MIS Quarterly

RESEARCH ARTICLE

NETWORK EFFECTS: THE INFLUENCE OF STRUCTURAL CAPITAL ON OPEN SOURCE PROJECT SUCCESS[1]

Param Vir Singh
David A. Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A. {psidhu@cmu.edu)

Yong Tan
Michael G. Foster School of Business, University of Washington, Seattle, WA 98195 U.S.A. {ytan@u.washington.edu}

Vijay Mookerjee
School of Management, University of Texas at Dallas, Dallas, TX 52242 U.S.A. {vijaym@utdallas.edu}

What determines the success of open source projects? In this study, we investigate the impact of ***network social capital*** *on open source project success. We define network social capital as the benefits open source developers secure from their membership in developer collaboration networks. We focus on one specific type of success as measured by the rate of knowledge creation in an open source project. Specific hypotheses are developed and tested using a longitudinal panel of 2,378 projects hosted at SourceForge. We find that network social capital is not equally accessible to or appropriated by all projects. Our main results are as follows. First, projects with greater* ***internal cohesion*** *(that is, cohesion among the project members) are more successful. Second,* ***external cohesion*** *(that is, cohesion among the external contacts of a project) has an inverse U-shaped relationship with the project's success; moderate levels of external cohesion are best for a project's success rather than very low or very high levels. Third, the* ***technological diversity*** *of the external network of a project also has the greatest benefit when it is neither too low nor too high. Fourth, the number of direct and indirect external contacts positively affects a project's success such that the effect of the number of direct contacts is moderated by the number of indirect contacts. These results are robust to several control variables and alternate model specifications. Several theoretical and managerial implications are provided.*

Keywords: Social networks, open source software development, cohesion, project success, team composition

Introduction

Recent years have witnessed the growing popularity of the open source approach to software development. Some well-known examples include Linux, Apache, Perl, R and MySql. A comparison of open source and closed source projects has revealed that open source has several advantages over the closed source model of software development. Open source software (OSS) is arguably more creative, better in quality, and cheaper to produce, as bugs are found and fixed more rapidly (Paulson et al. 2004). Proponents of open source have argued that it has the potential to replace closed source as the predominant mode of software development. Several researchers have argued that hybrid forms of development that borrow the most effective techniques from OSS and proprietary development may be appropriate (Mokus et al. 2002; Mokus and Herbsleb 2002). Major technological firms such as IBM, Unisys, and Novell, as well as large venture capitalists, are investing generously in open source communities (Shankland 2002). The surprising success of open

[1] Al Hevner was the accepting senior editor for this paper. Ravi Bapna served as the associate editor.

The appendices for this paper are located in the "Online Supplements" section of the *MIS Quarterly*'s website (http://www.misq.org).

source software has radically changed our conception of how innovation should be managed and prompted, which in turn has created a need for new models and theories to better understand innovation among distributed developer communities (von Hippel and von Krogh 2003).

The success of the open source approach to software development has contributed to an exponential growth in the number of such projects, which are hosted at sites such as SourceForge, Savannah, Tigris, and BerliOS. However, Chengalur-Smith and Sidorova (2003) note that with a few exceptions, the majority of the projects at open source hosting sites have been unsuccessful. This finding has generated immense interest among researchers in identifying the factors that affect the success of open source projects. Several software-specific characteristics have been attributed to an open source project's success, including the software type, intended audience, reputation of developers, licensing scope, and organizational sponsorship (Chengalur-Smith and Sidorova 2003; Lerner and Tirole 2005; Stewart et al. 2006). While these studies shed light on several important factors that affect the success of open source projects, a significant limitation of these studies is that they fail to consider the project production process and the broader social environment in which developers work. Thus, these studies implicitly rely on an atomistic and/or under-socialized view of open source projects according to which the production of the software is governed only by the characteristics of the software and is independent of the knowledge resources available to the project developers. Given the open source ideology of uninhibited sharing of knowledge and expertise and the underlying belief that such sharing increases the rate of knowledge creation, it is surprising that little research has focused on the impact of knowledge sharing across projects on the success of individual projects (exceptions being Grewal et al. 2006; Singh 2010). In this study, we control for software characteristics and emphasize the role of social networks in the focal project. We therefore believe that this study captures the underlying open source philosophy that promotes the sharing of knowledge resources across boundaries. In this study, we define project success in terms of the rate of knowledge creation, and we address the following questions: Does the structure of social networks related to an open source project influence its rate of knowledge creation? If so, what structural properties will enhance the rate of knowledge creation?

To resolve these questions, we investigate the impact of *network social capital* on open source project success. There is considerable ambiguity about the precise definition of social capital. We focus on network social capital defined by Portes (1998, p. 6) as "the ability of actors to secure benefits by virtue of their memberships in social networks or other social structures." The main focus of our study is to investigate how the network social capital available to a project affects that project's success. We find several interesting results. While projects with greater *internal cohesion* (that is, the cohesion among the project members) are found to be more successful, greater *external cohesion* (that is, cohesion among the external contacts of a project) does not always give rise to such benefits. Instead, moderate levels of external cohesion are best for a project's success. Similarly, the *technological diversity* of contacts also has the greatest benefit when it is neither too low nor too high. Finally, we find that the number of direct and indirect external contacts is positively correlated with a project's success such that the number of direct ties is moderated by the number of indirect ties. These results were obtained using a longitudinal panel of 2,378 projects hosted at SourceForge. These results are robust with respect to several model specifications and different sets of control variables. Moreover, the results are not just statistically meaningful but also economically significant.

By providing an understanding of the influence of network social capital on open source project success, this research makes several important contributions. First, we provide a more comprehensive examination of the influence of network social capital on project success than currently exists in the literature and provide recommendations that are relatively feasible for project managers to carry out. Grewal et al. (2006) measure social capital through network centrality and find that some measures of network centrality are better for project success (as measured by the number of CVS commits[2] and the number of software downloads), while other measures can detract from project success. Because centrality measures (specifically, betweenness and eigenvector centrality) for a given individual depend heavily on how other individuals are connected in the overall network, it may be unrealistic to assume that someone may be able to significantly influence this measure. Moreover, the centrality measures lose significance once we introduce our measures for network social capital. In addition, a study by Singh (2011) measures social capital through macro-level network properties. Singh investigates factors, specifically small-world properties such as the clustering coefficient, inverse of average path length, and their interactions, that make the entire open source community more or less conducive to successful software development as

[2]Software developers use a concurrent versioning system (CVS) to manage the software development process. CVS stores the current version(s) of a project and its history. A developer can access a complete copy of the code, work on this copy, and then check in his or her changes. The modifications are peer-reviewed. The CVS updates the peer-reviewed modified file automatically and registers it as a commit. CVS keeps track of changes, who made the changes, and when the changes were made.

measured by number of CVS commits. The implications of social network structure have not been considered with respect to the project level. Second, while most studies of network structure and project success (Grewal et al. 2006; Singh 2010), as well was project success in general (Chengalur-Smith and Sidorova 2003; Lerner and Tirole 2005), use cross-sectional data, we conduct our analysis using longitudinal data. By using longitudinal data, we are able to account for project-specific, unobserved heterogeneity; lagged relationships between project success and network measures; serial correlation and cross-sectional clustering, thus lending greater reliability to our findings. Third, while most studies on open source success have examined a small number of projects, our study uses several thousand projects, making our findings more robust and generalizable.

The rest of the paper is organized as follows. First, we discuss the formation of developer collaboration networks in the open source context. Hypotheses are then developed and data collection and measure construction are explained. Next, we discuss the model specifications, and provide the results. In the penultimate section, we discuss theoretical and practical implications, the limitations of our study, and future research directions. Conclusions are presented in the final section.

Open Source Developer Collaboration Networks

The production of open source software is varied, but it follows a basic pattern (Fogel 2005). A project originates when a developer registers at an open source website to share source code with others. Interested developers may access the code and start modifying it. Developers choose tasks voluntarily and usually collaborate as a team to incorporate their individual creations into a single, seamless body of source code. Once an executable version of the software is developed, it is released to users for testing and feedback. The software continues to evolve: new features are added, existing features are modified, and bugs are fixed. The entire process involves sharing ideas and joint problem solving that fosters strong social bonds among project members.

An open source developer may work on several projects at the same time and, hence, belong to multiple projects. The relationships between developers and projects can be represented by an *affiliation* network. An affiliation network is a special kind of two-mode social network that represents the affiliation between a set of actors and a set of social events (Wasserman and Faust 1994). Developers and projects represent the nodes of this network. Graphically, an affiliation network can be represented as a bipartite graph. Two developers are related if they work on the same project. Similarly, two projects are related if they share one or more developers.

Figure 1 illustrates an affiliation network. The top row of the figure represents projects, and the middle row represents developer cliques. The bottom row of the figure represents the global network that emerges from distinct developer projects. As more developers and projects are added to the community, the network evolves into a giant mesh of relationships.

Theory and Hypotheses

Open source Software Development

Nuvolari (2005) describes open source as a "collective invention" in which developers freely provide one another pertinent information to solve nontrivial technical problems. Prior research suggests that a significant amount of new software originates from designing improvements and recombining existing knowledge using novel approaches (Sacks 1994). Rather than designing from scratch, existing modular architectures are frequently inherited. For example, the GNU and FreeBSD projects closely resemble UNIX architecture (Narduzzo and Rossi 2003). Software often evolves from an initially integrated system by recombining and continually adapting existing modules.

In addition to the inheritance of code, developers also learn problem-solving skills that can be applied to other projects through their participation in software development (Boh et al. 2007). Although each software system is different and is customized to meet specific user expectations, the knowledge gained from one project can be used to improve the quality or reduce the effort needed to develop solutions for related projects (Basili and Caldiera 1995; Singh et al. 2010). It is clear from the above discussion that software development requires a considerable amount of abstract, theoretical, practical, and experiential knowledge (Sacks 1994). Such informal information is generally held by developers and projects teams (Singh 2010); access to such knowledge resources helps developers design better software, anticipate potential problems, quickly resolve thorny problems, and better incorporate user requirements during development.

While there is no universal formula for producing successful open source software, it has been hypothesized that innovative activities such as software development are affected by

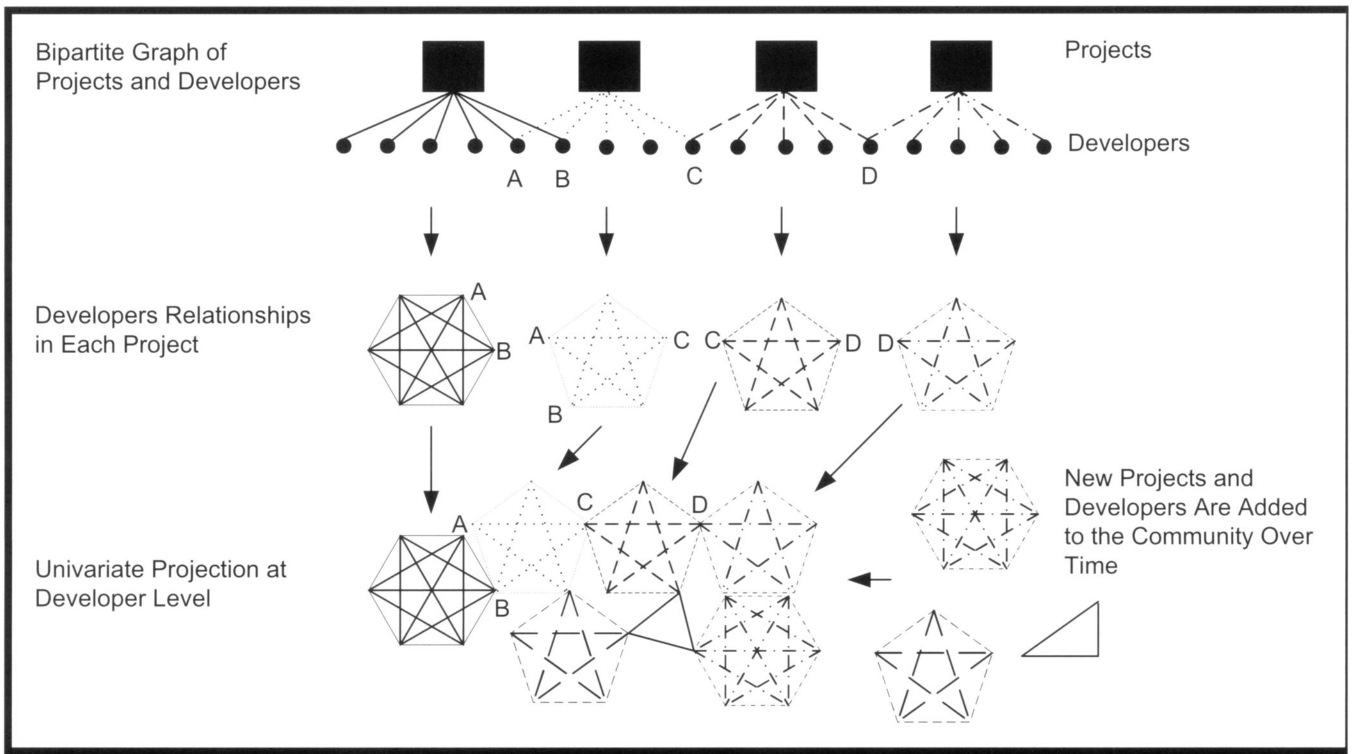
Bipartite Graph of Projects and Developers
Projects
Developers
A
B
C
D
Developers Relationships in Each Project
Univariate Projection at Developer Level
New Projects and Developers Are Added to the Community Over Time

Figure 1. An Open Source Developer Collaboration Network

the accessibility and diversity of knowledge resources available to a given project (Fleming 2001). The greater is the access to and diversity of these resources, the greater is the ability of project members to solve technical problems as well as create novel software that provides value to a user community. While a project has access to the source code of a large number of other open source projects, its relationships within the open source community help direct the focal project members in searching for available and relevant material. Given that the structure of a social network greatly influences the dynamics of information and knowledge seeking and diffusion within a network (Burt 2004; Granovetter 1973; Coleman 1988), the structure of collaborative relationships is likely to significantly influence the success of a project's knowledge-creation activities.

Internal Cohesion

Due to a lack of face-to-face interaction and an absence of formal management structures, communication in open source software projects is very difficult (Herbsleb and Mockus 2003; Herbsleb et al. 2006; Kuk 2006; Olson and Olson 2000; Singh and Tan 2005). Hence, the open source development process requires internal cohesion among project members (Stewart and Gosain 2006), which can promote knowledge sharing. Internal cohesion among developers has been found to increase with strong interpersonal connections, including those that occur through repeat collaborations (Szulanski 1996).

Prior research has shown that cohesion (that is, the existence of strong bonds) among project members gives rise to trust, reciprocity norms, and shared identity, all of which lead to a high level of cooperation and knowledge sharing (Coleman 1988). Internal cohesion within a project affects the willingness and motivation of individual developers to invest time and energy in sharing knowledge with others (Reagans and McEvily 2003; Szulanski 1996; Uzzi 1997). Cohesive project members are likely to develop a shared understanding of problems and solutions to greatly facilitate communication and learning (Brown and Duguid 1991). This increased ability enhances the rate of knowledge diffusion within a project, which greatly helps in knowledge creation.

However, as a project becomes very cohesive, its members start thinking similarly and develop a tendency to overlook alternate ways of solving problems (Burt 1992). Kuhn (1992) argues that very cohesive projects can, in fact, perform poorly in terms of knowledge-creating tasks because they overlook important information that does not match their current thinking and thus they overlook opportunities to innovate. Hence, we argue that a moderate level of internal cohesion among project members should be beneficial for project success, as this would enable richer and greater amounts of knowledge to be reliably exchanged without the drawbacks that are associated with extreme cohesion.

> **Hypothesis 1:** *A moderate level of internal cohesion within a project is better for a project's success than very high or very low levels of internal cohesion.*

External Resources

External resources are acquired from the relationships that project members have with developers outside of the focal project. The structure and type of these relationships affects the ability of outside developers to transfer various types of knowledge that potentially affect the success of the focal project. In the following, we focus on the external network structure and technological characteristics of external contacts that affect the diversity of external knowledge available to the focal project.

External Cohesion

As discussed earlier, individuals involved in an innovative task such as software development benefit through access to rich and reliable information. Cohesiveness in a project's external resources improves access to these resources; the same (or similar) knowledge is available via multiple paths. Another benefit of multiple paths is that developers can validate the knowledge received through different paths. However, too much cohesion within external resources has its drawbacks because information tends to be relatively redundant within a cohesive group (Burt 2004; Granovetter 1973). Cohesion also encourages adherence to established standards and conventions that could adversely affect experimentation or the use of novel approaches or perspectives (Uzzi and Spiro 2005). Social psychologists also suggest that members of a cohesive group tend to overlook important information that does not match with their current thinking (Kuhn 1970).

Large, distinct groups preserve the distinctness of knowledge within each group, and the actors that bridge these groups benefit by adopting best practices and relevant knowledge from each group (Singh 2010; Uzzi and Spiro 2007). Thus, a project should benefit most when its external contacts have few ties among them, enhancing information-transmission capacity and promoting trust and reciprocity norms, but not so much that they result in the homogenization of knowledge resources and perspectives.

> **Hypothesis 2:** *A moderate level of external cohesiveness is better for project success than very low or very high levels of external cohesiveness.*

Technological Diversity

It is not just how the external contacts are connected with each other that matters; the characteristics of the contacts may also affect the diversity of knowledge that can be potentially appropriated from them. Open source projects differ in terms of technological content. Contacts with different forms of technological expertise than that of the focal project are likely to provide new information and knowledge. Technological diversity between two projects provides developers opportunities to pool different kinds of knowledge, capabilities and perspectives. This sharing may foster creativity and novel solutions to existing problems (Sampson 2007). Because new knowledge results from recombining existing knowledge, access to technologically diverse knowledge may increase the number of possible reconfigurations.

Although it is clear that knowledge transfer across technologically diverse projects is beneficial for the recipient, transfer across technological boundaries can also be problematic. It is likely that the resources and capabilities of a developer are limited to the specific technological area of the project, limiting the utility and transfer of embedded capabilities and resources across projects. Prior research has shown that individuals learn new ideas by associating them with what they already know (Reagans and McEvily 2003). Hence, developers would find it easier to absorb relatively familiar knowledge but would find it more difficult to absorb unfamiliar ideas. Given that developers are more likely to assimilate external knowledge that is related to their existing knowledge, the technological similarity between two connected projects would ease knowledge sharing and transfer among them (Cohen and Levinthal 1990).

Based on these arguments, we hypothesize a nonlinear relationship between the technological diversity of a project's external network and project success. Thus, a project would benefit the most when external knowledge sources have diverse technological capabilities, thereby providing a large number of new perspectives and resources, but not so diverse as to prevent efficient assimilation.

> **Hypothesis 3:** *A moderate level of technological diversity of contacts is better for a project's success than very low or very high levels of technological diversity.*

Methods

To test our hypotheses, we collected data from SourceForge.net, which is the primary hosting site for open source projects, hosting more than 100,000 open source projects (van Antwerp and Madey 2008). Researchers interested in investigating issues related to open source have frequently used SourceForge (Crowston et al. 2003; Grewal et al. 2006; Singh 2010; Singh and Phelps 2009; Singh and Tan 2005; Xu et al. 2005). SourceForge organizes projects by technical foundries, which helps increase the visibility of a project within a foundry to developers of that foundry. SourceForge provides web space as well as services such as mailing facilities, discussion forums, CVS repository hosting, and download servers to open source projects to organize and coordinate their software development activities.

Data on projects and the developers associated with each project were obtained from the SourceForge database. We considered only those developers for a given project who are registered as project members at SourceForge. These data were supplemented with data on code development from the CVS log files from CVS project repositories.

To ensure that developers with names associated with projects in the SourceForge database have also actively participated in the projects during the period under consideration, we matched their contribution efforts and period of involvement using CVS log files, project communications, and project documentation, which typically lists all contributors. While the presence of a developer's name in the list of contributors provides conclusive evidence of that developer's involvement, only 38 percent of projects in our sample had project documentation. Developers also utilize various other forms of communication, such as Internet Relay Chat (IRC) and personal e-mail. Because of the ephemeral and/or private nature of these mediums, we were unable to collect and analyze these communications for our study. In addition to these ways of measuring project involvement, a developer may also provide code to a project manager to update the CVS commits. Thus, to indicate a developer's involvement in a project, we require that the developer has sent at least 10 messages in project e-mail archives or contributed at least 5 times to CVS commits in addition to formal membership in the project. These conditions ensure that the developers we consider to be part of a project are very likely involved in the project during the time period under study.

Network Construction

When using network data, one must define a network boundary that contains the collection of actors and relationships of interest. As recommended by Marsden (2005), we follow a *whole network* approach for network construction. This is the predominant approach used in situations in which an appropriate network boundary can be established (Laumann et al. 1983). We denote the Python foundry in our study as the network of interest; projects not in the foundry are considered outside the boundary. The choice of this network boundary is acceptable for two reasons. First, a foundry represents a focused software development platform; hence, more efficient knowledge-sharing is possible within a foundry rather than across foundries. Second, we analyzed memberships for 2,000 randomly selected developers who work on multiple projects and found that only approximately 4 percent of these developers worked across two or more foundries. The foundry has also been used as a network boundary for open source projects at SourceForge in related research (Grewal et al. 2006; Singh 2010; Singh and Phelps 2009).

Developer and project affiliation data needed to construct the network was collected for projects that were registered from November 1999 (that is, SourceForge's inception date) to November 2004 at SourceForge from the SourceForge database. The first snapshot of the SourceForge data was taken in January 2003, followed by a second snapshot in November 2004. We chose these two snapshots to construct the networks because these are the first two available snapshots for the Python foundry; in addition, a difference of approximately two years between the snapshots allows sufficient variation in network characteristics.[3]

A large proportion of projects hosted at SourceForge show no activity (Chengalur-Smith and Sidorova 2003). These projects are denoted as dead nodes in the network, and the relationships involving them do not facilitate any knowledge transfers or spillovers. Including such projects in the network may lead to a misleading indicator of network properties. Hence, we do not include them in network formation. We consider projects as *inactive* that show no activity between November 1999 and November 2006. All active projects within the Python foundry were selected, and the associated developers identified. Separate affiliation networks were constructed for each snapshot of the foundry. The final sample includes 1,472 projects and 3,328 unique developers in January 2003; the corresponding numbers grew to 2,378 projects and 5,601 unique developers in November 2004.

[3]Open source projects do not have a fixed deadline, and once registered for a project, developers rarely drop out. The variation in network properties is primarily due to the creation of new projects or the addition of new developers to existing projects.

Measures

Dependent Variable

Our theoretical arguments and subsequent hypotheses rely on a project's rate of knowledge creation as a measure of its success. Hence, our dependent variable should represent the amount of knowledge created by a project. Extant software development research suggests the use of modification requests (MRs) as a measure of rate of knowledge creation by a project that follows an incremental software development approach (Boh et al. 2007). The MR measure, which is similar in concept to a work order, represents the addition of new functionality as well as the modification or repair of old functionality. In open source, the CVS commit transaction measures a basic addition of functionality similar to that taken into account by the MR measure in a commercial development environment (Mockus et al. 2002). Hence, we use the number of CVS commits as a measure of rate of knowledge creation for a project. This is also one of the success measures corresponding to the level of activity in a project, as suggested by Crowston et al. (2003). This measure of project success is consistent with the literature on information system success (DeLone and McLean 1992). In addition, the two studies most closely related to our work, namely, Grewal et al. (2006) and Singh (2011), have also used CVS commits as a measure of project success. Using this measure allows us to compare our results with earlier studies.

Benefits from a relationship are evident only after the relationship has been established. This calls for the use of a lag between network variables and the success measure (Gulati and Gargiulo 1999). Hence, we measure the dependent variable $\textit{CVS Commits}_{it+j}$ as the number of CVS commits for project i in j years subsequent to network construction date t. For instance, for the network constructed in November 2004 and $j = 1$, we count the number of CVS commits for a project from November 2004 to November 2005.

Focal Independent Variables[4]

Internal Cohesion: We used *repeat ties* as a measure of a project's internal cohesion. The construct for strong interpersonal connections indicates the presence of repeat collaborations among project members (Uzzi 1997). We counted the total number of projects (both past and present projects) on which each pair of project members works or has worked. We divided this number by the total number of pairs that exist in a project to compute a measure of *repeat ties* for the project. A high score of repeat ties indicates that the project is composed of developers who have worked or are working together on several projects. To capture the curvilinear relationship that we proposed in Hypothesis 1, we also square repeat ties and use this term as an independent variable.

External Cohesion: Our measure of external cohesion for a given developer is Burt's (1992) network constraint. This measures the extent to which a project member's external alters (that is, contacts) share relationships with each other. Higher values of this constraint for a project imply that its external alters are more connected, thereby indicating greater external cohesion. The square of external cohesion is also included as an independent variable to capture the curvilinear relationship proposed in Hypothesis 2.

Technological Diversity: We first define the technological position of each project. Extant software engineering research suggests that the technological position of a software project can be defined in terms of the type of project (such as gaming or Internet applications), programming language, user interface, and operating system (Jones 1984; Sacks 1994). Each of these dimensions represents different types of technical expertise. Project type represents the application domain knowledge, whereas the other three represent the tools and expertise that comprise knowledge of the process, data, and functional architecture (Kim and Stohr 1998). Software engineering research has shown that the similarity of domain and tools affect the amount of knowledge that can be reused from one project to another (Banker and Kaufman 1991; Lee and Litecky 1999). Following Jaffe (1986), the technological diversity between two projects p and q is calculated by the angular separation or uncentered correlation of their technological positions. Technological diversity varies from zero to one, with a value of one indicating the greatest possible technological diversity between two projects. We calculate the technological diversities of a focal project with all of the projects with which it shares a developer. We sum these measures and divide it by the number of such projects to calculate the technological diversity measure for the focal project. We also include a square of technological diversity as an independent variable to test the curvilinear relationship hypothesized in Hypothesis 3.

Control Variables

Direct Ties: For each project member, we counted the number of ties that the project member has with developers other than focal project members. We take an average of this number over all project members to compute a measure of direct ties for the project. This variable controls for the ability

[4]Details on constructing the key independent variables are presented in the online supplement to this paper.

of the project to acquire tacit knowledge from external sources.

Indirect Ties: For each project member, we could count the number of developers with whom the member does not have a direct tie but can reach through others. However, this does not take into account the weakening of tie strength as the distance between two developers increases. Hence, we used a frequency decay measure for indirect ties as proposed by Burt (1992), which accounts for such decay. We divided this measure by the number of project members to calculate a measure of indirect ties for a project. This measures controls for the capacity of a project to acquire explicit knowledge from outside.

Project Human Capital and Ability: We included the number of developers (i.e., project size) associated with a project to account for human capital actively involved in a project. To gauge the prior capacities of the project, we calculated pre-sample CVS commits by measuring their cumulative value for each project up to the network construction date. This specification follows the pre-sample information approach in order to control for different knowledge stocks across projects at the time of production and other historical factors that cause subsequent differences in the dependent variable that are difficult to control.

User Input and Market Potential: Although all projects belong to the same foundry, the software produced by the project may differ in terms of market potential and extent of user participation. Users play a critical role in the evolution of an open source product (von Hippel and von Krogh 2003). Activities such as bug reports, bug fixes, and user support are user-driven activities. We control for these activities by constructing two variables: *support* is the cumulative number of support requests answered, and *bugs* is the cumulative number of bugs closed. We follow Grewal et al. and construct a variable *page views*, which is the cumulative number of project pages viewed to control for market potential and general interest in the project.

Project Age: We control for the age of the project by calculating a *project age* variable as the number of months since a project's inception at SourceForge according to the network construction date. To control for a potential nonlinear effect of project age on the dependent variables, we also incorporate a square of *project age* into the model. This variable accounts for the potential complexity associated with software as the code becomes larger.

Project Characteristics: Following extant research that attributes project success to software characteristics, we constructed an extensive range of variables to account for software characteristics. We controlled for a project's type, organizational sponsorship, language, intended audience, license type, and user interface by constructing dummy variables. To account for complexity related to the stage of software development, we calculate the fraction of all new commits that are modifications to existing files. This captures the extent to which new functions have been added. Values close to one for this variable indicate that the software is in a maintenance phase.

Model Specification

Initial investigations revealed that the dependent variable and some of the independent variables were not normally distributed. In such a case, linear regression analysis might yield biased parameter estimates that cannot be easily interpreted (Gelman and Hill 2007). Therefore, as suggested by Gelman and Hill (2007), we performed a logarithmic transformation on the dependent, non-normally distributed independent variables. Investigations also showed that the squares of the relevant variables were highly correlated with them. The correlations were high enough to raise concerns regarding multicolinearity, which, if uncorrected, may lead to inflated standard errors and even inconsistent or unstable estimates (Greene 2003). As suggested by Gelman and Hill, we mean-centered these variables before taking their squares, which reduced the correlation to acceptable levels.[5] To avoid any confusion, we denote the log-transformed variables by adding "ln" to the variable name. We also denote the mean-centered variables by including "*mc*" in the variable name.

Even though we incorporate several control variables to account for the inherent differences among different projects, unobserved heterogeneities may still exist among projects. For example, one project may employ the bug reporting tool provided by SourceForge differently than other projects. Unobserved attributes such as motivation, the quality of involved developers, and the amount of complexity involved in a code may potentially influence project success. To account for this unobserved heterogeneity, we model it as a hierarchical Bayesian project random effects framework (Allenby and Ginter 1995). This implies that we incorporate heterogeneity in the effects of different covariates on project success by allowing the corresponding coefficients to vary across projects. The model specification is as follows:

[5]The descriptive statistics of the variables and the correlation matrix of the transformed variables are provided in the supplement to this paper.

$$\begin{aligned}\ln DV_{it} = {} & \beta_{0i} + \beta_{1i} mcRT_{it} + \beta_{2i}(mcRT_{it})^2 + \beta_{3i} mcEDT_{it} + \beta_{4i} mcEIT_{it} + \\ & \beta_{5i} mcEDT_{ut} \times mcEIT_{it} + \beta_{6i} mcEC_{it} + \beta_{7i}(mcEC_{it})^2 + \beta_{8i} mcTD_{it} \\ & + \beta_{9i}(mcTD_{it})^2 + \beta_{10i} \ln TSIZE_{it} + \beta_{11i} \ln PDV_{it} + \beta_{12i} \ln BUGS_{it} \\ & + \beta_{13i} \ln SUP_{it} + \beta_{14i} \ln Page_{it} + \beta_{15i} mcAGE_{it} + \beta_{16i}(mcAGE_{it})^2 \\ & + \beta_{17i} T + \beta_{18i} FR_{it} + \beta_{19i} EN_i + \beta_{20i} SP_i + \\ & \sum_{m=1}^{ui-1} \delta_{mi} UI_{un} + \sum_{k=1}^{ia-1} \lambda_{ki} IA_{ik} + \sum_{h=1}^{ty-1} \eta_{hi} TY_{ih} + \sum_{g=1}^{os-1} \gamma_{gi} OS_{it} + \sum_{r=1}^{lt-1} \tau_{ri} LT_{ur} + \varepsilon, \end{aligned}$$

$$\theta_i = v' Z_i + \varepsilon_{\theta i} \tag{1}$$

where i is the project; t is the network year; $\theta_i = \{\beta_0, \beta_1, \ldots, \beta_{20}, \delta_1, \ldots, \delta_{ui-1}, \lambda_1, \ldots, \lambda_{ia-1}, \eta_1, \ldots, \eta_{ty-1}, \gamma_1, \ldots, \gamma_{os-1}, \tau_1, \ldots, \tau_{lt-1}\}$ is a set of random effects parameters, Z_i is a vector of ones; v is the matrix of parameters, which also represent the mean effect size; $\varepsilon_{\theta i} \sim N(0, \Sigma_{\theta i})$; ln*DV* is the log of the dependent variable *CVS Commits*; *mcRT* is the mean-centered repeat ties for the project; *mcEDT* is the mean-centered external direct ties of the project; *mcEIT* is the mean-centered indirect ties of the project; *mcEC* is the mean-centered external cohesion of the project; *mcTD* is the mean-centered technological diversity for the project's alters; ln*TSIZE* is the log of the number of developers in the project; ln*PDV* is the log of pre-sample dependent variables; ln*BUGS* is the log of the number of bugs closed; ln*SUP* is the log of the number of support requests answered; *EN* is an indicator variable that equals one if the network year is January 2003 and zero otherwise; *FR* is the fraction of CVS commits for the project that represents changes to existing files; and *SP* is the indicator variable that equals one if the project received organizational sponsorship and zero otherwise. *OS* represents a set of dummy variables, one for each operating system except the reference category, which is denoted as *os*. Note that *os* is the total number of categories of the operating system. *UI* represents a set of dummy variables, one for each project user interface except the reference category, which is denoted as *ui*. Note that *ui* is the total number of categories for the project user interface. *IA* represents a set of dummy variables, one for each intended audience of the project except the reference category, which is denoted as *ia*. Note that *ia* is the total number of categories for the project's intended audience. *TY* represents a set of dummy variables, one for each project type except the reference category, which is *ty*. Note that *ty* is the total number of categories for project type. *LT* represents a set of dummy variables, each of which indicates the presence of viral and/or copyleft clauses in the software license. ε_{it} is the panel error term.

A few specification issues should be discussed here. The first issue involves reverse causality. Reverse causality relates to whether a project's performance shapes the network, rather than the reverse. In our case, this concern is negligible because our network constructs are measured prior to measuring the dependent variable. This lag structure also reduces concerns regarding simultaneity. In addition, as suggested by Mouw (2006), we control for project-specific unobserved effects as well as a pre-sample dependent variable, which further minimizes simultaneity concerns. A second concern relates to the potential of serial correlation. The inclusion of a pre-sample dependent variable makes the model state dependent (Heckman 1991). We also analyze the residuals for serial correlation and find this is not a problem.

Results

Figure 2 shows the developer collaboration network size and component structure for January 2003 and November 2004. In both years, there is one large component and several smaller components of varying sizes. Had we followed a snowball approach by starting with a few projects and collecting information on other projects with which they share developers, and repeating the process several times, we would have potentially missed the majority of isolates and smaller components. The size of the large component increases from 966 developers and 317 projects in January, 2003, to 2,001 developers and 519 projects in November, 2004. This increase in size is the result of several smaller components merging with the large component and the addition of new developers and projects over time. Overall, the networks in the two periods are relatively sparse. The percentage of isolates (that is, projects for which project members do not have any external ties) is relatively stable over time at 35.67 percent in January, 2003, and 36.54 percent in November, 2004. Similarly, the density of the overall network (or proportion of potential network ties that are actually realized) is relatively stable over time at 0.29 percent in January, 2003, and 0.32 percent in November, 2004. Network centralization is the degree to which a few developers dominate the network; a value of zero implies that all developers have the same number of ties. Network centralization is 4.82 percent in January, 2003, and 2.30 percent in November, 2004. This indicates a moderate level of network centralization, implying that although most developers are involved in network formation, some are more active than others. In January, 2003, a total of 81.2 percent of the developers worked on only one project, 11.7 percent worked on two projects, and the remaining worked on more than two projects. In November, 2004, a total of 79.6 percent of the developers worked on only one project, 13.2 percent worked on two projects, and the remaining worked on more than two projects. In January, 2003, the average project size was 2.8, and its standard deviation was 4.1. The average project size for November, 2004, was 2.9, and its standard deviation was 6.4.

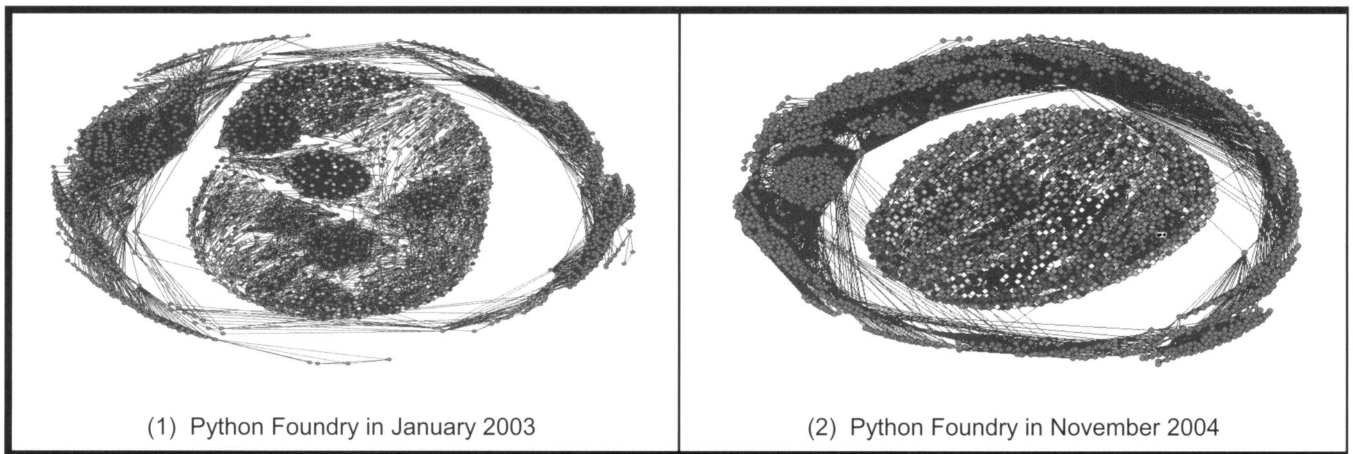
(1) Python Foundry in January 2003
(2) Python Foundry in November 2004

Figure 2. Developer Collaboration Network Size and Component Structure (Developers are color coded by component. A color version of the figure is included with the online supplement. Isolates are not shown.)

In Table 1, we report the results of the regression analyses using hierarchical Bayesian random-effects estimation. The dependent variable leads the independent variables by one year. Model 1 presents the base model with only the control variables. Model 2 adds internal cohesion to Model 1. Model 3 adds external cohesion to Model 2, and Model 4 adds technical diversity to Model 3. Parameter means are reported in the table; standard errors of the parameter means are reported below them. The standard deviation of the parameters is reported next to the parameter means in brackets. Time period and project characteristic effects, while estimated, are not reported to conserve space. We refer to the full specification (that is, Model 4) to discuss the results.

The results indicate that there is substantial unobserved heterogeneity among projects. When compared to the parameter means, the standard deviation of parameters is considerable and significant. For instance, for bug reporting, the one standard deviation interval around the parameter mean is 0.087 to 0.307. This would imply that the success of some projects is impacted approximately three times as much by bug reporting activity than other projects. This highlights the inherent difference among projects with respect to the use of these tools. In other words, some projects may use them more appropriately than others.

Focal Independent Variables

Our first hypothesis states that internal cohesion has a curvilinear effect on project success. The results do not support this hypothesis. While the coefficient corresponding to internal cohesion is positive and significant (1.693, $p < 0.001$), the coefficient of its square is negative and not significantly different from zero (–0.428, $p > 0.1$). A possible explanation for the partial support of this hypothesis is that the internal cohesion in the dataset is not high. The internal cohesion is possibly less than the inflection point value at which it starts to have a negative impact on project success.

Consistent with Hypothesis 2, we find that external cohesion has a curvilinear effect on project success. The coefficient for external cohesion is positive and significant (4.569, $p < 0.001$), whereas the coefficient for its square is negative and significant (–4.998, $p < 0.001$). Figure 3 plots the mean CVS commits versus external cohesion for projects with 1,000 and 5,000 CVS commits at the mean level of external cohesion of 0.18. As argued above, as external cohesion increases from its minimum value, it has positive effect on CVS commits but begins to have a negative effect on mean CVS commits as it increases beyond the limit of 0.65. The coefficient for technical diversity is positive and significant (1.578, $p < 0.001$), whereas the coefficient for its square is negative and significant (–6.124, $p < 0.001$). This provides support for Hypothesis 3. The effect of technical diversity on mean CVS commits for projects with 1,000 and 5,000 CVS commits at the mean level of technological diversity of 0.48 is plotted in Figure 4. An increase in technical diversity has a positive effect on mean CVS commits when it is below the limit of 0.6 but has a negative effect as it moves beyond this limit. A comparison of Figures 3 and 4 also indicates that the magnitude of effect of technical diversity on external contacts is significantly smaller than the effect due to cohesive external contacts.

Table 1. Results of the Hierarchical Bayesian Random Effects Models (*n* = 2,378; *Obs* = 3,850)

	1		2		3		4	
Independent Variables								
Variable Name								
Internal Control								
Repeat Ties			2.324***	[0.49]	1.465***	[0.71]	1.693***	[0.65]
Repeat Ties Squared			-0.652	[0.39]	-0.419	[0.41]	-0.428	[0.39]
External Network								
Ext Cohesion					8.426***	[1.36]	4.569***	[0.98]
Ext Cohesion Squared					-9.745***	[2.17]	-4.998***	[1.51]
Tech Diversity							1.578***	[0.39]
Tech Diversity Squared							-6.124***	[1.78]
Indirect Ties	2.249***	[0.69]	2.249***	[0.69]	2.249***	[0.69]	2.246***	[0.69]
Direct Ties	0.584*	[0.30]	0.585*	[0.30]	0.584*	[0.30]	0.581*	[0.30]
Direct Ties × Indirect Ties	-3.457***	[0.44]	-3.457***	[0.44]	-3.457***	[0.44]	-3.461***	[0.44]
Market Potential and User Interest								
Page Views	0.020	[0.13]	0.026	[0.13]	0.010	[0.13]	0.002	[0.13]
Support	0.284*	[0.20]	0.244*	[0.20]	0.246*	[0.20]	0.224*	[0.20]
Bugs	0.148***	[0.11]	0.166***	[0.11]	0.177***	[0.11]	0.197***	[0.11]
Project Live Cycle Effects								
Project Age	-0.210	[0.09]	-0.107	[0.09]	-0.668*	[0.09]	-1.191***	[0.09]
Project Age Squared	5.466***	[2.17]	5.903***	[2.17]	5.014***	[2.17]	4.729***	[2.17]

The posterior variance of the coefficient is given in brackets. ***Indicates a 99% confidence interval not including zero. **Indicates a 95% confidence interval not including zero. *Indicates a 90% confidence interval not including zero.

All models include a time dummy, project characteristic controls, and a constant. Project characteristics controls include software type (i.e., communication, database, desktop utility, education, games, Internet, multimedia, office, printing, scientific system, security, software development, system, terminal, text editor, and other), intended audience (i.e., end users, system administrators, and developers), user interface (i.e., Microsoft, KDE, web-based, non-interactive, curses-ncurses, gnome, cocoa, handheld, and other), operating systems (i.e., POSIX, BSD, Sun Solaris, SGI/RIX, Windows, IBM OS/2, Palm OS, Apple Mac, Embedded, OS independent, OS kernel, and other), language (i.e., English and other), sponsorship (i.e., yes and no), and license (i.e., viral, copyleft, and other). For all of these project characteristics, the category "other" was the reference category. Maintenance phase was indicated by the fraction of all CVS commits that were modifications to existing files. Indirect ties, direct ties, and project age are scaled down by a factor of 1000, 100, and 100, respectively.

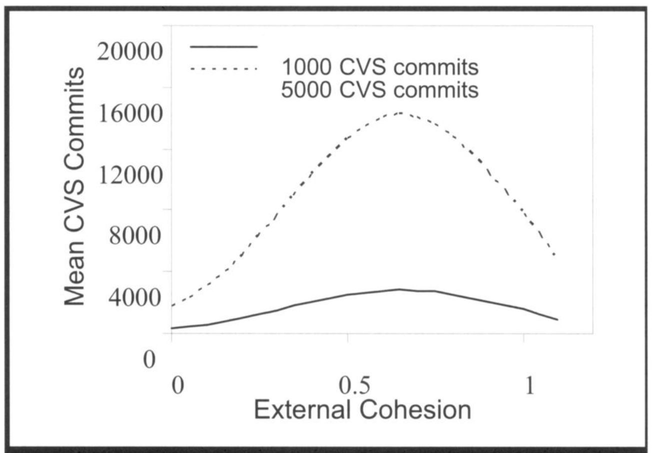

Figure 3. CVS Commits Versus External Cohesion

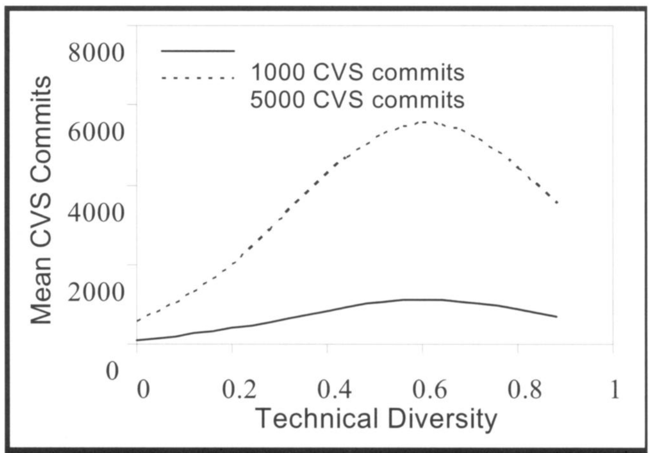

Figure 4. CVS Commits Versus Technical Diversity

Controls

Some of the controls merit further discussion. We find that the coefficient for direct ties is positive and significant (0.581, $p < 0.05$). The coefficient for indirect ties is also positive and significant (2.246, $p < 0.001$). The coefficient for the interaction between direct and indirect ties is negative and significant (–3.461, $p < 0.001$). Note that the coefficient for direct ties is insignificant. This is due to the different interpretation of direct ties with and without the interaction term. Because the interaction term is significant, direct and indirect ties represent simple effects rather than true main effects in the presence of the interaction term (Jaccard and Turrisi 2003). To understand the true effect of direct and indirect ties, it is important to consider their structure and that of their interaction. Both direct and indirect ties are mean-centered. The effect of each on CVS commits is conditioned on the other taking a value of zero (i.e., the mean value). For example, 0.581 is the effect of a unit change in direct ties on CVS commits when the value of indirect ties is at its mean value. Similarly, 2.246 is the effect of a unit change in indirect ties on CVS commits when the value of direct ties is at its mean value. The total effect of a unit change in direct ties when the value indirect ties is not at its mean value is 0.581 – 3.461 × indirect ties. The total effect of unit change in indirect ties when the value of direct ties is not at its mean value is 2.246 – 3.461 × direct ties. The range of the total effect of a unit change in direct ties is 0.581 ± 3.461 × 0.162, where 0.162 is the standard deviation of indirect ties. Similarly, the range of the total effect of the unit change in indirect ties is 2.246 ± 3.461 × 0.228, where 0.228 is the standard deviation of direct ties. Note also that the magnitude of the effect of direct ties is significantly lower than the magnitude of the effect of indirect ties.

Regarding market potential and user interest variables, bugs and support affect CVS commits positively. Project size has a positive and significant effect on CVS commits. The pre-sample CVS commits variable is positive and significant, indicating its importance as a control for project-level unobserved heterogeneity.

Robustness Checks

We performed several robustness checks on our results. Note that the means of establishing membership in a project process explained in the "Methods" section are not perfect. For example, we may have omitted a developer who is involved in the project but exclusively participates on IRC chat channels and always forwards her work to the project manager to upload CVS commits. To test the robustness of our results with respect to potential omissions of involved developers, we reconstructed the entire sample by including all developers that appear on the developer list of a project on SourceForge. All measures were calculated for this new sample, and the analysis was performed again to check for robustness. The results from the new sample are qualitatively consistent with those reported in Table 1.

The network measures are meaningful only for projects with at least one project member working on more than one project. The preponderance of zero values for the network measures due to isolates increases the correlation among network measures. Although variance inflation factor (VIF) indicates that multicolinearity is not a problem, we performed some additional tests to check for the robustness of results. We ran the analysis again, using only those projects that had at least one project member working on more than one project. The selected sample had significantly lower correlation among network measures and the results from this sample provide qualitatively consistent results to those shown in Table 1. We also estimated the models using subsamples after randomly omitting one-third of the observations. The results indicate that the reported estimates are stable. To test for the appropriateness of lag length between independent and dependent variables, we ran multiple lag specifications ranging from 9 months to 18 months. The results were qualitatively consistent with those shown in Table 1. As we mentioned earlier, our dataset is unbalanced, as there are 906 new projects in the November 2004 snapshot. If this unbalances the panel structure, the estimated coefficients may be inconsistent. To test for the consistency of the estimators, we followed Nijman and Verbeek (1992) and performed a Hausman test to compare the random effects of regression estimates on both a balanced and an unbalanced panel. The Hausman test for selectivity bias was insignificant. In this scenario, the results from the unbalanced panel are consistent and efficient. When we incorporate Grewal et al.'s (2006) measures of betweenness and eigenvector centralities, our results do not change, although these two centrality measures turn out to be insignificant predictors of project success. Note that their third measure degree centrality is very similar to our direct ties measure. Similarly, our results are also robust to the inclusion of Singh's (2011) measures of small world properties in the estimation. Some of the independent variables are still skewed even after log transforming them. To test if the skewness in the variables is affecting the results, we performed a median regression. The results from the median regression are qualitatively similar to those presented in Table 1. Finally, we have assumed that the network structure is exogenous. It is possible that it may be endogenous. Better developers may self-select themselves to work on projects with a greater chance of success because they know who to work with, as they have worked with them in the past. Hahn et al. (2007) showed that prior success on a project does not affect the likelihood of two developers working together on a future project on SourceForge. We have also independently tested this relationship in our data and found a similar effect. Further, the inclusion of past CVS commits should account for the effect of past project success on network structure.

Discussion

Theoretical and Practical Implications

In this study, we investigated how a project's structural social capital affects its success. We developed theoretically driven hypotheses on how different structural properties affect project success. We tested these hypotheses using longitudinal data from a large sample of 2,378 projects hosted at SourceForge. The results indicated support for all hypotheses. The results were robust to a number of controls and model specifications.

This work has several theoretical and practical implications. Theoretically, we argued that a moderate amount of internal cohesion among project members would contribute more to project success than extreme (that is, either low or high) amounts. However, we found that internal cohesion has a linear relationship with project success. As discussed earlier, this may be a result of low levels of internal cohesion among the project members in our dataset. However, this finding is consistent with the results obtained in related research (Singh 2010;Stewart and Gosain 2006). However, our measure for internal cohesion differs from these previous studies, as we focused on relationships among project members that exist beyond the focal project that may affect cohesion within the focal project. Although Singh's (2011) measure of cohesion is based on external relationships among project members, it is a macro-level measure of cohesion based on overall clustering across an entire community of open source developers. This macro-level measure assumes that cohesion is similar across all projects in a network, whereas our measure allows for internal cohesion to differ across projects within a network. Stewart and Gosain (2006) investigate the issue of project effectiveness in open source from a psychological perspective and find that cohesion leads to shared norms, beliefs, and values among project members, all of which positively influence project effectiveness. Our results also support their findings, albeit through a social structural analysis. Furthermore, Stewart and Gosain measure project success by the number of developers on the project and the number of work weeks devoted to the project, whereas our measure of success is the rate of knowledge creation by the project. Together, these findings suggest that internal cohesion has a multidimensional impact on different aspects of the success of open source projects.

Prior research on project diversity has only investigated the role of global structural holes on its performance in a linear relationship (Reagans and Zuckerman 2001; Reagans et al. 2004). However, we found that the absence of structural holes (that is, the presence of cohesion) among the external alters of a project has an inverse U-shaped relationship with project success. We find that a moderate amount of external cohesion facilitates both the access to and diversity of knowledge resources available to the project. Xiao and Tsui (2007) point out that the ability of structural holes to generate social capital is affected by the cultural context of the community. Future research should investigate whether the nonlinear relationship between external cohesion and project success identified in this study is a result of the cultural context of this study or a limitation of prior research that does not incorporate the possibility of a nonlinear effect of external cohesion.

Finally, existing research that investigates the impact of diversity on knowledge creation generally measures diversity by the extent of structural holes (Grewal et al. 2005; Reagans and Zuckerman 2001; Reagans et al. 2004; Singh 2010; Uzzi and Spiro 2005) present in an individual's network (an exception being Sampson 2007). Our results indicate that an individual can also obtain access to diverse information if its external alters are technologically diverse. We found that knowledge spillover effects are more pronounced for projects for which external alters are moderately technologically diverse.

Overall, our results have much to offer the knowledge spillover literature on open innovation communities. We find that knowledge spillovers are not equally accessible to or appropriated by everyone. Direct access through direct ties and indirect access through indirect ties affect a project's ability to access these external knowledge resources. Through direct ties, a project member can pool resources from only the projects he/she has worked on in the past or is working on at present. It is the indirect ties that expand a project's search space and provide it with access to knowledge from a larger base.

Our results highlight several insights for open source developers and project managers in established software development firms that are participating in open source processes. Team composition is often a central concern for open source project managers. Because open source communities grow at an exponential rate, it is increasingly more difficult for emergent projects to receive the attention of the open source community. This makes it more difficult for managers of new projects to attract developers. Given the importance of project composition to the eventual success of a project, as illustrated by our study, open source hosting sites should try to develop a recommendation system to provide an appropriate pool of developers to managers of new projects. We suggest that projects should be composed of developers who not only share cohesive ties but also have reliable access to diverse external knowledge resources. Hence, managers should try to recruit developers who have either successfully worked together in the past or worked separately with a common third party. Furthermore, these developers should form moderately cohesive external groups. Thus, the managerial focus in the context of open source software development should be at identifying, recruiting, and retaining such developers.

The frequency-decayed indirect ties are highly correlated with a simple count measure of indirect ties, which is also a measure of component size. This finding also suggests that the network component size also affects the success of embedded projects. As illustrated by Figure 2, the network is composed of multiple components, and a significant proportion of projects are isolates. As a community, open source participants should try to devise strategies to incorporate these isolates as well as smaller components into the major component. It is unlikely that open source participants have a complete understanding of their entire network. Open source hosting sites could potentially benefit these participants by providing them a graphical network representation of the relationships in their networks. This would help developers that belong to smaller components or isolates to tap into the resources of a larger component by joining it.

Open source developers who work on multiple projects at a given time should choose to work on projects that are moderately technologically diverse from one another. Such technological diversity would enhance their performance on each project. In established firms that promote their employees to work on open source projects, managers should encourage employees to work on projects that are moderately technologically diverse and/or help developers tap into distinct bodies of knowledge.

Limitations and Future Research Directions

We do not wish to overstate our results, as this study has a number of limitations. We measure the effect of network structure on project success. The causal mechanism we assumed is that network structure affects knowledge transfer. However, we did not observe knowledge transfer directly but rather infer it from the relationship between network structure and project success. Future research should investigate the exact mechanism by which such knowledge transfer occurs through field studies or controlled experiments. Knowledge may flow to projects through other mechanisms. Collaborations in the open source community represent only one form of a relationship. Because open source developers are usually employees of commercial software development firms, a relationship among them may exist by joint participation in projects at the same firm. Moreover, a developer may learn about knowledge embodied in an unconnected project simply by using the software it produces or by analyzing its source code.

We have focused on only one aspect of project success as measured by the rate of knowledge creation by an open source project. Future research can investigate the impact of structural social capital on other aspects of open source project success, such as user acceptance. Future research should also investigate the issue of network endogeneity more rigorously by modeling the coevolution of network formation and project success similar to the work of Burk et al. (2007).

The data for this study followed the network evolution over the course of only a few years. This leads to a situation in which a few developers work on more than one project. As a result, internal cohesion was not high enough to provide a robust test for its curvilinear relationship with project success. Future research should consider collecting data that document network evolution over several years. Furthermore, a clique-based network construction may not be a perfect representation of social ties among open source developers. We may be counting ties as present when they actually do not exist. Understood as an errors in variable problem, this limitation suggests that we are underestimating the true effects of network variables in our estimation (Wooldridge 2007).

We cannot account for an individual who is not a formal member of the project but contributes patches. Future research can devise more sophisticated methods to collect social network data on open source developers using surveys or field studies to address this shortcoming. It may also be interesting to study whether an imbalance in social capital rather than the average social capital among project members has any impact on project success.

Finally, the extent to which the insights obtained in our study are applicable to closed-source environments is an open question. Even in closed source environments, knowledge sharing is allowed within a firm. Our results should encourage researchers as well as managers of such firms to explore whether project performance is enhanced by our suggestions on team composition even within closed source contexts.

Conclusions

This study makes several contributions to existing and emerging literature in several areas of study. First, we add to the emerging literature that investigates open source project success (Chengalur-Smith and Sidorova 2003; Grewal et al. 2006; Lerner and Tirole 2005; Singh 2010; Stewart et al. 2006) by analyzing how structural social capital influences the success of embedded projects. We have shown that developer collaboration networks may be an important mechanism in knowledge spillovers and that the specific structure of a project's relationships can have an important impact on its success. Second, we also add to the open source research that investigates these issues from a social network perspective (Fleming and Waguespack 2007; Grewal et al. 2006; Hahn et al. 2006; Singh 2010; Xu et al. 2005) by showing how relationships beyond the focal project may affect cohesion within the focal project. We have shown that those relationships are beneficial for projects that result in within- project high cohesion but moderate cohesion across a project's alters. We have also shown that the technological characteristics of alters may affect the amount of knowledge that can be transferred. Finally, we add to the social network and project composition literature that investigates the influence of the extent of network closure and the presence of structure holes in an ego's network on the ego's performance (Reagans and Zuckerman 2001; Reagans et al. 2004). We find that both network closure and structural holes are important for knowledge transfer in an open source environment. Our results suggest that a moderate amount of network closure and structure holes in the external network of a project benefits it the most. Several interesting theoretical and practical implications for the open source community as well as firms investing in open source software development are also discussed. We note the limitations of this study, which open up exciting avenues for future research.

References

Allenby, G., and Ginter, J. 1995. "Using Extremes to Design Products and Segment Markets," *Journal of Marketing Research* (32:4), pp. 392-403.

Banker, R., and Kauffman, R. 1991. "Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study," *MIS Quarterly* (15:3), pp. 375-401.

Basili, V., and Caldiera, G. 1995. "Improve Software Quality by Reusing Knowledge and Experience," *Sloan Management Review* (37:1), pp 55-64.

Brown, J. S., and Duguid, P. 1991. "Organizational Learning and Communities of Practice: Towards a Unifying View of Working, Learning and Innovation," *Organization Science* (2:1), pp. 40-57.

Boh, W., Slaughter, S., and Espinosa, J. 2007. "Learning from Experience in Software Development: A Multilevel Analysis," *Management Science* (53:8), pp. 1315-1331.

Burk, W. J., Steglich, C., and Snijders, T. 2007. "Beyond Dyadic Interdependence: Actor-Oriented Models for Co-Evolving Social Networks and Individual Behaviors," *International Journal of Behavioral Development* (31), pp. 397-404.

Burt, R. S. 1992. *Structural Holes*, Cambridge, MA: Harvard University Press.

Burt, R. S. 2004. "Structural Holes and Good Ideas," *American Journal of Sociology* (110:2), pp. 349-399.

Chengalur-Smith, S., and Sidorova, A. 2003. "Success and Survival of Open-Source Projects: A Population Ecology Perspective," in *Proceedings of the 24th International Conference on Information Systems*, A. Massey, S. March, and J. I. DeGross (eds.), Seattle, WA, December 14-17, pp. 782-787.

Cohen, W. M., and Levinthal, D. A. 1990. "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly* (35:1), pp. 128-152.

Coleman, J. 1988. "Social Capital in the Creation of Human Capital," *American Journal of Sociology* (94), pp. S95-S120.

Crowston, K., Annabi, H., and Howison, J. 2003. "Defining Open Source Software Project Success," in *Proceedings of the 24th International Conference on Information Systems*, A. Massey, S. March, and J. I. DeGross (eds.), Seattle, WA, December 14-17, pp. 327-340.

DeLone, W. H., and McLean, E. R. 1992. "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research* (3:1), pp. 60-95.

Fleming, L. 2001. "Recombinant Uncertainty in Technological Search," *Management Science* (47:1), pp. 117-132.

Fleming, L., and Waguespack, D. M. 2007. "Brokerage, Boundary Spanning, and Leadership in Open Innovation Communities," *Organization Science* (18:2), pp. 165-180.

Fogel, K. 2005. *Producing Open Source Software: How to Run a Successful Free Software Project*, Sebastobol, CA: O'Reilly Media.

Gelman, A., and Hill, J. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*, New York: Cambridge University Press.

Granovetter, M. 1973. "The Strength of Weak Ties," *American Journal of Sociology* (78:6), pp. 1360-1380.

Greene, W. H. 2003. *Econometric Analysis* (6th ed.), Upper Saddle River, NJ: Prentice Hall.

Grewal, R., Lilien, G. L., and Mallapragada, G. 2006. "Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems," *Management Science* (52:7), pp. 1043-1056.

Gulati, R., and Gargiulo, M. 1999. "Where Do Interorganizational Networks Come From?," *The American Journal of Sociology* (104:5), pp. 1439-1493.

Hahn, J., Moon, J., and Zhang, C. 2006. "Impact of Social Ties on Open Source Software Project Formation," in *Proceedings of the 2nd International Conference on Open Source Systems*,

E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi (eds.), Como, Italy, June 8-10, pp. 307-317.

Heckman, J. 1991. "Identifying the Hand of Past: Distinguishing State Dependence from Heterogeneity," *American Economic Review* (81:2), pp. 75-79.

Herbsleb, J., and Mockus, A. 2003. "An Empirical Study of Speed and Communication in Globally-Distributed Software Development," *IEEE Transactions on Software Engineering* (29:6), pp. 481-494.

Herbsleb, J. D., Mockus, A., and Roberts, J. A. 2006. "Collaboration in Software Engineering Projects: A Theory of Coordination," in *Proceedings of the 27th International Conference on Information Systems*, Milwaukee, WI, December 10-13.

Jaffe, A. B. 1986. "Technological Opportunity and Spillovers in R&D: Evidence from Firms' Patents, Profits and Market Value," *American Economic Review* (76), pp. 984-1001.

Jaccard, J., and Turrisi, R. 2003. *Interaction Effects in Multiple Regression*, Thousand Oaks, CA: Sage Publications.

Jones, T. C. 1984. "Reusability in Programming: A Survey of the State-of-the Art," *IEEE Transactions on Software Engineering* (10:5), pp. 484-494.

Kim, Y., and Stohr, E. A. 1998. "Software Reuse: Survey and Research Directions," *Journal of Management Information Systems* (14:4), pp. 113-148.

Kuhn, T. 1970. *The Structure of Scientific Revolutions*, Cambridge, UK: Cambridge University Press.

Kuhn, T. 1992. *The Structure of Scientific Revolutions*, Cambridge, UK: Cambridge University Press.

Kuk, G. 2006. "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List," *Management Science* (52:7), pp. 1031-1042.

Laumann, E., Marsden, P., and Prensky, D. 1983. "The Boundary Specification Problem in Network Analysis," in *Applied Network Analysis: A Methodological Introduction*, R. S. Burt and M. J. Minor (eds), Thousand Oaks, CA: Sage Publications, pp. 18-34.

Lee, N-Y, and Litecky, C. R. 1997. "An Empirical Study of Software Reuse with Special Attention to Ada," *IEEE Transactions on Software Engineering* (23:9), pp 537-549.

Lerner, J., and Tirole, J. 2005. "The Scope of Open Source Licensing," *Journal of Law, Economics, & Organization* (21:1), pp. 20-56.

Marsden, P. V. 2005. "Recent Developments in Network Measurement," in *Models and Methods in Social Network Analysis*, P. Carrington, J. Scott, and S. Wasserman (eds.), New York: Cambridge University Press, pp. 8-30.

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

Mockus, A., and Herbsleb, J. 2002. "Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source," *Proceedings of the 2nd Workshop on Open Source Software Engineering*, pp. 35-37.

Mouw, T. 2006. "Estimating the Causal Effect of Social Capital: A Review of Recent Research," *Annual Review of Sociology* (32), pp. 79-102.

Narduzzo, A., and Rossi, A. 2003. "Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed," Department of Computer and Management Sciences, University of Trento, Italy (available online at http://repec.cs.unitn.it/Q/Doc/078.pdf).

Nijman, T., and Verbeek, M. 1992. "Nonresponse in Panel Data: The Impact on Estimates of a Life Cycle Consumption Function," *Journal of Applied Econometrics* (7), pp. 243-257.

Nuvolari, A. 2005. "Open Source Software Development: Some Historical Perspectives," Eindhoven Centre for Innovation Studies (available online at http://opensource.mit.edu/papers/nuvolari.pdf).

Olson, G. M., and Olson, J. S. 2000. "Distance Matters," *Human-Computer Interaction* (15:2/3), pp. 139-178.

Paulson, J. W., Succi, G., and Eberlein, E. 2004. "An Empirical Study of Open-Source and Closed Source Software Products," *IEEE Transactions on Software Engineering* (30:4), pp. 246-256.

Portes, A. 1998. "Social Capital: Its Origins and Applications in Modern Sociology," *Annual Review of Sociology* (24), pp. 1-24.

Reagans, R., and McEvily, B. 2003. "Network Structure and Knowledge Transfer: The Effect of Cohesion and Range," *Administrative Science Quarterly* (48:2), pp. 240-267.

Reagans, R., and Zuckerman, E. 2001. "Networks, Diversity, and Productivity: The Social Capital of Corporate R&D Projects," *Organization Science* (12:4), pp. 502-517.

Reagans, R., Zuckerman, E., and McEvily, B. 2004. "How to Make the Team: Social Networks vs. Demography as Criteria for Designing Effective Projects," *Administrative Science Quarterly* (40:1), pp. 101-133.

Sacks, M. 1994. *On-the-Job Learning in the Software Industry*, Westport, CT: Quorum Books.

Sampson, R. C. 2007. "R&D Alliances and Firm Performance: The Impact of Technological Diversity and Alliance Organization on Innovation," *Academy of Management Journal* (50:2), pp. 364-386.

Shankland, S. 2002. "Standardized Linux Gains Support," CNET News, October 8 (available online at http://news.cnet.com/Standardized-Linux-gains-support/2100-1016_3-961296.html).

Singh, P. V. 2010. "The Small World Effect: The Influence of Macro-Level Properties of Developer Collaboration Networks on Open-Source Project Success," *ACM Transaction on Software Engineering and Methodology* (20:2), Article 6.

Singh, P. V., and Phelps, C. 2009. "Determinants of Open Source Software License Choice: A Social Influence Perspective," Working Paper, Carnegie Mellon University (available online at http://papers.ssrn.com/sol3/papers.cfm?abstract_id= 1436153).

Singh, P. V., and Tan, Y. 2005. "Stability and Efficiency of Communications Networks in Open Source Software Development," in *Proceedings of the 15th Annual Workshop on Information Technologies and Systems*, Las Vegas, NV, December 10-11.

Singh, P. V., Tan, Y., and Youn, N. 2010. "A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects," *Information Systems Research*, forthcoming.

Stewart, K. J, Ammeter, T. A., and Maruping, L. 2006. "Impacts of License Choice and Organizational Sponsorship on User

Interest and Development Activity in Open Source Software Projects," *Information Systems Research* (17:2), pp. 126-144.

Stewart, K. J., and Gosain, S. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Projects," *MIS Quarterly* (30:2), pp. 291-314.

Szulanski, G. 1996. "Exploring Internal Stickiness: Impediments to the Transfer of Best Practice Within the Firm," *Strategic Management Journal* (17), pp. 27-43.

Uzzi, B. 1997. "Social Structure and Competition in Interfirm Networks: The Problem of Embeddedness," *Administrative Science Quarterly* (42:1), pp. 35-67.

Uzzi, B., and Spiro, J. 2005. "Collaboration and Creativity: The Small World Problem," *American Journal of Sociology* (11:2), pp. 447-504.

van Antwerp, M., and Madey, G. 2008. "Advances in the SourceForge Research Data Archive (SRDA)," in *Proceedings of the 4th International Conference on Open Source Systems*, Milan, Italy, September 7-10 (available online at http://www.nd.edu/~oss/Papers/srda_final.pdf).

von Hippel, E., and von Krogh, G. 2003. "Open Source Software and the 'Private–Collective' Innovation Model: Issues for Organization Science," *Organization Science*, (14:2), pp. 209-225.

Wasserman, S., and Faust, K. 1994. *Social Network Analysis*, London: Cambridge University Press.

Wooldridge, J. 2007. *Econometric Analysis of Cross Section and Panel Data*, Cambridge, MA: The MIT Press.

Xiao, A., and Tsui, A. S. 2007. "When Brokers May Not Work: The Cultural Contingency of Social Capital in Chinese High-Tech Firms," *Administrative Science Quarterly* (52:3), pp. 1-31.

Xu, J., Gao, Y., Christley, S., and Madey, G. 2005. "A Topological Analysis of the Open Source Software Development Community," in *Proceedings of the 38th Hawaii International Conference on System Sciences*, Los Alamitos, CA: IEEE Computer Society Press.

About the Authors

Param Vir Singh is an assistant professor of Information Systems at the Tepper School of Business, Carnegie Mellon University. His research interests include dynamic structural models, hidden Markov models, social networks and open source software development. A primary focus of his research is to design and study the effect of policy interventions on knowledge worker behavior. His research has been published or is forthcoming in various outlets including *Information Systems Research, ACM Transactions on Software Engineering and Methodology*, and *Journal of Management Information Systems*.

Yong Tan is an associate professor of Information Systems and Evert McCabe Faculty Fellow at the Michael G. Foster School of Business, University of Washington. His research interests include economics of information systems, social networks, electronic commerce, and software engineering. He has published in various journals such as *Management Science, Information Systems Research,* and *Operations Research*. He is an associate editor of *Management Science* and *Information Systems Research*.

Vijay S. Mookerjee is the Charlges and Nancy Davidson Distinguished Professor of Information Systems at the School of Management, University of Texas at Dallas. He holds a Ph.D. in Management, with a major in MIS, from Purdue University. His current research interests include social networks, optimal software development methodologies, storage and cache management, content delivery systems, and the economic design of expert systems and machine learning systems. He has published in and has articles forthcoming in several archival Information Systems, Computer Science, and Operations Research journals. He serves (or has served on) on the editorial board of *Management Science*, *Information Systems Research*, *INFORMS Journal on Computing*, *Operations Research*, *Decision Support Systems*, *Information Technology and Management*, and *Journal of Database Management*.