Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2008 Proceedings

International Conference on Information Systems (ICIS)

2008

Network Effects in OSS Development: The Impact of Users and Developers on Project Performance

Sherae L. Daniel
University of Pittsburgh, daniesr@ucmail.uc.edu

E. Ilana Diamant
University of Pittsburgh, eid3@katz.pitt.edu

NETWORK EFFECTS IN OSS DEVELOPMENT: THE IMPACT OF USERS AND DEVELOPERS ON PROJECT PERFORMANCE

Les effets de réseaux sur le développement des logiciels libres : l'impact des utilisateurs et des développeurs sur la performance des projets

**** Research-in-Progress ****

Sherae L. Daniel
Katz Graduate School of Business
University of Pittsburgh
sldaniel@katz.pitt.edu

E. Ilana Diamant
Katz Graduate School of Business
University of Pittsburgh
eid3@katz.pitt.edu

Abstract

The ability to acquire knowledge is an important determinant of performance for organizations. Developers and users can contribute knowledge to multiple OSS projects, and thereby create links between them through which knowledge can flow and facilitate performance. The contributions a project receives will affect its performance differently depending on the role of the participant and their relationship to other projects. The ability of projects to implement knowledge contributions into code will depend on the level of competition in the knowledge niche in which they exist. We examine how project performance is affected by user- and developer networks, and propose hypotheses relating network density, diversity, and competition to a project's knowledge contributions and implementation.

Résumé

Nous examinons l'impact des réseaux d'utilisateurs et de développeurs sur la performance des projets de développement de logiciels libres. En ce sens, nous formulons un ensemble d'hypothèses reliant la densité du réseau, la diversité, la concurrence, le partage des connaissances et la performance.

Keywords:

Network Effects in OSS Development: the Impact of Users and Developers on Project Performance

** Research-in-Progress **

Introduction

Open Source Software (OSS) has been around for several years (Raymond, 2001). It has attracted the interest of developers but is also becoming popular among less technical users that are adopting OSS systems for cost-saving and security reasons (Nichols and Twidale, 2003). The popularity of OSS can be seen in the increasing variety of applications based on OSS (e.g., MYSQL, PERL, APACHE, LINUX) and in the recent trends of IT corporations to open parts of their code libraries. For instance, Sun has made several Java libraries for mobile devices open-source. Apple has released a software development kit for i-phone applications to developers of third-party applications. As of March 2008, there were 36 i-phone and 53 Facebook application development projects in Sourceforge.

Research Problem

OSS systems and applications are increasingly becoming an alternative to commercial software both in the non-profit and for-profit sector (Wheeler, 2007; Bulkeley, 2003). While there are many successful examples of high-profile OSS (eg., Linux, Perl, MySQL), many OSS projects that share a platform (eg., Sourceforge) stop being active one year after their launch and over 80% of all projects remain inactive (Chengalur-Smith and Sidorova 2003, Stewart et al. 2006). That failure may be due to their inability to get knowledge contributions. Software development is knowledge-intensive; OSS teams need technical knowledge about programming languages and knowledge about user needs. In addition, changes in software development approaches, programming languages, and in the interoperability and security requirements of systems suggest that OSS projects operate in a volatile environment (Iansiti and MacCormack, 1997). For organizations that seek to innovate in volatile environments, the acquisition of knowledge and expertise is critical (Cohen and Levinthal, 1990). Knowledge however is a limited resource for OSS projects because of their dependence on voluntary contributions of developers and users. Without sufficient contributions of knowledge, identifying and solving software bugs and adding functionality can be challenging for OSS teams, and the overall development process in those projects may slow down or stop. Attracting contributions is a challenge because OSS projects depend on volunteer labor (von Hippel and von Krogh, 2003) and because OSS projects compete for the attention, time and effort of participants who frequently have a range of projects they can use or help develop (e.g., there are about 15,000 projects within the MUD game category listed in Sourceforge, as of 04/2008).

Research Approach

OSS projects that share a platform can serve as knowledge reserves for each other so that knowledge contributed to a given project can be reused, recombined or drawn upon and be of use to other projects in the platform. OSS networks where projects are the nodes and developers are the relationships among projects have been shown to facilitate project success (Grewal et al, 2006; Singh et al, 2007). In addition to developers, users can also serve as channels through which knowledge flows across projects. Users bring knowledge to OSS projects by reporting bugs, suggesting new features, editing documentation, and generally commenting on the software's usability (Eklund et al., 2002; Nichols and Twidale, 2003). OSS projects tend to have substantial user communities: for instance, von Krogh et al. (2003) note that in the Freenet project, 356 individuals participated on the discussion list, while there were only 30 developers. Likewise, in a study of Apache, Mockus et al. (2000) find that 3,000 people contributed problems with the software while 400 people developed code. Knowledge contributions from users can help developers refine their coding and improve the software's usability. User involvement has been shown to improve the process and outcomes of software development (Hartwick and Barki, 1994, 2001). Users' contributions also helps sustain the developers' interest in a project: for instance, Nickell (2001) observed that developers perceive "a user-base to be a motivating factor in developing applications". For these reasons we explore the effects of both developer- and user project networks on project performance. We develop hypotheses relating knowledge contributions to an OSS project's knowledge implementation capability. Knowledge contribution is defined as the submissions of knowledge (expertise and/or information) to OSS projects by developers or users; knowledge

contributions can be in the form of code and can also include bug reports, feature requests, discussion forum posts with comments and suggestions about improving the software's usability, enhancing its performance, and increasing its compatibility with other applications. Knowledge implementation is defined as the coding and building of the software that developers perform in any given OSS project.

We adopted a network approach because it allows us to assess the extent to which a project can benefit from the knowledge that is available in other projects, and because it allows us to examine the differential contributions of users and developers to project performance. We expect a difference because developers and users are distinct in their priorities, patterns of contributions and associated network structures (Berdou, 2007). In the networks, the nodes are the projects and the links are the developer or user relationships. A developer relationship between two projects exists when a developer contributes to both projects; a user relationship between two projects exists when a user contributes to both projects. To examine the relationships between project networks and knowledge contributions and implementation, we pose the following research question:

RQ: How does the structure of a project's network defined by developer- or user relationships affect that project's ability to acquire and implement knowledge?

OSS Project Networks

We argue that the structure of a project's network affects its ability to acquire and implement knowledge (Burt, 1992; Coleman, 1988). The type of relationship (developer, user) and the structure of a project's relationships affect the amount and kinds of knowledge to which a project has access, and its ability to implement that knowledge into code modules. Taking the perspective that project-relevant knowledge exists in the projects' relationships, we examine the effects of a project's ego network structure on a project's ability to acquire and implement knowledge towards the development of software. A project's ego network is the set of projects (alters) with which that project (focal) has direct ties. A direct tie is formed when a developer (user) contributes to the focal project and another project. We explore two characteristics of a project's ego network structure: density and diversity.

Network Density

The density of a project's ego-network is determined by the presence of direct ties. The greater the number of direct ties that are present out of all possible ties in a project's ego-network, the denser that ego-network. A dense ego-network can facilitate the amount and speed by which knowledge can reach a project from its alters (Burt, 1992).Access to knowledge enables the generation of alternative solutions to a problem, and stimulates consideration of approaches that have been tried in similar situations. In the case of OSS projects, project teams that can access solutions that have been tried, adapted and applied in the development tasks of other projects can get development-related knowledge from those projects.

Developers

Developer participation involves collaboration and coordination of the software development process. It involves person-to-person communicative actions which have been shown to facilitate the development of shared mental models of the software and the coordination of the development process (Espinoza, 2001, 2007). Greater density in a project's developer network can help developers build shared mental models and also reflects the presence of shared norms and trust among the developers of a focal project and its alters. Shared mental models, norms and trust increase the effort that developers put into a task (Stewart and Gosain, 2006) and encourage the exchange of expertise (Boh et al, 2007). Developers are also likely to contribute to projects that have similar programming languages because the learning barriers for contributing to multiple projects are lower when they involve similar programming expertise. A dense developer network then will likely include developers that have been exposed to similar software problems; as a result, the learning barriers for contributing to multiple projects will be lower, and the exchange of developer knowledge among those projects will be higher. Projects with dense developer networks will therefore tend to receive more contributions from their developers.

H1: The density of a project's developer network will be positively associated with developer knowledge contributions.

Users

While the developers' participation involves person to person communicative actions, users generally submit feature requests and bug reports individually to the development team. This mode of communication is person-to-group and is less interactive compared to person-to-person (interpersonal) mode because it can take place without the users interacting with each other in order to contribute ideas (e.g., Hollingshead and Bonito, 1998, Bonito, 1996). Interpersonal communication seems to be critical to idea generation: individuals tend to generate ideas based on things they have discussed or articulated with others; discussion helps formulate and refine a thought into an idea that is actionable or implementable (eg., Obstfeld 2007, Burt 1992).Users are submitting contributions to projects without having to communicate and collaborate in order to submit their contributions. That can hamper their ability to recognize opportunities that an application can be enhanced and improved upon and to formulate concrete and specific suggestions for desired features. The limited person-to-person communication among users will not benefit projects even when their user networks are dense. Instead it might render the denser user connections less valuable. Denser user networks reflect users' exposure to the same or a similar set of software applications and as a result, to a less diverse set of software features and usability problems. Less diverse exposure to usability problems will limit users' ability to recognize opportunities for improving an application. Also, given the limited person-to-person communication among users in the software platform, their opportunities to interact with other users and refine ideas into concrete and specific conceptualizations of features to recommend to the developer team are also limited. Those two factors, narrower exposure to usability problems and limited person-to-person communication with other users stand to limit the benefits of dense user networks. Projects with denser user connections will thus receive fewer user knowledge contributions.

H2: The density of a project's user network will be negatively associated with user knowledge contributions.

Network Diversity

Network diversity is defined as the structure of a focal project's ties to other projects that allows the focal project to span multiple projects without creating redundant ties. Diversity is reflected in a network's structural holes, which are gaps between nodes in a social network (Burt 1992). Projects that have ties with a relatively disconnected set of projects create structural holes in their networks. The presence of structural holes generates "information benefits" (Burt, 1992). The benefit of the structural holes is in the greater diversity of the knowledge pools from which expertise can be drawn and applied to software development.

Developers

OSS projects that have diverse developer networks are drawing software development expertise from a greater variety of knowledge pools. However, because those projects involve different software problems, programming languages and potentially different user needs, their shared developers will have to expend greater effort in multi-tasking across diverse projects. This is because there are learning barriers when projects use different programming languages, as an example. Even when the learning barriers across diverse projects are low and developers are highly adept at overcoming them, their .attention and effort are limited cognitive resources; when working on a variety of knowledge-intensive problems, individuals tend to spend significant amount of their resources 'switching gears' across problems (Louis and Sutton, 1991) which limits the attention and time they can devote to generating solutions to those problems. Projects with diverse developer networks will tend to draw developers that spread their cognitive resources on multiple and diverse problems across those projects; as a result their contributions to any single project will be limited as the diversity of projects on which they're working increases.

H3: The diversity of a project's developer network will be positively associated with developer knowledge contributions.

User Network Diversity

The limitations of dense user networks can be overcome by increasing the diversity of the user networks of OSS projects. Structural holes in a project's user network can be beneficial because the value of the user network is the innovative development ideas they carry. Projects with a diverse user network have access to a diverse pool of experience with other OSS projects which improves their innovation potential (Reagans and Zuckerman, 2001). Exposure to a variety of software applications, and to a variety of functionality that has been implemented in other

projects can be used to inform the development group of potential features. Suggestions can include ideas about software features, add-ons and usability enhancements. Projects whose user network has structural holes are more likely to receive knowledge contributions from those users.

H4: The diversity of a project's user network will be positively associated with user knowledge contributions.

Knowledge Implementation

There are two kinds of knowledge that go into creative tasks such as software development: 'awareness' knowledge, which has to do with identifying problems and missing features in the software, and 'how-to' knowledge, which reflects the ability to implement solutions to the identified problems (Tornatzky and Fleischer, 1990). 'Awareness' knowledge can come from both developers and users; however, the content of that knowledge will tend to differ because those two groups tend to have different expertise and value different things in a system (Nichols and Twidale, 2003). The potential expertise differences between developers and users, and the experimental nature of the projects imply that the knowledge contributed by the developers and users will likely be different. To capture the impact of both types of knowledge in greater precision we assess them separately (knowledge contributions from developers, knowledge contributions from users).

Users

Groups performing highly creative tasks tend to perform better when they get inputs from a diverse set of participants (Nemeth, 1986). Inputs from diverse sources function as sense-making prompts that facilitate the generation of novel insights and solutions (Nemeth, 1986; Levine and Resnick, 1993). User inputs to software that is under development can function similarly, prompting the developers to experience a cognitive challenge to what they already know about a problem and generate alternatives about possible solutions. The more inputs from users that a project acquires the more software solutions its development team will be able to implement.

H5: Knowledge contributions from users will be positively associated with a project's knowledge implementation.

Developers

While users focus on usability, developers are likely to focus on code quality and technical performance. OSS projects, being non-commercial, tend to encourage experimentation in terms of the code: developers enjoy participation because they can gain knowledge by building highly experimental software and might place less emphasis on usability. Ideas from developers can offer opportunities for development skill refinement. Also, the more knowledge developers contribute to a project the more they can implement it into software solutions because contributing their own ideas helps build shared norms (Stewart and Gosain, 2006). Shared norms facilitate the coordination of the implementation of ideas. Specifically, when there are shared norms developer contributions are phrased in terms that can make it easy to turn suggestions into features. Projects whose developers contribute more suggestions and ideas for the software will be more likely to implement those ideas

H6: Knowledge contributions from developers will be positively associated with a project's knowledge implementation.

Competition

Knowledge inputs that go into the implementation of software are not drawn from a dedicated team but rather, developers and users volunteer their suggestions, bugs, and code to OSS projects. Projects are therefore competing for their developers' and users' attention, time and effort which are limited resources. We consider the boundaries of the project-to-project competition to be the programming language that OSS projects use. A programming language can be conceptualized as a knowledge area that involves a distinct set of programming- and usability skills and expertise. In the terminology of organization theory, a knowledge area is a competitive niche that includes entities which depend on and compete for the same set of resources (Hansen and Haas, 2001; Podolny et al, 1996; Hannan and Freeman, 1977). OSS projects using the same programming language can be considered as occupying a competitive knowledge niche because they depend on the voluntary contributions of developers and users with programming skills and usability interests associated with that language. More explicitly, we define a knowledge niche to be a category of software applications that are based on the same programming language. Projects are in the same competitive niche when they use the same type of application/programming language. Projects within a

knowledge niche are competing for similar knowledge resources (similar development skills) and are therefore facing greater competition than projects across different knowledge niches. Within a given niche, the number of entities, or projects, defines the degree to which that niche is "crowded" (Hansen and Haas, 2001). The greater the number of projects in a niche, the more crowded the niche, and the greater the competition among projects inside that niche for the attention, time and effort of developers and users.
Projects that have received development suggestions and ideas still depend on developers' effort to implement them into actual code. The ability of projects to convert knowledge contributions they receive from users and developers into implemented code will depend on the competition they face. Greater competition, reflected in a larger number of projects in a knowledge niche will hamper the projects' ability to implement ideas and suggestions into code. Lower competition, reflected in smaller number of projects in a knowledge niche will increase a project's ability to convert ideas and suggestions it receives into code. Because the knowledge contributions that a project receives can come from users and/or developers, we distinguish between competition for the implementation of the users' contributions, and competition for the implementation of the developers' contributions. We expect competition to diminish the impact of both kinds of contributions (users' and developers') on project performance.

H7a: The effect of developer knowledge contributions on knowledge implementation will be moderated by competition: greater competition dampens the effect of developer knowledge contributions on knowledge implementation.

H7b: The effect of user knowledge contributions on knowledge implementation will be moderated by competition: greater competition dampens the effect of user knowledge contributions on knowledge implementation.

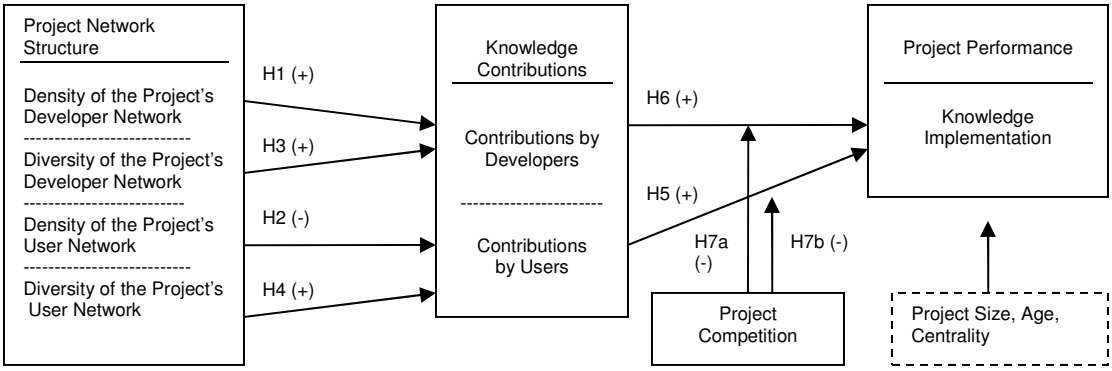

Figure 1. Research Model

Methodology

Constructs and Operationalizations

Construct	Operationalization	Description
Network Density	Density of a project's ego network (developer/user)	The proportion of actual vs. possible ties between a focal project and other projects with which it has a (user- or developer-) direct tie (Coleman, 1988). Formula: D = 2T/[n(n-1)]. T: number of ties among ego and alters, n: number of alters in the ego-network.
Network Diversity	Structural Holes in the Developer/User Network	Extent to which the relationships between the focal project and other projects in the (user or developer) network have structural holes. Measured by the network constraint index (Burt, 1992)

Knowledge Contributions	Number of user (developer) contributions to a project	Number of feature requests, bug reports, posts to the project's discussion forums separated by developers or users.
Competition	Crowdedness of a knowledge niche	Number of projects using the same programming language
Project Performance	Knowledge Implementation	Number of CVS commits in a project
Controls	Project Size	Number of developers and users in a project
	Project Age	Length of time since the project was launched on SourceForge
	Project Centrality*	A project's betweeness centrality score

Sample Description

We randomly sampled projects in the KDE category that were registered in Sourceforge between December 26, 2001 and June 23, 2002. From those we included only those projects that use the CVS repository and that had non-zero code commits. We use these restrictions to limit variation that may be related to the platform used, the registration date or the use of development tools. This yielded a sample of 91 projects used to test the research model. These 91 projects have 216 developers associated with them. The 216 developers made contributions to 383 projects. The 220 users made contributions to 166 projects. 57 (out of the focal 91) projects do not have an ego user network and 34 do. Membership data of each project in the sample was used to create two project-by-project matrices, one based on the users and one based on developers. We plan to follow a sociometric approach which is appropriate for archival-based network analysis. All network analyses will be run in UCINET.

Model Specification

The research model will be tested with the following OLS regression models:

$$KC\_users_{(i)} = \beta_0 + \beta_1 Density_{(user)} + \beta_2 StructHoles_{(user)} + \beta_3 size_i + \beta_4 age_i + \beta_5 Centr_{(user)} + e_{ij}$$

$$KC\_developers_{(i)} = \beta_0 + \beta_1 Density_{(dev)} + \beta_2 StructHoles_{(dev)} + \beta_3 size_i + \beta_4 age_i + \beta_5 Centr_{(dev)} + e_{ij}$$

$$Knowledge\_Implementation_{(i)} = \beta_0 + \beta_1 KC\_users_{(i)} + \beta_2 KC\_developers_{(i)} + \beta_3 KC\_developers_{(i)} \times Tech\_Similarity_{(i)} + \beta_4 size_i + \beta_5 age_i + e_{ij}$$

Conclusion

Our research is limited in that it is not likely to capture the knowledge processes of highly prominent projects, such as Apache, who do not typically use a common development platform like Sourceforge. It is also limited in that we are unable to capture the value of knowledge from sources outside the development platform. Nevertheless, this study makes two significant contributions. First, we examine the impact of network structure on project performance depending on the role of the individual that links the projects. Second, we examine the value that users add to OSS projects. As usability becomes an important determinant of OSS adoption, the role of users and their value to OSS projects will need to be examined in greater detail.

* We will treat centrality as a control rather than a main variable for pragmatic reasons: recommendations to project administrators to increase their project's centrality are hardly practical as they involve 'increasing' the project's connections in an unspecific manner. On the other hand, interventions targeting the project's network density and diversity can be more theoretically grounded and more practical for administrators.

References

Berdou, E. (2007). Managing the Bazaar: Commercialization and Peripheral Participation in Mature, Community-led Free/Open Source Software Projects. Unpublished PhD thesis, London School of Economics. http://opensource.mit.edu/papers/, accessed Aug. 16, 2008.

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

Bulkeley, W. M. "Free Software—Out of the Shadows," *The Wall Street Journal*, Technology (A Special Report), March 31, 2003, p. R6.

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

Bonito, Joseph A. (1996) Topical Contributions to Group Discussions: Assessing the Contribution of Topic Knowledge to Participation. Presented at International Communication Association annual Meeting, Chicago.

Cataldo, M., and Wagstrom, P. A., and Herbsleb, J.,and Carley, K. M . (2006). Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'06)*

Chengalur-Smith and Sidorova. 2003. Survival of Open-Source Projects: A Population Ecology Perspective. *24th International Conference on Information Systems.*

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

Coleman, J. 1988. Social Capital in the Creation of Human Capital. *American Journal of Sociology* 94.

Eklund, S. Feldman, M.,Trombley, M.,and Sinha, R. 2002. "Improving the Usability of Open Source Software: Usability Testing of StarOffice Calc," *Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems (CHI 2002)*

Espinosa, J. A., Kraut, R.E., Slaughter, S. A., Lerch, J. F., Herbsleb, J. D., & Mockus, A. (2001). Shared mental models and coordination in large-scale, distributed software development. In *Proceedings, International Conference in Information Systems*, New Orleans, LA, December 16- 19.

Espinosa, A., Slaughter, S., Kraut, R., & Herbsleb, J. (2007). Familiarity, Complexity and Team Performance in Geographically Distributed Software Development. *Organization Science*, July-August,18, pp. 613 – 630

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

Grewal, R., G.L. Lilien, G. Mallapragada. 2006. Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. *Mgmt. Sci.* 52(7) pp.1043–1056.

Freeman, J. and Hannan, M.T. (1977). *The Population Ecology of Organizations*, American Journal of Sociology, 82, 1977, 929-964.

Hansen, M.T., and Haas, M. R., (2001). Competing for Attention in Knowledge Markets: Electronic Document Dissemination in a Management Consulting Company. *Administrative Science Quarterly*, 46(1), pp. 1-28

Hartwick, J. and Barki, H. 2001."Communication as a Dimension of User Participation," *IEEE Transactions on Professional Communication*, 44, 1,pp.21-36.

Hartwick, J. and Barki, H. 1994."Explaining the Role of User Participation in Information System Use," *Management Science*, 40, 4, pp.440-465.

Hollingshead, A. and Bonito, J. A. 1998. Participation in Small Groups. Communication Yearbook, 20. pp. 227-261.

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

Krackhardt, D. 1990 Assessing the political landscape: Structure, cognition, and power in organizations. *Administrative Science Quarterly*, 35: 342–369.

Levine, J., & Resnick, L. (1993). Social foundations of cognition. Annual Review of Psychology, 44, 585–612

Louis, M.R. and Sutton, R.I. (1991) Switching cognitive gears: From habits of mind to active thinking. Human Relations, 44:55-76.

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

Nemeth, C. (1986). Differential contributions of majority and minority influence. Psychological Review, 93, 23–32

Nichols, D. and Twidale, M.B (2003). The Usability of Open Source Software. *First Monday*, 8(1) URL: http://firstmonday.org/issues/issue8_1/nichols/index.html

Nickell, S. 2001. "Why GNOME Hackers Should Care about Usability," GNOME Usability Project, at http://developer.gnome.org/projects/gup/articles/why_care/, accessed 16 April 2008.

Obstfeld, D. (2007).Riffing*: The* Multivocality of Innovative Action. *Paper presented at INFORMS Conference,* Nov. 2007, Pittsburgh, PA.

Podolny, J. M., Stuart , T.E., and Hannan, M.T.(1996). Networks, Knowledge, and Niches: Competition in the Worldwide Semiconductor Industry, 1984-1991. *The American Journal of Sociology*, Vol. 102, No. 3 (Nov., 1996), pp. 659-689.

Raymond, E.S. 2001. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. *O' Reilly*, Sebastopol, CA.

Reagans, R., and E. Zuckerman 2001. Networks, Diversity and Performance:The Social Capital of R&D Units." *Organization Science,* 12: 502–517.

Reagans, R., Argote, L., & Brooks, D. 2005. Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together. *Management Science, 51*(6), 869-881.

Rogers, E. M. 1995. *Diffusion of Innovations* (4th ed.), The Free Press, New York.

Singh, P.V. 2007. Open Source Software and the Small World Phenomenon.*Proceedings of the 28th ICIS*.

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

Tornatzky, L. and M. Fleischer. 1995. *The Processes of Technological Innovation*, Lexington Books.

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

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

Wheeler, D. (2007) "Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers"
http://www.dwheeler.com/oss_fs_why.html#market_share. Accessed on Aug, 18, 2008