Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2007 Proceedings

International Conference on Information Systems (ICIS)

December 2007

Open Source Software Development and the Small World Phenomenon: An Empirical Investigations of Macro Level Colaboration Network Properties on Project Success

Param Singh
University of Washington

OPEN SOURCE SOFTWARE DEVELOPMENT AND THE SMALL WORLD PHENOMENON: AN EMPIRICAL INVESTIGATION OF MACRO LEVEL COLLABORATION NETWORK PROPERTIES ON PROJECT SUCCESS

Param Vir Singh
Information Systems and Operations Management
Michael G. Foster School of Business
University of Washington
Seattle, Washington
psidhu@u.washington.edu

Abstract

In this study, I investigate whether small world properties of developer collaboration networks influence developer productivity. Small world networks are characterized by two macro level network properties: dense clustering and short average path length. Structure of relationships among developers affects how information and knowledge is shared among them. I argue that networks characterized by both high clustering and short average path length would positively influence the productivity of the developers by providing them with speedy and reliable access to more and more varied information. Using hierarchical linear modeling, I analyze extensive data collected from 2013 open source projects and 8512 developers belonging to 15 different networks hosted at Sourceforge.net. I find evidence that 'small world' properties of these networks positively affect their members' productivity. The findings are robust to a number of controls and model specifications.

Keywords: Open source software development, social networks, small world networks, innovation, collaborative software development

Introduction

Recent studies have acknowledged and emphasized the importance of relationships of developers among the larger community in open source software (OSS) for their success (Fleming and Waguespack 2007; Grewal et al. 2006; Madey et al. 2002; Singh and Tan 2005; Xu et al. 2005). As OSS developers work concurrently across several projects, they weave a network of direct and indirect relationships among them. These relationships act as channels that facilitate flows of resources in a network of developers, with each developer acting as an originator as well as a receiver (Coleman 1988; Granovetter 1973).

Organizational social networks research is profuse with evidence indicating that differential location in an inter-organizational/inter-innovator network results in organizations/innovators having divergent capabilities for benefiting from such networks (Ahuja 2000; Baker 1990; Gulati 1998; Gulati and Gargiulo 1999). This literature emphasizes the inter-actor linkages as closed conduits, characterized by legal arrangements such as non-disclosure agreements and exclusive licensing contracts that transfer intellectual property rights designed to ensure that only the parties to a given connection can benefit from the information that is exchanged (Owen-Smith and Powell 2004). This implies that the actors who are more central to the information exchange in the network are likely to be advantaged in their innovative activities through better access to information.

However, OSS philosophy is based on principles and practices that promote the free and frictionless sharing of intellectual properties such as production and design processes and outcomes (Weber 2004). A direct implication of such a philosophy is that the linkages among developers in OSS are channels, rather than closed conduits, that diffusely and imperfectly direct the flow of knowledge and resources and facilitate huge amounts of knowledge spillovers in the network. The developers who are central to the information flow would not only benefit themselves by their positions, but would also make the whole network richer by facilitating the information to reach those developers who would otherwise be at a positional disadvantage. In such a scenario, it is not the specific location in the network but the overall network structure that is important. As we begin to unravel the community-based collaborative software development paradigm, it becomes pertinent to investigate: Are some collaboration structures more conducive to software development than others? If so, what properties characterize such structures? Some network structures may be more conducive to information flow than others, and the location of a developer in such a network would provide him or her with knowledge advantages over developers who are not similarly situated.

As a step towards addressing these questions, I investigate how the macro level properties of *large scale* networks of OSS developers influence the productivity of the developers. A large scale network is defined as the relationships that exist among members of a population, such as OSS developers working on the same but sufficiently broad technological platform (Wellman 1988). I focus on large scale networks as they provide a meaningful context for knowledge and resource sharing among members. I specifically focus on small world properties of large scale networks: *clustering* and *average path length*. These properties are macro level properties of a network. Network theorists have coined the term *small world* to designate the networks that exhibit the simultaneous existence of localized pockets of dense clustering and short average path length (Milgram 1967; Watts and Strogatz 1998; Watts 1999). I argue that networks with dense clustering and short average path length together (small world networks) would positively influence software development activities of the member developers.

Clusters, by definition, include developers who have more relationships connecting them to each other than to other developers in the network. Clustering fosters trust among developers, which in turn encourages knowledge sharing (Coleman 1988). Clustering also provides redundant paths for knowledge transmission, resulting in high transmission capacity for the network. I argue that clustering would positively influence software development activities by increasing knowledge sharing and the information transmission capacity among developers.

The greater the number of developers that act as intermediaries in information or knowledge transfer, the more time the transmission will take and the more likely it is that it will lose integrity (Bartlett 1932). The number of links that lie in a path for information or knowledge transfer between any two developers in a network represents the path length between them. Short average path length in a network provides access to diverse resources and minimizes information decay during transfer, while increasing the speed of transmission (Uzzi and Spiro 2005). I argue that short average path length of a large scale network would positively influence the software development activities of member developers by providing them quick access to a wide variety of reliable resources.

I begin by introducing the emerging literature on small world networks. Then I describe the formation of social networks among OSS developers. I then explain the process of OSS development and the importance of access to

information and resources for OSS developers. From this, I develop hypotheses relating macro level network properties and developer productivity. I test my hypotheses using detailed data collected from the OSS projects hosted at Sourceforge.

Small World Networks

Though formal investigation of the relationship between small world structures and organizational outcomes has started only recently, the interest in the small world phenomenon arose following the set of experiments conducted by Milgram (1967). These experiments were designed to find out the probability that two randomly selected individuals would know each other. The results were quite surprising in that the average number of persons needed to connect any two randomly selected persons was found to be six. It gave rise to the famous notion of "six degrees of separation." It showed that seemingly unconnected people are surprisingly quite close in social space.

More detailed analysis of the reasons behind this finding suggests that people tend to interact in dense clusters. This idea of formation of clusters stems from the notion of homophily in network formation – similarity breeds connection (McPherson et al. 2001) (i.e., friends of a friend are generally friends). All friends share a direct relationship with each other, that is, they have one degree of separation. If some of the friends belong to multiple clusters, it gives a cluster access to a completely new set of people with an additional degree of separation. These friends then act as bridges that connect different clusters. The presence of dense clustering and bridging relationships dramatically reduces the average path length between any two people in the network (Watts 1999; Watts and Strogatz 1998).

A large volume of social network thought concerned with knowledge and information diffusion has followed from this view of the network structure. The "weak ties" argument by Granovetter (1973) follows logically from this perspective; people who belong to multiple clusters have to spread their resources across the clusters and, hence, have weak ties with other members in the cluster but have access to new diverse information. Burt (1992) coined the term "structural hole" to represent people who bridge these clusters; the people filling the holes have access to information and resources from both clusters and can obtain "brokerage" benefits. Contrary to Burt's view of advantage to people spanning structural holes, Krackhardt (1999) argued that such people are more constrained in their activities since they have to play by the rules of both clusters to stay members of each.

Though there was a common consensus on the small world view of the network in a variety of settings, the formal structural analysis of small world properties of a network was not undertaken for a long time until Watts and Strogatz (1998) formalized the properties by using graph theoretic notion. They calculated measures of clustering and average path length to represent small world properties. They proposed a small world coefficient, which is the ratio of clustering to average path length, to represent the small worldliness of the network; the higher values of this coefficient indicate greater small worldliness of the network. Small world network structure has been observed in a variety of settings: OSS developers relationship networks (Xu et al. 2005), ownership links among German firms (Kogut and Walker 2001), regional innovator networks (Fleming et al. 2007), Broadway artist collaboration networks (Uzzi and Spiro 2005), firm alliance networks (Baum et al. 2003), and others.

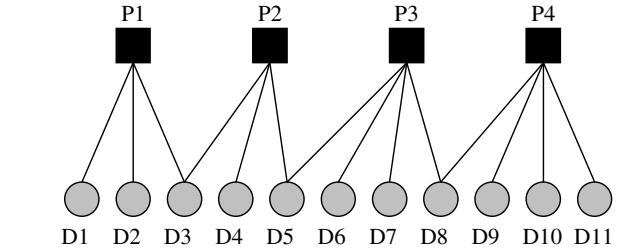
Organizational researchers have investigated the influence of small world properties on creativity and innovation activities in several settings. Uzzi and Spiro (2005) study the collaboration network of Broadway musical artists and conclude that the small world properties of a network positively influence the artistic and financial success of the musicals they produced. Schilling and Phelps (2007) find that the firms embedded in alliance networks that represent small world properties are more innovative than firms that are not embedded in such networks. Fleming et al. (2007) investigate regional innovation and find that although short path length fosters innovation, clustering does not.

OSS Developer Collaboration Networks

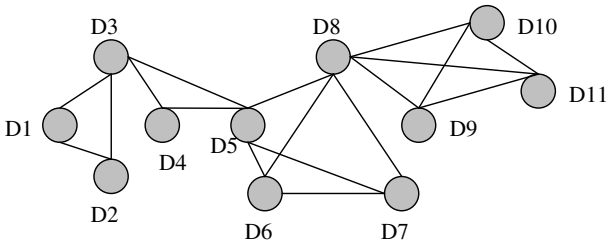
OSS developers may work on several projects at the same time and, hence, belong to multiple projects. A relationship exists between two developers if they work together on a project. Similarly, a relationship between two projects exists if they share some developer(s). Such a scenario of relationships can be represented by an affiliation network. An affiliation network is a special kind of two-mode social network that represents the affiliation of a set of actors with a set of social events (Wasserman and Faust 1994). Developers and the projects represent the two

modes of this network. In an affiliation network, relationships among members in one mode are based on linkages established in the other mode (Wasserman and Faust 1994).

A social network can be represented as a graph where nodes represent actors and lines joining the nodes represent relationships. Graphically, the social network of an OSS community can be represented by a bipartite graph depicting the developer-project network. Figure 1a shows a bipartite graph of a sample population of four projects and 11 developers. A relationship between a project and a developer is shown by a line. The unipartite projection (i.e., relationships among members of a mode at the developer level) is shown in Figure 1b. By definition, all the developers who work on a given project are related to each other. Note that the line between any two nodes indicates the existence of a relationship, but not the quality of the relationship. As more developers join the community and new projects begin, the network evolves into a giant mesh of relationships.



(a) Project-Developer Network (Bipartite Graph)



(b) Developer Network (Unipartite Projection Graph)

FIGURE 1. Social Network of 4 Projects and 11 Developers

Notes: Projects (developers) are indicated by black squares (grey spheres).

Networks of OSS developers at Sourceforge tend to be highly clustered (Xu et al. 2005). This observation could be a consequence of homophily or absorptive capacity constraints playing a role in a developer's decision to join a project. By the homophily argument, developers are more likely to work with developers with whom they have enjoyed working in the past. This may lead to redundant relationships among a group of developers. By the absorptive capacity argument, developers are more likely to work on several projects that require similar technological skills than on projects that require very different skills (Cohen and Levinthal 1990; Grewal et al. 2006; Xu et al. 2005). In fact, the developer networks at Sourceforge, the largest OSS hosting repository, have been found to resemble small world networks (Xu et al. 2005). A network of OSS developers for Gnome Foundry, a broad technological platform at Sourceforge, is shown in Figure 2. This network is very highly clustered with a very short average path length.

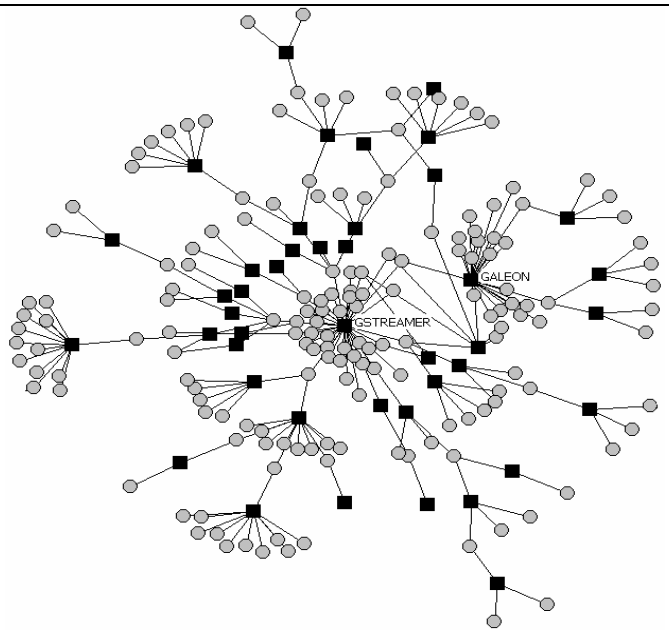


FIGURE 2a. Affiliation Network for the Largest Component of the Gnome Foundry

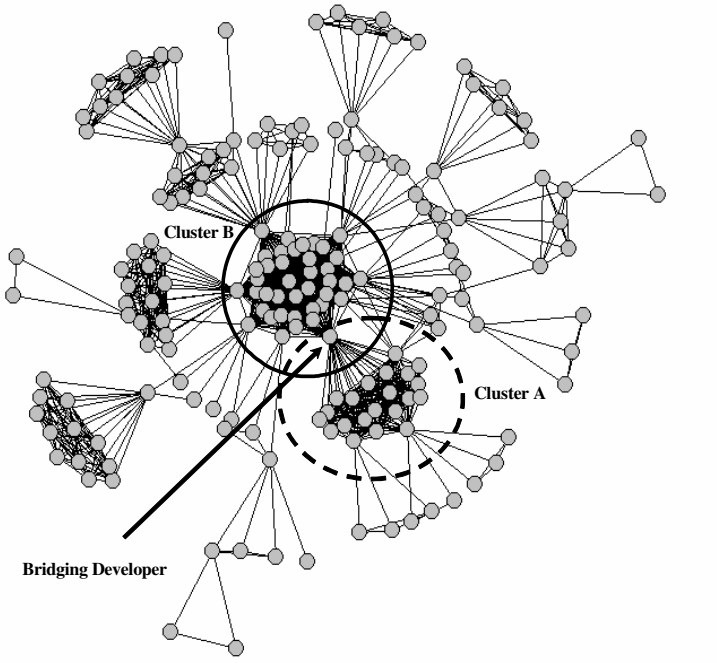


Figure 2b. Unipartite Projection of 2a

Notes: Projects are indicated by black squares. Developers are indicated by grey spheres. A few projects (such as GStreamer and Galeon) have many developers, whereas most projects have only a few. There are several clusters in the network. Two different clusters (A and B) are shown by circles around them. A bridging developer is also indicated acting as a bridge between clusters A and B.

OSS Development Process

OSS is developed by voluntary developers working in teams. In contrast to traditional software development, which follows an engineering archetype, OSS follows an evolutionary archetype for software development (Blaauw and Brooks 1997; Weber 2004). In an evolutionary archetype, roles are not assigned, and the problem definition evolves over time. Software is developed under a modular structure, which allows developers to work in parallel (Baldwin and Clark 2006; MacCormack et al. 2006; Weber 2004). Developers follow an adaptive process to solve technical problems. There are an infinite number of paths to solve a problem and choosing among various alternatives involves a high degree of experimentation, requiring leeway and discretion among developers (Weber 2004). This process requires an effectual normative environment that promotes trust and cooperation among developers.

Prior research suggests that the search processes that lead to the creation of new knowledge embodied in complex artifacts such as software most often involve, to a substantial extent, a recombination of known conceptual and physical materials (Fleming 2001). Rather than designing a software module from scratch, previously existing modular architecture is frequently inherited (Baldwin and Clark 2006; MacCormack et al. 2006; Narduzzo and Rossi 2003). For example, the GNU and the FreeBSD projects closely resemble the UNIX architecture (Narduzzo and Rossi 2003). Software evolves from an initially integrated system by recombination of modules and evolutionary adaptation (Baldwin and Clark 2006; Fleming 2001; MacCormack et al. 2006; Narduzzo and Rossi 2003). Key inputs into this process include access to and understanding of a variety of knowledge elements, such as current design problems, related existing solutions, technical know-how, failed approaches, user requirements, current market trends, etc. (Fleming 2001; Kuk 2006; Hargadon and Sutton 1997; Schilling 2003; Schilling and Phelps 2007). Developers exploit their access to recombinatory resources – a broad range of technological solutions – by making analogies between current design problems, related existing solutions, and failed approaches (Hargadon and Sutton 1997; Fleming 2001). Developers who have greater access to and understanding of these recombinatory resources would have an advantage in their efforts to develop software. I focus on measures of productivity that would quantify a project's success in terms of the amount of technical development required. This measure of project success is consistent with the literature on Information System success (DeLone and McLean 1992) and has been used in closely related research (Grewal et al. 2006).

Hypothesis Formulation

Relationships among developers in a network provide them with two types of knowledge benefits: resource sharing and knowledge spillovers (Ahuja 2000). Resource sharing allows developers to combine know-how and physical assets, whereas knowledge spillovers provide information about current design problems, failed approaches, new breakthroughs, and opportunities (Ahuja 2000; Kogut and Zander 1992). Before we discuss the influence of network structure, the subtle difference between know-how and information needs to be emphasized here. Know-how corresponds to accumulated practical skills or expertise, and involves a significant tacit component (Ahuja 2000; von Hippel 1988). Information is primarily explicit knowledge that includes facts, propositions, and symbols that can be easily codified and transmitted without a loss of integrity (Kogut and Zander 1992).

The structure of relationships among developers affects the flow of information and resource sharing in the network. Only relationships that involve frequent interaction among developers (relationships within a cluster) will be able to provide the benefits of resource sharing as well as knowledge spillovers (Ahuja 2000; Kogut and Zander 1992; Uzzi 1997). Relationships in which developers do not interact frequently (relationships across clusters) will be able to provide benefits comprised mainly of knowledge spillovers (Ahuja 2000; Kogut and Zander 1992; Uzzi 1997).

Clustering

Clustering measures the extent to which two connected developers in a network share a relationship with a common third developer (i.e., it measures the fraction of all triads that are closed). Clustering or local network closure develops group solidarity and identity, which helps monitor the behavior of members, and also helps carry out sanctions against developers who do not follow group norms (Coleman 1988; Ullmann-Margalit 1977). This reduces uncertainty in the behavior of others and, hence, increases levels of trust among developers in an exchange (Kollock 1994).

Trust and reciprocity in a relationship promote frequent interactions among developers. This strengthens their relationship, which promotes information and resource sharing among them. Once introduced into the network, information is quickly relayed to the developers in the cluster. Redundant (multiple) paths between two developers in a cluster ensure the reliability of the information received (Coleman 1988; Kollock 1994). This also discourages members from acting opportunistically since such information would be quickly relayed to other members.

As clusters are likely to be composed of developers who have similar or complementary technological skills, clusters provide a meaningful and useful structure for information exchange (Schilling and Phelps 2007; Uzzi and Spiro 2005). Clusters promote shared understanding of the problem and stimulate joint problem solving (Schilling and Phelps 2007). Developers have the ability to call upon their fellow developers in the cluster for help in solving innovative problems. Developers benefit from clustering as it increases the amount, quality, and the likelihood of receiving information and resources.

Hypothesis 1: *The clustering of a network will be positively related to the technical success of the software produced by the developers embedded in it.*

Average Path Length

Though information is transmitted quickly and reliably within a cluster, the speed and integrity of knowledge diffusion across the network is directly related to the average path length in the network (Fleming et al. 2007; Schilling et al. 2007; Uzzi and Spiro 2005). Average path length measures the average shortest distance between any two actors in the network. Shorter average path length increases the vision (knowledge of potential resources) of the developers in the network. It also maximizes the speed of transmission and minimizes decay in information while in transmission (Schilling and Phelps 2007; Watts 1999). Developers embedded in such networks are less likely to behave opportunistically as such information would quickly be relayed to the entire network, thus hurting their reputation. Moreover, such networks provide a larger audience for developers to show their programming prowess to accrue reputation benefits (Lerner and Tirole 2002).

Short average path length indicates that the network would be rich in direct relationships that help in efficient resource pooling and knowledge spillovers (Ahuja 2000). Developers benefit from exposure to new information through easier access to non-local perspectives (Fleming et al. 2007). Short average path length in the network provides developers access to different technological and organizational perspectives. Since the software development process involves recombining existing knowledge with novel approaches, developers embedded in networks exhibiting short path length will be advantaged in their development efforts due to their access to diverse knowledge resources and new perspectives.

Hypothesis 2: *The average path length of a network will be negatively related to the technical success of the software produced by the developers embedded in it.*

Methods

To test these hypotheses, I focus on OSS projects hosted at Sourceforge. Sourceforge is the primary hosting place for OSS projects provided by the OSTG group and accounts for about 90 percent of all OSS projects. Compared to Sourceforge's more than 120,000 OSS projects, its competitors – such as Savannah, Tigris, Bounty Source, and BerliOS – have collectively less than 10,000 projects. Though not all OSS projects are hosted at Sourceforge, it has been argued that it is the most representative of the OSS movement, in part because of its popularity and the large number of developers and projects registered there (Grewal et al. 2006; Madey et al. 2002; Xu et al. 2005). Researchers interested in investigating issues related to the OSS phenomenon have predominantly used Sourceforge data (Grewal et al. 2006; Madey et al. 2002; von Hippel and von Krogh 2003; Xu et al. 2005). Sourceforge provides both web space and services such as a mailing facility, discussion forums, CVS repository hosting, and download servers for OSS projects to organize and coordinate the software development. Projects hosted at Sourceforge are classified under 25 broad technology platforms called project foundries. For example, Java, Perl, Games, PHP, Python, Basic, 3D, Tool Command Language, Gnome, Linux Kernel, and Vector Graphics are few of the foundries at Sourceforge.

Network Construction

Consistent with the empirical social networks research dealing with large networks (e.g., Grewal et al. 2006; Gulati and Gargiulo 1999), I establish network boundaries to render the data more manageable. I follow the predominant procedural tactic for defining a network boundary: attributes of actors or events that rely on membership criteria (Wasserman and Faust 1994). In an affiliation network, a boundary is specified for both the actors and the events simultaneously (Wasserman and Faust 1994). Accordingly, my network construction criterion is based on developers and projects being a member of a foundry. The choice of using participation in a foundry as a network boundary is particularly important and fitting for two reasons. First, each foundry represents a focused technology platform, and hence, more efficient knowledge sharing would be possible within a foundry rather than across foundries. Using the foundry as a network boundary for OSS projects at Sourceforge has been used in related research (Grewal et al. 2006). Second, I analyzed memberships for 2000 randomly selected developers who work on multiple projects and found that only approximately 4 percent of those worked across two or more foundries.

Developer and project affiliation data were collected for projects that were registered at Sourceforge from November 1999 (Sourceforge's inception date) to January 2003 from the Sourceforge database hosted by the OSTG group.[1] By January 2003, Sourceforge had enough mass of registered developers and projects in each foundry for social capital to be a significant resource. As Chengalur-Smith and Sidorova (2003) note, a large number of projects at Sourceforge never show any activity. Including such projects in the network construction may bias the network measures as those projects are essentially dead nodes in the network which would not allow any information or knowledge flow. Hence, I construct the network of developers using the project-developer snapshot data available in January 2003 for active projects only. All active projects within a foundry were selected, along with the associated developers. A separate affiliation network was constructed for each foundry. I restrict my attention to large components in these foundries since small world properties (specifically average path length) are undefined for networks involving disconnected components. A component in a network is composed of all developers who can reach each other through lines of relationships (i.e., any two developers in the component have a finite path length between them). Developers from two different components are mutually unreachable and have a path length of infinity between them. This approach has been followed in most of the literature that investigates the small world phenomenon (Baum et al. 2003; Davis et al. 2003; Fleming et al. 2007; Kogut and Walker 2001). The limitations and implications of these sampling decisions are presented in the discussion section.

Each affiliation network was used to create unipartite projections for the developer network. From each unipartite projection, an undirected binary adjacency matrix was developed. An adjacency matrix represents the relationships between any two developers in the network. The row and the column headings represent the developers. A value of one corresponding to two developers in the network indicates the existence of a relationship between them; a value of zero indicates the absence of such. The adjacency matrix is undirected as the relationship between two developers is mutual. For each network, measures were calculated from the undirected binary adjacency matrices. Some of the foundries did not have enough relationships among developers to be considered a large network and hence were not included in the sample. The development activity data for these projects was collected from the homepage of each project at Sourceforge by using web-agents. The final sample includes 2013 projects and 8512 developers across 15 networks.

Dependent Variable: CVS Commits

To operationalize project success, I use a number of CVS commits. Similar measures have been used in a related study by Grewal et al. (2006), who investigate the impact of network embeddedness on project success.

Software developers use Concurrent Versioning System (CVS) to manage the software development process. CVS stores the current version(s) of the project and its history. A developer can check-out the complete copy of the code, work on this copy, and then check in the revised code with his or her changes. This allows multiple developers to work on the code concurrently on their own working copies and send their modifications to the central server. The modifications are peer reviewed. The CVS updates the modified file automatically and registers it as a "commit". The server only accepts changes made to the most recent file versions of a file, thus avoiding conflicting commits. The CVS keeps track of what change was made, who made the change, and when the change was made. This

[1] https://zerlot.cse.nd.edu/mywiki/index.php?title=Dataset

information can be gathered from the log files of the CVS repository of a project. CVS commits provide a measure of novel invention that is externally validated by peers. As CVS commits signify only meaningful changes to the code and represent a way in which knowledge creation is instantiated (Crowston et al. 2003; Grewal et al. 2006), I use the number of CVS commits as a measure of project success.

Benefits from a relationship can be observed only after the relationship has been established. The relationships among developers at Sourceforge, once established, remain for a long time since (1) OSS projects do not have a fixed deadline and are in continuous development (Weber 2004), and (2) the associated developers at Sourceforge rarely drop out (Crowston et al. 2003). This characteristic calls for the use of lag specification between network structure and success measures (Gulati and Gargiulo 1999). I measure the dependent variable, *CVSit*, as the number of CVS commits for project i in t years since January 2003. For instance, for $t = 1$, I count the number of CVS commits for a project from January 2003 to January 2004.

Independent Variables

Clustering Coefficient Ratio (CCR)

The clustering coefficient measures the degree of clustering on a graph, i.e., the extent to which two related developers share a relationship with a common third developer. A clustering coefficient is a measure of the degree to which the network contains localized pockets of dense connectivity (Watts and Strogatz 1998; Watts 1999). Following Watts and Strogatz (1998), I measure the clustering coefficient (*CC*) for the actual graph as follows:

$$CC = \frac{3 \times \text{number of triangles on the graph}}{\text{number of connected triples of vertices}}.$$

The "triangles" are trios of vertices where each is connected to another two, and "connected triples" are trios where at least one is connected to the other two (Newman et al. 2001; Watts and Strogatz 1998). To account for the three different configurations that may arise for a closed trio of three vertices, a factor three is added in the numerator (Watts and Strogatz 1998). This also ensures that the *CC* lies strictly between 0 and 1. The value of 1 would represent that all the developers in the graph share a direct relationship with each other (i.e., extreme clustering). A value of 0 would indicate that no two developers share a relationship with a common third (i.e., zero clustering).

However, the unipartite projection of the bipartite graph may be a misleading indicator of small world if not properly corrected (Newman et al. 2001). This is due to the fact that, by definition, the developers associated with a given project share a direct relationship with each other. All developers associated with a project having more than two developers constitute triangles through their intra-project links. Hence, the unipartite projection of the bipartite graph may overstate the clustering coefficient and under-state the path length (Newman et al. 2001). Following Newman et al.'s suggestion, I use the clustering coefficient of a random bipartite graph (*rCC*) with identical degree distribution as that of my actual network to normalize the *CC*. Newman et al. (2001) provide this clustering coefficient, which accounts for clustering between teams over and above the artifactual clustering within teams. The clustering coefficient ratio (*CCR*) is then calculated as the ratio of the clustering coefficient of the actual graph to that of the random graph:

$$CCR = CC / rCC .$$

Inverse Path Length (IPLR)

The path length (*PL*) for the actual graph is calculated as the weighted average of the path length of each developer in the network (Uzzi and Spiro 2005). The path length of the actual graph is normalized by the path length of the random graph (*rPL*) with identical degree distributions. The inverse path length ratio (*IPLR*) is then calculated as the ratio of *rPL* to *PL*:

$$IPLR = rPL / PL .$$

Control Variables

Network Density

I control for the density of the network in which a project is embedded by calculating a variable *Network Density*, which measures the proportion of possible links that are actually present in the graph (Wasserman and Faust 1994). It goes from 0 (i.e., no links present) to 1 (i.e., all possible links are present). This controls for the rate and extent to which information diffuses in the network (Yamaguchi 1994).

Page Views

To control for the market potential of and user interest in the project, I calculate a variable *Page Views* as the cumulative number of page views for a project since its inception until January 2003 (Grewal et al. 2006; Krishnamurthy 2002; Mockus et al. 2002).

Bugs and Support

Studies have shown that service and maintenance provided by a software manufacturer are critical factors that impact the adoption and evolution of software (Grewal et al. 2006; von Hipple and von Krogh 2003; Weber 2004). To control for the service and maintenance activities provided by the project I use two variables: *Bugs* and *Support*. I calculate the variable *Bugs* as the cumulative number of bugs closed by the project since its inception until January 2003. The variable *Support* is calculated as the cumulative number of support requests answered by the project since its inception until January 2003. Both these variables represent encounters initiated by users of the software and, hence, also signify the role that the larger open source community plays in project evolution (Crowston et al. 2003; Grewal et al. 2006; Krishnamurthy 2002; Mockus et al. 2002).

Project Age and Size

Software life cycle may also influence project success measures since projects are likely to see relatively higher CVS commit rates close to the inception of the project compared to later stages when the complexity and advanced development of the software makes it harder to make improvements. I control for these issues by calculating a *Project Age* variable as the number of months since the project's inception at Sourceforge until January 2003 (Grewal et al. 2006; Singh and Tan 2006). To control for any non-linear relationship between age and project success measures, I also include a squared term of *Project Age* (Singh and Tan 2006). I calculate a variable *Project Size* as the number of developers involved in a project (Chengalur-Smith and Sidarova 2003; Singh and Tan 2006). This controls for the labor and knowledge capital at hand for a project.

Project Characteristics

I constructed an extensive range of variables to account for the project level heterogeneity on the dependent variables. At Sourceforge, a team is required to report the project's type, intended audience, language, and user interface. I control for all these variables by constructing dummy variables. The type of the project indicates the approximate potential market size of the software. I control for 14 different types of projects ranging from games to education to software utilities. Intended audience may influence the quality of developers that are attracted to a project. For example, software that is aimed at system administrators is likely to attract more sophisticated developers (Lerner and Tirole 2002; Roberts et al. 2006). I control for this effect by constructing dummies for system administrators, developers, and end users.

Pre-sample CVS

To control for remaining unobserved heterogeneity, I follow the pre-sample information approach proposed by Blundell et al. (1995). This approach relies on the argument that in studies regarding knowledge creation, the unobserved heterogeneity lies in different knowledge stocks possessed by the creators at the time of their entry into the sample (Blundell et al. 1995). The pre-sample variables serve as a control for the inherent knowledge capital of the creators. Using pre-sample variable in a cross-sectional equation provides a simple way to account for historical factors that cause current differences in the dependent variable that are difficult to account for in other ways (Wooldridge 2006). To control for unobserved heterogeneity concerning team productivity, I calculate a variable *Pre-sample CVS* as the cumulative number of CVS commits for a project since its inception at Sourceforge until January 2003.

Model Specification

The dependent variable, *CVS*, is a count variable but is heavily skewed. Therefore, I do a logarithmic transformation and approximate it as a continuous variable. The factors influencing the success of software are likely to be similar for the projects within a foundry, but may differ for projects across foundries (Cooper 2001; Chandrashekaran et al. 1999; Grewal 2006). All the projects in a network belong to the same foundry by construction and, hence, share similar factors influencing success and global network measure values. A multilevel/hierarchical linear model (HLM) is appropriate to model this nested structure where projects are embedded in networks/foundries (Raudenbush and Bryk 2001; Snijders and Bosker 1999). HLM accounts for the correlated errors within groups, and it may suffer from two other methodological threats to validity: reverse causality/simultaneity and unobserved heterogeneity (Raudenbush and Bryk 2001). I address these issues. In the present study, reverse causality would concern whether project success is shaping the network structure instead of the network structure influencing project success. Simultaneity would imply that both project success and network structure are being determined jointly by a system of simultaneous equations. In the present setup, this problem is negligible because independent variables are lagged by *t* number of years. Unobserved heterogeneity, in the present study, may arise from unobserved systematic differences in seemingly similar projects. Failure to control for unobserved heterogeneity may lead to spurious correlations (Wooldridge 2006). Though it is practically impossible to control for all possible sources of heterogeneity, there are a finite number of critical factors that may impact project success. At the project level, I control for project potential and user interest (*Page Views*), community support and service to users (*Bugs* and *Support*), innovative human capital (*Project Size*), project type, and intended audience. Finally, to address the issue of the appropriate lag structure of the independent variables, I employ alternative lags of independent variables relative to my dependent variable.

An HLM models both within- and between-group variability in a single analysis. In my analysis, the network represents the group (higher) level and the projects represent the individual (lower) level. I introduce the global network specific variables *CCR*, *IPLR*, and *Network Density* as group level variables in the model and use them to explain the variability between groups. The project-specific variables *Pre-sample CVScommits*, *Page Views*, *Bugs*, *Support*, *Project Age*, *Project Age Square*, and *Project Size* are introduced as individual level variables and are used to explain the within-group variability. All models were estimated using residual maximum likelihood estimation in *nlme* package in *R*. All the results presented are for a random intercept HLM. I tested for randomness of the individual level parameters using Deviance Tests as well as Akaike Information Criterion (AIC) as suggested by Snijders and Bosker (1999). The random intercept model outperforms all other specifications. None of the models were found to show any significant heteroskedasticity either at the individual or the group levels. The level one and level two residuals were also found to satisfy normality assumption when inspected.

Results

Table 1 provides the summary statistics for the dependent and independent variables. The mean, standard deviation, minimum, and maximum are provided for each variable. The maximum number of projects in a network was 813, and the minimum number was 16. The project size varied from 1 to 72 across projects.

Most of the project level control variables are heavily skewed. I performed a logarithmic transformation on the following control variables: *Page Views*, *Bugs*, *Support*, and *Pre-sample CVS*. To reduce the correlation between *Project Age* and its square, I mean center *Project Age* before taking its square. To avoid any confusion, I represent the log transformed variables by 'Ln' prefixed to the variable name. I also represent the grand mean centered variables by a subscript 'mc' attached to their name. Even after transformations, some of the correlations among variable were moderately high. This may lead to inflated standard errors and, in the worst case, unstable coefficients (Wooldridge 2006). I run several models with and without these highly correlated variables to ensure the stability of my results. I also estimated the model with generalized least squares. The results of these tests are consistent with the HLM results. To conserve space, I do not report these results in the paper. The results from the HLM model are reported in Table 2.

Table 1. Descriptive Statistics				
Total Number of Networks				

Total Number of Projects	2013			
Total Number of Developers	8512			
Variables	**Mean**	**SD**	**Min**	**Max**
CCR	1.31	0.33	1.02	2.42
PLR	1.36	0.29	0.91	2.32
Project Size	6.04	9.15	1.00	72.00
Page Views	355611.78	899312.08	0.00	39004745.0
Bugs	63.74	312.10	0.00	4831.00
Support	5.49	63.37	0.00	2396.00
Project Age	37.80	12.03	4.00	60.00

The results support both Hypothesis 1 and Hypothesis 2. Since the results are consistent across the two lag specifications, I will discuss the results for only Model 2.2. The R^2 at the group and the individual levels were calculated following the procedure given in Snijders and Bosker (1999). Model 2.2 yielded an individual level R^2 of 0.739 and a group level R^2 of 0.705. These R^2's can be interpreted as the proportional reduction of prediction error instead of the simple percentage of the variance accounted for.

Table 2. Hierarchical Linear Regression Results

DEPENDENT VARIABLE	$LnCVS_2$		$LnCVS_1$	
	MODEL 1.1	MODEL 1.2	MODEL 2.1	MODEL 2.2
FIXED EFFECTS				
Intercept	1.147***	1.323***	0.638***	0.842***
	(0.319)	(0.167)	(0.211)	(0.217)
LnPre-Sample CVS	0.433***	0.429***	0.400***	0.396***
	(0.020)	(0.019)	(0.018)	(0.018)
LnSupport	0.232	0.231	0.115	0.095
	(0.438)	(0.431)	(0.323)	(0.334)
LnBugs	0.256*	0.251*	0.224***	0.229***
	(0.141)	(0.140)	(0.082)	(0.084)
LnPage Views	0.073	0.069	0.048	0.047
	(0.101)	(0.092)	(0.114)	(0.114)
LnProject Size	0.118***	0.107***	0.092**	0.090**
	(0.039)	(0.040)	(0.037)	(0.036)
Network Density	-0.603	-1.452	-0.774	-1.388
	(1.513)	(1.130)	(1.366)	(0.883)
Project Age_{mc}	-0.798***	-0.810***	-0.757***	-0.759***
	(0.268)	(0.267)	(0.245)	(0.244)
Project Age_{mc} Square	0.921***	0.963***	1.059***	1.060***
	(0.321)	(0.317)	(0.219)	(0.219)
CCR		0.416*		0.485 **

		(0.222)		(0.234)
IPLR		0.731*** (0.213)		0.775*** (0.094)
Project Characteristic	Yes	Yes	Yes	Yes
RANDOM EFFECTS				
Intercept	0.097	0.008	0.077	0.010
Individual level	1.487	1.487	1.243	1.245
Log Likelihood	-2750.886	-2743.217	-2600.279	-2592.586

Standard Errors in Parentheses. ***p<0.01, **p<0.05, *p<0.1

The coefficient of *CCR* is positive and significant (0.485, p<0.05), indicating support for Hypothesis 1. In support of Hypothesis 2, we find the coefficient of *IPLR* to be positive and significant (0.775, p <0.01). The coefficient of *Pre-sample CVS* is positive and significant (0.396, p<0.01), which indicates the importance of including a measure of knowledge stocks of the project team at the time of production while estimating team productivity. As expected, *Project Size* has a positive and significant effect (0.090, p < 0.05), indicating that larger human capital is likely to produce more. The coefficient of *Bugs* is positive and significant (0.229, p<0.01). Surprisingly, the coefficients of *Support* and *Page Views* are insignificant. Density of the network is insignificant. *Project Age* has a curvilinear effect on productivity: it becomes harder to produce the software as it grows older. This indicates the importance of controlling for project life cycle effects while estimating project success.

Robustness Checks

The models with group level variables included indicate that the group level variables are able to explain almost all the variance between groups. In such a scenario, GLS estimates should be consistent with the HLM estimates. The GLS estimates confirm the HLM results.

To check for the robustness of my results to outliers, I re-ran all the models after removing the outliers. The results were found to be quite insensitive to outliers. Uzzi and Spiro (2005) found a nonlinear association with clustering, and argued that as clustering increases beyond a certain limit it will have a negative impact on creativity. I also tested this argument by incorporating a square term for *CCR* in my models. The effect of this square term was found to be insignificant. To check for robustness against any misspecification of the dependent variable, I also ran the models using different lags for the dependent variable. Using different lags does not yield qualitatively different results.

Discussion and Limitations

In this article, I investigated how macro level developer collaboration network properties influence the creativity of developers in terms of their productivity. I specifically focus on the small world properties of these large scale networks – clustering and average path length. The theoretical framework suggested that each of these properties of the network structure play a different role in the software development process. I tested these arguments using longitudinal data on a large panel of 2013 projects hosted across 15 different communities at Sourceforge. The empirical analysis provided support for the proposed hypotheses. The results were robust to number of controls and model specifications.

The results are in sync with earlier findings on team effectiveness in OSS. Clustering leads to trust and reciprocity norms, and provides sanctioning capacity to the community. These are essential antecedents for effective team performance in OSS (Bergquist and Ljungberg 2001; Gallivan 2001; Stewart and Gosain 2006). Earlier studies have focused on trust among developers within a project. As clusters are composed of several projects, this study shows that effective trust could also be established by relationships among developers across projects. These between-project relationships act as means of external enforcement.

Given that the embeddedness in small worlds influences project success in an OSS environment, managerial attention should focus on identifying, recruiting, and retaining "gatekeepers" – technical experts who span project boundaries and connect diverse bodies of technical expertise. They are critical to small world structure formation. Formal organizations such as IBM, Apple, Sun Microsystems, and Microsoft have started to show interest in and

support for the OSS movement. Some of these firms have started developing software under an open source license, which gives users access to the source code and allows them to contribute to the code. For such organizations, these technical gatekeepers are valuable resources who can accelerate the process of innovation by synthesizing knowledge from diverse bodies or by simply facilitating information flow from previously unreachable resources.

Though direct comparison with other related research is problematic due to subtle differences in the settings, my results are in sync with much of the theory developed in the area of small world networks (Fleming et al. 2007; Schilling and Phelps 2007; Uzzi and Spiro 2005). The context of this study differs from the above studies in that it provides a much richer environment for knowledge sharing. In more formal setups, such as firm alliances, the participating firms may also be competitors and may be reluctant to share their knowledge not specific to the alliance context (Schilling and Phelps 2007). In contrast, the open source philosophy is based on sharing and complete access to innovation to everyone (Grewal et al. 2006; Xu et al. 2005; Madey et al. 2002; Weber 2004). In the setting of Uzzi and Spiro (2005) and Fleming et al. (2007), the primary source of knowledge spillover is past experiences, which are susceptible to decay. In the context of open source, however, knowledge sharing occurs in real time. Most of the knowledge in the case of OSS is codified and has a largely explicit rather than tacit component. This makes information transfer more efficient (Weber 2004). Developers participating in OSS are motivated by intellectual curiosity and prefer to help others solve innovative problems. This is not the case in more formal settings such as firm alliances and patented technology innovation, where partners may be more strategic in their information sharing since they also compete in the market outside of a partnership and may be bound by contractual agreements prohibiting knowledge sharing outside an alliance (Grewal et al. 2006; Weber 2004).

I do not wish to overstate my results; there are several limitations to this analysis. I follow the small world literature in focusing on the largest component in each foundry, which limits the generalization of my results. Certain characteristics of the projects, such as popularity of the project, may attract more developers, which makes it more likely to belong to the large component. Hence, my results are applicable to projects that belong in the large component and caution must be taken when generalizing them to other projects. I do not include projects that did not show any activity since inception until January 2006, which may lead to sample selection bias. Unfortunately, due to data limitations, I am not able to investigate this. The links between projects across foundries may act as conduits of knowledge flow which I am not able to capture. Although I have used several controls to account for unobserved heterogeneity, some unobserved heterogeneity may still exist, which may introduce bias into the results. I wish to address this problem by use of longitudinal data in future research.

Another limitation is that I do not consider the quality or content of the link or the attributes of the developers. Both these factors could influence the quality of the knowledge flow. Developers may vary in their absorptive capacity for knowledge, and this may influence their project's success. Some projects may share several developers, which may have implications for knowledge sharing. Knowledge from some projects may be more relevant than knowledge from others. Furthermore, knowledge can flow between projects through other channels, such as developer interactions or learning from experience as a user of other software. It is also possible that for certain aspects of knowledge, ties across seemingly diverse technological platforms may provide potentially fertile information. Each of these limitations opens up an exciting area for future research.

Conclusions

I developed and empirically tested a theoretical basis for the impact of macro level properties of developer collaboration on members' project success measured in terms of subsequent code development. This research makes several contributions. First, whereas previous networks research in OSS has examined the impact of a project's network position on its success, this is the first study to analyze the impact of macro level properties of large scale networks on project success (Grewal et al. 2006). Second, it adds to the research on OSS that investigates relational properties among team members by analyzing those issues through a structural sociological lens (Stewart and Gosain 2006). Third, it adds to the research on OSS that investigates macro level structural properties by establishing a relationship between such properties and project success (Xu et al. 2005). Fourth, whereas previous research investigating OSS success has predominantly focused on project characteristics, this research establishes that the social capital of an OSS community is also an important indicator of project success (Chengalur-Smith and Sidarova 2003; Stewart et al. 2006; Lerner and Tirole 2002). Finally, it adds to the emerging small worlds literature by testing the small world phenomenon's impact in a completely new context. Several interesting implications for the OSS community, as well as for firms investing in OSS, are discussed. Limitations of this research are discussed, which open up interesting avenues for future research.

References

Ahuja, G. "Collaboration Networks, Structural Holes, and Innovation," *Administrative Science Quarterly* (45:3), 2000, pp. 425-455.

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

Baker, W.E. "Market Networks and Corporate Behavior," *American Journal of Sociology* (96:3), 1990, pp. 589-625.

Bartlett, F.C. *Remembering: A Study in Experimental and Social Psychology*, Cambridge University Press, London, 1932.

Baum, J.A.C., Shipilov, A.V., and Rowley, T.J. "Where Do Small Worlds Come From?" *Industrial and Corporate Change* (12:4), 2003, pp. 697-725.

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.

Blaauw, G.A., and Brooks, Jr., F.P. *Computer Architecture: Concepts and Evolution*, Addison-Wesley, Reading, Mass., 1997.

Blundell, R.R., Griffith, R., and Van Reenen, J. "Dynamic Count Data Models of Technological Innovation," *The Economic Journal* (105:429), 1995, pp. 333-344.

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

Chandrashekaran, M., Mehta, R., Chandrashekaran, R., and Grewal, R. "Market Motives, Distinctive Capabilities, and Domestic Inertia: A Hybrid Model of Innovation Generation," *Journal of Marketing Research* (36:2), 1999, pp. 95-112.

Chengalur-Smith, S., and Sidorova, A. "Survival of Open Source Projects: A Population Ecology Perspective," in *Proceedings of 24th ICIS*, Seattle, Washington, 2003.

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

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

Cooper, R.G. *Winning at New Products: Accelerating the Process from Idea to Launch*, Persues Publishing, Boulder, Colo., 2001.

Crowston, K., Annabi, H. and Howison, J. "Defining Open Source Software Project Success," in *Proceedings of 24th ICIS*, Seattle, Washington, 2003.

Davis, G.F., Yoo, M., and Baker, W.E. "The Small World of the American Corporate Elite 1982-2001," *Strategic Organization* (1:3), 2003, pp. 301-326.

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

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

Fleming, L., King III, C., and Juda, A. "Small World and Regional Innovation," *Organization Science,* forthcoming 2007.

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

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.

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

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, pp. 1043-1056.

Gulati, R. "Alliances and Network," *Strategic Management Journal* (19:4), 1998, pp. 293-317.

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

Hargadon, A., and Sutton, R.I. "Technology Brokering and Innovation in a Product Development Firm," *Administrative Science Quarterly* (42:4), 1997, pp. 716-749.

Kogut, B., and Walker, G. "The Small World of Germany and the Durability of National Networks," *American Sociological Review* (66:3), 2001, pp. 317-335.

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

Kollock, P. "The Emergence of Exchange Structures: An Experimental Study of Uncertainty, Commitment, and Trust," *American Journal of Sociology* (100:2), 1994, pp. 313-345.

Krackhardt, D. "The Ties That Torture: Simmelian Ties Analysis in Organizations," *Research in Sociology of Organizations* (16), 1999, pp. 183-210.

Krishnamurthy, S. "Cave or Community: An Empirical Examination of 100 Mature Open Source Projects," *First Monday* (7:2), 2002.

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

Lerner, J., and Tirole, J. "Some Simple Economics of Open Source," *Journal of Industrial Economics* (46:2), 2002, pp. 125-156.

MacCormack, A., Rusnak, J., and Baldwin, C.Y. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Prop Code," *Management Science* (52:7), 2006, pp. 1015-1030.

Madey, G., Freeh, V., and Tynan, R. "The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory," *Proceedings of the 8th AMCIS*, Dallas, 2002.

McPherson, M., Smith-Lovin, L., Cook, J. "Birds of a Feather: Homophily in Social Networks," *Annual Review of Sociology* (27), 2001, pp. 415-444.

Milgram, S. "The Small World," *Psychology Today* (1), 1967, pp. 60-67.

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

Narduzzo, A., and Rossi, A. "Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed," 2003. [Available: http://opensource.mit.edu/papers/narduzzorossi.pdf]

Newman, M.E.J., Strogatz, S., and Watts, D.J. "Random Networks with Arbitrary Degree Networks and Their Applications," *Physical Review* (64), 2001, pp. 1-17.

Owen-Smith, J., Powell, W. "Knowledge Networks as Channels and Conduits: The Effects of Spillovers in the Boston Biotechnology Community," Organization Science (15:1), 2004, pp. 5-21.

Raudenbush, S.W., and Bryk, A.S. *Hierarchical Linear Models: Applications and Data Analysis Methods*, Sage, Thousand Oaks, Cal., 2001.

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. 984-999.

Schilling, M.A. "Towards a General Modular Systems Theory and Its Application to Inter-Firm Product Modularity," *Academy of Management Review* (25:2), 2001, pp. 312-334.

Schilling, M.A., and Phelps, C.C. "Interfirm Collaboration Networks: The Impact of Large Scale Network Structure on Firm Innovation," *Management Science* (53:7), 2007, pp. 1113-1126.

Schilling, M.A., Vidal, P., Ployhart, R.E. "Learning by Doing Something Else: Variation, Relatedness, and the Learning Curve," *Management Science* (49:1), 2003, pp. 39-56.

Singh, P.V., and Tan, Y. "Stability and Efficiency of Communications Networks in Open Source Software Development," *Proceedings of the 15th Annual WITS,* Las Vegas, 2005.

Singh, P.V., and Tan, Y. "Planning to First Release: A Conditional Hazard Function Approach for Investigating Open Source Software Development Time," *Proceedings of the 16th Annual WITS*, Milwaukee, 2006.

Snijders, T.A.B, and Bosker, R. *Multilevel Analysis: An Introduction to Basic and Advanced Multilevel Modeling*, Sage, Thousand Oaks, Cal., 1999.

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

Stewart, K. J, Ammeter, T.A., and Maruping, L. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," *Information Systems Research* (17:2), 2006, pp. 126-144.

Ullmann-Margalit, E. *The Emergence of Norms*, Clarendon, Oxford, 1977.

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

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

Von Hippel, E. *The Source of Innovation*, MIT Press, Cambridge, 1988.

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. 209-225.

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

Watts, D.J. "Networks, Dynamics, and the Small World Phenomenon," *American Journal of Sociology* (105:2), 1999, pp. 493-527.

Watts, D.J., and Strogatz, S.H. "Collective Dynamics of 'Small World' Networks," *Nature* (393), 1998, pp. 440-442.

Weber, S. *The Success of Open Source*, Harvard University Press, Cambridge, Mass., 2004.

Wellman, B. "Structural Analysis: From Method and Metaphor to Theory and Substance", in *Social Structures: A Network Approach*, Wellman, B. and Berkowitz, S. (Eds.), Cambridge University Press, Cambridge, 1988, pp. 19-61

Wooldridge, J. *Econometric Analysis to Cross Sectional and Panel Data*, MIT Press, Cambridge, Mass., 2006.

Xu, J., Gao, Y., Christley, S., and Madey, G. "A Topological Analysis of the Open Source Software Development Community," *Proceedings of the 38th HICSS*, Hawaii, 2005.

Yamaguchi, K. "The Flow of Information through Social Networks: Diagonal-free Measures of Inefficiency and the Structural Determinants of Inefficiency," *Social Networks* (16:1), 1994, pp. 57-86.