portland, Oregon, 2003.

Zahra, S., and George, G. "Absorptive Capacity: A Review, Reconceptualization, and Extension," *Academy of Management Review* (27:2) 2002, pp 185-203.

Zeitlyn, D. "Gift economies in the development of open source software: anthropological reflections," *Research Policy* (32:7) 2003, pp 1287-1291.

Zhao, L., and Deek, F. "User Collaboration in Open Source Software Development," *Electronic Markets* (14:2) 2003, pp 89-103.