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

<table>
<thead>
<tr>
<th>Journal:</th>
<th><em>International Conference on Information Systems 2008</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID:</td>
<td>draft</td>
</tr>
<tr>
<td>Track:</td>
<td>Alternative Systems Development</td>
</tr>
<tr>
<td>Keywords:</td>
<td>OSS/FLOSS, Networks, User involvement/participation</td>
</tr>
</tbody>
</table>

**Abstract:** Knowledge acquisition is a key capability for organizations that seek to innovate, and a strong social network can facilitate it. Open source software (OSS) developer networks have been associated with project success. Users also participate in multiple OSS projects, however the users' participation is typically focused on usability while the developers' participation is focused on coding. We suggest that users can serve as a channel through which knowledge, especially usability-related knowledge, can move between projects. We propose distinct hypotheses for the effect of the network characteristics of user- and developer networks on project performance. The effect of network defined by user relationships is expected to be different from the effect of the network defined by the developers. This research is among the first to suggest that the impact of network characteristics on project success differs with the role of the participant who represents the relationship.
Network Effects in OSS Development: the Impact of Users and Developers on Project Performance

Abstract

Knowledge acquisition is a key capability for organizations that seek to innovate, and a strong social network can facilitate it. Open source software (OSS) developer networks have been associated with project success. Users also participate in multiple OSS projects, however the users’ participation is typically focused on usability while the developers’ participation is focused on coding. We suggest that users can serve as a channel through which knowledge, especially usability-related knowledge, can move between projects. We propose distinct hypotheses for the effect of the network characteristics of user- and developer networks on project performance. The effect of network defined by user relationships is expected to be different from the effect of the network defined by the developers. This research is among the first to suggest that the impact of network characteristics on project success differs with the role of the participant who represents the relationship.

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 that 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. While there are many successful examples, many OSS projects do not survive (Stewart et al. 2006). To understand how projects can be successful, we develop and test a theoretical model explaining how the user and developer networks among OSS projects enhance the absorptive capacity of OSS projects.

Research Problem
Software development is knowledge-intensive; OSS teams need technical knowledge about programming languages and knowledge about user needs. Many OSS projects fail to get knowledge contributions necessary to survive. Without sufficient contributions, 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 are not the only organizations that seek to innovate and require knowledge to be successful. For organizations that seek to innovate in volatile environments the absorptive capacity framework suggests acquiring knowledge is a critical capability. Absorptive capacity is the ability of an organization to acquire and use knowledge (Cohen and Levinthal, 1990). Changes in software development approaches, programming languages, and the interoperability of systems suggest that software development is a volatile environment (Iansiti and MacCormack, 1997). Because an OSS project is a type of organization for which knowledge acquisition is important we adopt the absorptive capacity framework to develop hypotheses relating an OSS project’s knowledge acquisition capability to its knowledge implementation capability. Knowledge acquisition is a project’s ability to acquire knowledge contributions. Contributions in OSS projects are in the form of code but can also include bug reports, feature requests, forum posts with comments and suggestions about improving the software’s usability, enhancing its performance, and increasing its compatibility with other applications. We conceptualize contributions to OSS projects as “knowledge contributions” to capture the fact that they can include code and/or feedback. Knowledge implementation is the coding and building of the software that developers perform.

OSS projects that share a platform can serve as knowledge reserves for each other. They can have useful development knowledge that other projects on the platform need. Therefore understanding how a project is related to other projects can be used to understand its ability to acquire knowledge. OSS project networks where projects are the nodes and developers participating on multiple projects are the relationships among projects have been shown to facilitate project success (Grewal et al, 2006; Singh et al, 2007). Developers who participate on multiple projects act as a channel through which knowledge form one project can get to another project. They bring knowledge to a project in the form of code and in the form of development suggestions.
Users can also contribute to multiple projects and we argue that they can serve as channels through which knowledge can flow 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). User participation 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 knowledge contributions.

The network approach allows us to assess the extent to which a project can benefit from the knowledge that is available in the other projects. More specifically, the network structure of projects affects their ability to acquire and implement knowledge. We make a distinction between the developer- and the user network because the knowledge contributions of those two participant types may result in distinct project network structures, which in turn might affect the projects’ knowledge acquisition differently. 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 acquisition 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 social network consisting of relationships with projects in its 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 can affect the amount and kinds of knowledge to which a project has access, and the process by which knowledge is implemented into existing modules. Taking the perspective that project-relevant knowledge exists in the projects’ relationships, we examine the effects of project-specific network structures on a project’s ability to acquire and implement knowledge towards the development of software. In particular, we explore two characteristics of network structure; density and diversity.

**Project Network Density**

A dense network can facilitate the amount and speed by which knowledge can reach a project from sources within its network (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.

**User Network Density**

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 because it can take place without the users interacting with anyone in a person-to-person mode (e.g., Hollingshead and Bonito, 1998, Bonito, 1996). For this reason projects with greater user network density whose members communicate in a person-to-group mode do not necessarily imply greater familiarity, experience and trust (e.g., Espinoza et al, 2001, 2007). This is because the users are not communicating with each other; rather they are submitting contributions to projects without having to collaborate in order to submit their contributions. This can minimize the traditional benefits of a dense network that depend on the development of familiarity, experience and trust. Also, given the limited interpersonal communication among users in the software platform, their opportunities to interact with other users and refine, reformulate
and/or generate ideas for software features are also limited. In addition, greater density in OSS project user networks might mean that users have been exposed to the same or a similar set of software solutions and problems across multiple projects. This would limit the features that they are able to request. As a result, projects with higher density user networks are likely to receive fewer knowledge contributions from users.

**H1: Projects with greater density in their user network will have lower user knowledge acquisition.**

**Developer Network Density**

While the users’ participation involves person-to-group communicative actions (e.g., sending a bug report), the developers’ participation involves to a large extent collaboration and coordination of the development process. Those are person-to-person communicative actions and have been shown to facilitate the development of shared mental models of the software and the coordination the development process itself (Espinoza, 2001, 2007). Greater density in a project’s developer network helps developers build shared mental models and also reflects the presence of shared norms and trust among the developers of a focal project and the other projects with which the focal project shares developers. 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 skills. 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.

**H2: Projects with greater density in their developer network will have greater developer knowledge acquisition.**

**Project Network Diversity**

In network terms, 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 tasks.

**User Network Diversity**

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.

**H3: Projects with greater diversity in their user network will have greater user knowledge acquisition.**

**Developer Network Diversity**

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.

**H4:** Projects with greater diversity in their developer network will have lower developer knowledge acquisition.

**Knowledge Implementation**

There are two kinds of knowledge that go into creative tasks such as software development: ‘awareness’ knowledge that has to do with identifying problems and missing features in the software and ‘how-to’ knowledge that 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. The potential expertise gaps 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 their impact on knowledge implementation in greater precision we assess them separately (knowledge acquisition from developers, knowledge acquisition 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 under development can function similarly, prompting the developers of that project to experience a cognitive challenge to what they already know about a given problem and generate alternatives about possible solutions. The more inputs from users a project acquires the more software solutions its development team will be able to implement.

**H5:** Greater knowledge acquisition from users will increase 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:** Greater knowledge acquisition from developers will increase a project's knowledge implementation.

**Competition**

Knowledge in the form of software development expertise is a limited resource. Knowledge inputs that go into the implementation of software solutions are not drawn from a dedicated group of developers, but rather, a set that contribute to multiple projects whenever they choose. Projects are therefore competing for developers’ inputs to build systems that meet the changing trends in software development and user interests. The more intense the competitive environment in which a project operates, the more consequential its knowledge acquisition will be. The ability to acquire knowledge is more consequential in highly competitive environments because the rapid changes in technology spurred by competition produce greater demand for new skills of software development. Projects with greater ability to acquire knowledge are more responsive to technological changes, which makes them more competitive than projects with lower knowledge acquisition capabilities. Knowledge acquisition confers competitive advantage because it enables a project to replenish its knowledge resources.
**H7**: The effect of developer knowledge acquisition on knowledge implementation is moderated by competition: greater competition amplifies the effect of developer knowledge acquisition on knowledge implementation.

![Figure 1. Research Model](image)

**Methodology**

**Constructs and Operationalizations**

<table>
<thead>
<tr>
<th>Construct</th>
<th>Operationalization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Density</strong></td>
<td>Density of a project’s developer ties</td>
<td>Average strength of a project’s ties in its immediate neighborhood. Measured by the density of a project’s ego network.</td>
</tr>
<tr>
<td><strong>Network Diversity</strong></td>
<td>Structural Holes</td>
<td>Extent to which the relationships between project p(i) and other projects in the user network have structural holes. Measured by the network constraint index.</td>
</tr>
<tr>
<td><strong>Absorptive Capacity</strong></td>
<td>Knowledge Acquisition From Users</td>
<td>Number of user contributions to a project (feature requests and posts to a project’s discussion forums).</td>
</tr>
<tr>
<td></td>
<td>Knowledge Acquisition From Developers</td>
<td>Number of developer contributions to a project (feature requests and posts to a project’s discussion forums).</td>
</tr>
<tr>
<td><strong>Competition</strong></td>
<td>Structural Equivalence</td>
<td>Similarity of a project’s pattern of ties to those of other projects</td>
</tr>
<tr>
<td><strong>Project Performance</strong></td>
<td>Knowledge Implementation</td>
<td>Number of CVS commits in a project</td>
</tr>
<tr>
<td><strong>Controls</strong></td>
<td>Project Size</td>
<td>Number of developers and users in a project</td>
</tr>
<tr>
<td></td>
<td>Project Age</td>
<td>Length of time since the project was launched</td>
</tr>
<tr>
<td></td>
<td>Project Centrality*</td>
<td>A project’s betweeness centrality score</td>
</tr>
</tbody>
</table>

*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.*
The network analysis will be done following a sociometric approach which is appropriate for archival-based network analysis. 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. Cells in the two matrices represent user- and developer-relationships between projects, respectively: a relationship between two projects reflects the fact that the same participant has made at least one contribution to both projects. Consistent with network analysis conventions, each network variable was assessed with a single indicator (Krackhardt 1990). All network analyses will be run in UCINET. We plan to have preliminary results for presentation in December 2008.

**Model Specification**

We will test our research model with the following OLS regression models:

\[
\begin{align*}
KA_{\text{users}}(i) &= \beta_0 + \beta_1 \text{Density}_{(\text{user})} + \beta_2 \text{StructHoles}_{(\text{user})} + \beta_3 \text{size}_i + \beta_4 \text{age}_i + \beta_5 \text{Centr}_{(\text{user})} + e_{ij} \\
KA_{\text{developers}}(i) &= \beta_0 + \beta_1 \text{Density}_{(\text{dev})} + \beta_2 \text{StructHoles}_{(\text{dev})} + \beta_3 \text{size}_i + \beta_4 \text{age}_i + \beta_5 \text{Centr}_{(\text{dev})} + e_{ij} \\
\text{Knowledge\_Implementation}(i) &= \beta_0 + \beta_1 \text{KA\_users}(i) + \beta_2 \text{KA\_developers}(i) + \beta_3 \text{KA\_developers}(i) \times \text{Struct\_Equivalence}(i) + \beta_4 \text{size}_i + \beta_5 \text{age}_i + e_{ij}
\end{align*}
\]

**Conclusion**

Our research will be 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 also is limited in that we are unable to capture knowledge about desired features that users gain from using applications outside the development platform. Nevertheless, we make two important contributions to theory. Our first contribution is in suggesting that the impact of social network structure on project performance depends on the role of the individual that links the projects. Because the contribution behavior of users and developers differ, their roles need to be examined separately. Our second contribution is in examining the knowledge benefits that OSS projects can derive through their user network. While previous studies have focused on the developers’ contributions, our study is among the first to explore the value that users add to OSS projects. As the role of users becomes more important for organizations that seek to innovate with open-source software, it will become important to identify how projects benefit from their user ties and from the knowledge that those ties bring to projects.
References


