3. Analysis of Survival of Open Source Projects: a Social Network Perspective

Jing Wu  
National University of Singapore  
wujing@comp.nus.edu.sg

Qian Tang  
National University of Singapore  
ctang@comp.nus.edu.sg

Abstract
The popularity of open source is unassailable. It has been widely adopted for different purposes, including web servers, e-mail servers, operating systems, and programming languages. Although some open source projects, like Apache, Sendmail and Linux, have achieved extraordinary success, lots of projects are lackluster or with no developing activity at all. To deepen our understanding of open source projects, it is essential to explore the factors that have contributed to its success or failure. The objective of this research is to analyze the survival of open source projects from the social network perspective. In particular, we have studied the impact of the communication pattern of open source projects on the project success, defined in view of both the supply side (developers) and the demand side (end users), while incorporating control factors such as project-specific characteristics. Based on empirical data collected from SourceForge.net, a popular open source hosting web site, our results show that the communication pattern has significant impacts on the open source success.

Keywords: Open Source, Communication Pattern, Social Network Theory, Success

Introduction
Open source software (OSS) refer to those programs “whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or the modified program without having to pay royalties to previous developers” (Wheeler 2003). OSS involves a copyright-based license to keep private intellectual property claims out of the way of both software innovators and software adopter, while preserving a commons of software code that everyone can access (O’Mahony 2003). It is typically created within OSS projects, often initiated by an individual or a group that wants to develop a software product to meet their own needs.

Ever since the first OSS was developed by Richard Stallman (GNU) in the 1970’s, a myriad of open source applications have been developed, ranging from office productivity such as StarOffice, to database (mySQL) and thousands of specialized scientific applications. Nowadays, OSS has been widely adopted for different purposes, including web servers (Apache, iPlanet/Netscape), e-mail servers (Sendmail), programming languages (Perl, Java, Python, GCC, Tk/TCL), and operating systems (Linux, BSD Unix). More than 65 percent of public websites are backed by the open-source Apache web server; 80 percent of the world’s e-mail traffic is managed by Sendmail; and nearly 40 percent of large American corporations make use of the open-source GNU/Linux operating system (Weber 2004).

Various case studies have contributed to a better understanding of the OSS phenomenon. Hertel et al. (2003) focused on factors determining the level of engagement in the Linux project. Krogh et al. (2003) analyzed the strategic process by which new individuals joined the community of developers of FreeNet, a peer-to-peer network of information distribution.
Lakhani and von Hippel (2003) considered the nature and the functioning of the community of developers of the Apache software. These studies shed new light on how large communities of developers arise, work and coordinate to achieve the success of an open source project. However, previous case studies are limited to large and popular projects only. While a closer look on such large and popular projects is crucial to better understand how communities work effectively, they may not be sufficient enough to represent the whole open source world.

Several large open source projects have achieved extraordinary success and are among the most prominent software used in the technology industry. However, many open source projects have been lackluster or with no active development activity at all. Many die at the beginning, while others survive, but with little momentum behind them (Thomas and Hunt 2004). Growing up in public can be hard to do! The failure of large number of open source projects begs several key questions: How is OSS dealing with growing pains? What is its long-term sustainability? Why do some open source projects achieve success while some others do not? What are the factors that could influence the success or failure of the open source projects? To deepen our understanding of the OSS, it is essential to explore the factors that have contributed to its success or failure.

Open source projects represent virtual teams collaborating over the Internet with an aim to produce free software. Such virtual team has been creating value through developing and spreading new knowledge and capabilities, fostering innovations, building and testing trust in working relations, relying heavily on information and communication technologies to accomplish their tasks (Powell et al 2004). For virtual teams to achieve their objectives and successfully complete their task, information must be effectively exchanged. Thus, communication and coordination have been found to be two major aspects that significantly affect the performance of virtual teams (Johansson et al. 1999; Maznevski and Chudoba 2001). OSS development is a complex socio-technical activity, requiring people to interact with each other. Thus, it’s interesting to study the communication pattern of open source development teams to find out the importance of coordination and communication and further uncover the association between success and communication pattern characteristics (social network attributes).

While others have studied the determinants of open source success (e.g. Fershtman and Gandal 2004; Comino et al. 2005; Sen 2005; Colazo et al. 2005; Stewart et al. 2006; Grewal et al. 2006), this paper is among the first to explore open source success through the lens of social network perspective. Based on empirical data, we study the impact of the communication pattern of open source projects on the project success, as well as considering project-specific characteristics. The success of open source is determined from both supply and demand sides. The main objective is to identify the presence and significance of factors in predicting the success of an OSS project. We collect data from SourceForge.net, the largest repository of open source projects, which is widely used in most OSS studies.

This paper is organized as follows. Section 2 introduces the theoretical background of communication pattern and explains why and how it can be applied to the open source study. We provide definition of key concepts such as open source success and communication pattern. Then we propose the research model and the hypotheses with respect to the impact of communication pattern of project teams on the success of open source projects in Section 3. We describe the operational details of our empirical research, such as criteria for project
selection and measures of constructs in Section 4, followed by discussions of the results. Finally, Section 5 concludes this paper with directions of future research.

**Theoretical Background**

In this study, we propose that the social structure of the open source project teams may play a critical role in the success of open source projects. Based on social network theory, we investigate the interactive communications among open source contributors in order to find the impact of communication pattern on open source success. In this section, we define key concepts such as success, social structure, social network analysis, and communication pattern in the open source environment.

**Communication Pattern of Open Source Project Teams**

Open source development is one of the prime manifestations of virtual teams collaborating over the Internet. Such virtual teams create value through developing and spreading new knowledge and capabilities, fostering innovations, and building and testing trust in working relations (Kidane and Gloor 2007). The advent of information and communication technologies provides instantaneous global accessibility for the open source community. Software development is a complex socio-technical activity. The developers of an open source project collaborate via interactions or communications in forms of email exchange, message boards, etc. (Steve 2004). The social aspect of software development deals with how people interact, behave and organize in the community. The communication pattern among individuals and groups can be applied to describe the network of relationships within the project team. Specifically, it helps to identify the prominent patterns in such networks, trace the flow of information (and other resources), and discover the relationship between such social network structure and the final product i.e. the software system (Kidane and Gloor 2007). Thus, from the social network perspective, we study the open source success by analyzing the impact of communication pattern of project team.

Communication pattern describes the structure of interactions during communication. Social structure, a term frequently used in social theory, refers to entities or groups in definite relation to each other, to relatively enduring patterns of behavior and relationship within social systems (Scott 2002). Investigating social structure is a useful way to understand team practice such as coordination, control, socialization, continuity and learning (Freeman 1979; Scacchi 2002). Software engineers have realized that there are inevitable linkage between the group performance and the social structure of the development team. Therefore, a better understanding of the social structure can help with the development planning (Scacchi 2002). Crowston and Howison (2005) interviewed a member of the Apache Foundation’s incubator team at ApacheCon 2003. The incubator team indicated that they were concerned that overly heavy reliance on a small number of (possibly corporate funded) developers was a major threat to the sustainability of the project and thus to the suitability of the project for Apache incubation (Crowston and Howison 2005). The study of social structure is helpful in understanding the reasons for such concerns since it provides an assessment measure of finding the crucial member of the projects as well as how important each developer is in the project.

To best understand interaction in the social structure, especially the developers’ communication pattern in the open source project, we employ the theory of social network analysis (SNA). SNA (also called social network theory) has emerged as a key technique in many fields such as sociology, anthropology, statistics, mathematics, information sciences, education, and psychology. SNA aims to understand the relationships between people,
groups, organizations, and other types of social entities (Granovetter 1973; Wasserman et al. 1994; Wellman et al. 1998) by description, visualization, and statistical modeling. It models social relationships in terms of nodes and ties. Nodes represent the individual actors or groups within the network, and ties or links show interactions or exchange of information flows between the nodes. In the context of open source projects, nodes are the developers, and ties are the interactions (i.e. communications) between the developers.

The communication pattern of a project team can be characterized by several attributes. Social network theory suggests that the centrality and density of a group are related to its efficiency of problem solving, perception of leadership and the personal satisfaction of participants (Scott 2002). Thus, centrality and density are considered in this study. The concepts of density and centrality refer to differing aspects of the overall “compactness” of the network (Scott 2002). Density describes the general level of cohesion in the network; centrality describes the extent to which this cohesion is organized around particular focal points. Centrality and density, therefore, are important complementary measures (Scott 2002). Density measures how closely a network is connected, which in turn determines the readiness of a group in response to changes. It is defined as the percentage of ties that exist in a network out of all possible ties.

Centrality can be defined on an individual or overall level for a network. The centrality of an individual node refers to the number of direct links to other nodes in a network. If we define the link between nodes as communications, a person with a high centrality represents a major channel of information exchange. In some sense he is a focal point of communication, at least with respect to the others with whom he is in contact. At the opposite extreme is a point of low degree. The occupant of such a position is likely to to be seen as peripheral. His position isolates him from direct involvement with most of the others in the network and cuts him off from active participation in the ongoing communication process. Thus, the centrality indicates whether a group member is “in the thick of things” (Freeman 1979; Mullen et al. 1991). One can also define the centrality of a network as a whole. In our study, we do not focus on an individual developer’s centrality, but on centrality of an entire project team. Project centrality is defined as the centrality of a whole project team. It explains the inequality of the developers’ contributions to the project: high score of project centrality implies that the power of individual developers varies rather substantially, and overall, positional advantages are rather unequally distributed in this network. Social network theory (Leavitt 1951) suggested that the speed and efficiency of a network in solving problems should be related to the substantial inequality of the developers’ contributions to the project.

**Success of Open Source Projects**
Apart from licensing terms, OSS has other distinct features than proprietary software. OSS development frequently depends on volunteers coordinating their efforts without the governance of a common organizational employer, and the end product is often provided for free (Feller and Fitzgerald 2000). Therefore, unlike traditional firm-driven endeavors, open source projects are not always driven by direct profit motives (Lakhani and Wolf 2003). The success indicators of commercial software projects such as market share, on time and on budget delivery cannot be readily applied in the OSS setting. With OSS development, usually there is no pre-determined deadline, a priori budget, or set of specifications (Scacchi 2002) and market share of OSS is difficult to calculate. Therefore, different indicators are necessary to define the success of open source projects.
Success is a subjective concept and therefore it is not always clear how to define success. Raymond (1998) defines successful OSS projects as characterized by a continuing process of volunteer developers fixing bugs, adding features and releasing software “often and early”. Given that a large number of OSS projects are abandoned by the developers, it is critical that being able to attract contributors on an ongoing basis to keep the project sustainable (Markus et al. 2000). Crowston et al. (2003) explored success measures in the Information Systems literature and suggested a portfolio of success measures that includes measures of the development process. Subsequently, Crowston et al. (2004) analyzed four success measurements by using the data from SourceForge.net and suggested that a project that attracts developers, maintains a high level of activity, fixes bugs and has many users downloads can described as a success. There are some other scholars advocating different success measurements. For example, Colazo et al. (2005) singled out two particular items from those success measures: the number of developers joining in a project and the relative level of the developers’ productivity while they were engaged in the project (i.e. contribution). Comino et al. (2005) utilized the development stage (i.e. planning, pre-alpha, alpha, beta, stable and mature) of a project as the representation of the level of success of a project. Fershtman and Gandal (2004) considered an alternative definition of system success based on output per contributor; examined how the type of license, the programming language, the intended audience and other factors affect the output per contributor in OSS projects. Sen (2005) made use of project popularity (defined by Freshmeat.net) as the measure for OSS’s installation base. Stewart et al. (2006) adopted user interest as the measurement of OSS project success. In particular, they used the development activity to measure the development-oriented success. Grewal et al. (2006) adopted two kinds of success measures: the number of CVS commits as an indicator of successful technical refinement; and the number of downloads over the life span of a project as the indicator of market or commercial success.

Since the open source development relies on voluntary input, attracting and motivating contributors are key factors in its success. In our study, we consider success from both “supply side” and “demand side”. Both sides have different criteria to assess project success. For the developers, development activity is a key indicator of the project success: high development activity shows that the developers in the project continuously contribute to the project; the project will die until it has no developing activity at all. For the users, project popularity is a main measure of the project success: high popularity shows that there are many users interested in and tend to use this project; the project will die if nobody makes use of it for a period.

In summary, our research is based on the theoretical fields of social structure, social network analysis, and the measurement of OSS success. To the best of our knowledge, we are among the first to explore the success determinants of open source projects through social network perspective and in particularly, the impact of communication pattern on the success of the open source.

Research Model
This study focuses on an important potential determinant of open source project success: communication pattern of the development team. We thus propose hypotheses with regard to how communication pattern may impact the success of the open source projects. Specifically, we define the following indicators to capture communication pattern of project team: (1) project centrality, which indicates the inequality of the developers’ contributions in the project; and (2) project density, which represents the readiness of the group to respond to
changes and how close a network is to realize its potential. In the meanwhile, both development activity and project popularity are adopted to capture project success from the supply side and demand side respectively. The model is shown in Figure 1.

### 3.1 Communication Pattern and Project Success

Project centrality measures the difference or inequality of contributions among developers in the project, i.e., it examines whether there is an outstanding group of contributors in the project. The investigation of project centrality can shed light on whether the inequality of the developers’ contributions to the projects affects the success of the project. Social network theory suggests that when project centrality is high, the power of individual developers varies rather substantially. Networks with high centrality have the advantage of speedy and flexible information diffusion within the network (Cummings and Cross 2003). The developers, with high centrality, may have access to and be able to call on more of the resources of the network as a whole. They may take charge of exchanging most information or resource among the project team members. In this way, the core developers can filter the useful information and helpful resources, distribute this high quality information among the team members, and discard the less meaningful messages, which increases the efficiency and quality of the communication among the developers. Thus, the project has access to greater resources and gets better information quality. The development process, which involves tasks such as debugging, document writing, upgrading, patching, and consulting, can be better handled with better resource allocation and high quality information. Indeed, research in social networks shows that centrality is an important indicator of group performance (Freeman et al. 1980). Since a project with higher project centrality is better organized and owns more efficient information exchange mechanism, we propose that more centralized project is likely to achieve higher development activity.

In the strategic management literature, the well coordinated development teams, who are able to attract users, resources, and collaborators, are considered as a key determinant of the
software success. In a highly centralized project, the communications are largely dependent on the core developers, who are in charge of exchanging most information or resource among the project team members. Since the open source developers usually simultaneously participate in more than one projects or affiliate to other commercial projects, once they are not interested in the current project or have not enough energy to focus on this project, this project would lose the centre, and the response to the users’ requests would be largely delayed. The communication efficiency will highly reduced by missing this developer. Thus, as the practice indicated, heavily reliance on a single developer threatens the survival of the open source projects. This may largely deter the users who want to seek the continuous support and maintain from the project teams. Therefore, we propose that high centralize project will be less likely to attract more users, and thereby less likely to increase the popularity of the project.

Hypothesis 1: Centralized projects will be more likely to achieve higher development activity.
Hypothesis 2: Centralized projects will be less likely to increase the popularity of the projects.

In software development groups, higher density indicates a greater degree of interaction among the members in the process of making decisions, which facilitate collaboration. However, higher project density makes the dissemination of knowledge taken longer, as it needs to travel through the extended hierarchies. For example, in a three-developer project, if the density is very high, i.e., developer A has communications with B, B with C, and C with A, developer C may obtain the same information from both A and B. In projects with many developers, there may be more repeated information exchanged among developers. The efficiency of the communication, therefore, is largely reduced in high density projects. Previous literature indicates that effective communications among the team members is the key to project success (Suchan and Hayzak 2001). For open source development teams, to achieve the objectives and successfully complete their task, information must be effectively exchanged (Powell et al. 2003). Thus, we propose that density of the project will negatively affect the development activity of the project.

While project density may negatively affect the development activity, it may also positively affect the project popularity. In higher density project, the project’s ability to respond to changes is stronger, since higher density indicates a greater degree of interaction among the members in the process of making decisions. For software users, in case that any developer could not contribute to the project, e.g. the developer could not respond to bug fixing, support or feature request, high density project will not largely affected by missing this developer because other developers will take charge of his responsibility. Therefore, high-density project will attract more users and subsequently increase the popularity of the project.

Hypothesis 3: Project density will negatively affect the project development activity.
Hypothesis 4: Project density will positively affect the project popularity.

Project-Specific Characteristics and Project Success
Apart from the communication pattern, there are several project-specific factors that may also influence the success of the project, such as types of license, project complexity, programming language, project maturity, target audience and financial support.
Project-Specific Characteristics and Development Activity
Lerner and Tirole (2005) suggested that the restrictiveness of license serve to protect the interests of developers by limiting the possibility of exploitation of their contributions by commercial parties. From the viewpoint of the developers, commercialization of the open source project is undesirable. Commercialization may reduce visibility of developers’ contributions and reputation benefits, and it would also drive away the developers: they have the freedom to utilize and customize the software before commercialization; however, they need to pay money for using it and could not customize the software as they want if the software becomes proprietary. Thus, the restrictiveness of license may play a positive impact on development activity. In addition, financial support of the project from donors will help maintain the development activity; software written in more popular programming languages may attract a larger pool of developers, thus in predicting development activity; finally, the development activity is likely to increase with project complexity.

Hypothesis 5A: More complex projects will be likely to achieve higher development activity.
Hypothesis 5B: Projects with financial support will be likely to achieve higher development activity.
Hypothesis 5C: Projects with more restrictive license will be likely to achieve higher development activity.
Hypothesis 5D: Projects with more popular programming language will be likely to achieve higher development activity.

Project-Specific Characteristics and Popularity
From users’ perspective, the restrictiveness of license may limit what a user can do with the software. They may perceive less value of the software, especially among those seeking potential commercial interests. Furthermore, the usefulness of software is dependent on its compatibility with other applications. Thus, more restrictive license may limit the users’ ability to employ the code in conjunction with software distributed under less restrictive license. In addition, the restrictiveness of license may inhibit the exploitation of some cross-application compatibility, consequently reducing the potential gains that would otherwise be realized. Thus, the restrictiveness of license will play a negative impact on popularity of the project. In addition, software written in more popular programming languages may also attract a larger number of users. What’s more, the popularity is likely to increase with project maturity. Finally, some projects are specifically developed for particular user groups. Those targeted at end users may attract more users, and subsequently increase the project popularity.

Hypothesis 6A: More mature projects will be more likely to be popular.
Hypothesis 6B: Projects targeting at end users will be more likely to be popular.
Hypothesis 6C: Projects with restrictive license will be less likely to be popular.
Hypothesis 6D: Projects with more popular programming language will be more likely to be popular.

Research Method
Project Selection
As with most empirical study on open source projects, the data is collected from SourceForge.net, which is the world's largest online repository of open source applications, providing open source developers with useful tools to control and manage software development. At the time of this study (November 2006), SourceForge.net hosted more than 133,029 open source projects on a wide diversity of topics, with more than 1,425,354 registered developers. It also offers a variety of services to hosted projects, including site
hosting, mailing lists, bug tracking, message boards, file archiving, and other project management tools. SourceForge.net provides a large sampling population of open source projects with extensive details, and thereby is the best site to collect data on open source projects’ performance and attributes.

In order to investigate the communication pattern of the project teams, we make the analysis of interactions during the bug fixing process, which involves intensive interactions (Crowston 1997; Raymond 1998). It is the site of claims of effectiveness made for open source projects: a quick response to bugs has been a particular strength of the open source development Therefore, we base the analysis on SourceForge’s bug tracking system which enables users and developers to report and discuss bugs. A bug report includes basic information about the bug that can be followed up with a trail of correspondence. Our program downloads the HTML files containing bug report pages for the selected projects.

Clearly not all projects were suitable for our study: some are in fact individual projects (Krishnamurthy 2002) which are not suitable for analyzing communication pattern of the projects, and some do not make bug reports available or do not use the SourceForge Tracker system, which are the source of communication pattern data. In addition, to better understand and analyze the communication pattern of the projects, we restricted this study to projects that listed more than three developers at the time of selection in January 2007. We chose three developers as a criterion because we are interested in team interactions and communicative interactions, which are not suitable for individual projects. We choose samples from top 1,000 active projects ranked by SourceForge.net. After excluding projects with less than three developers, projects without bug report on SourceForge.net, and projects without bugs, we obtain a final sample size of 56 projects.

Measures
The project success is captured from both supply side – development activity, and demand side – project popularity. Development activity is measured by the number of closed bugs in a fixed period (i.e. one month to the data collection date); popularity is measured by the number of downloads in a fixed period. Both of two measures are reported on SourceForge.net.

The main objective of this study is to investigate the impact of communication pattern of project team on open source success. Project centrality and density are selected to capture communication pattern. Project centrality is measured by project degree centrality, which describes the inequality or variance of developers’ contribution in the network (Freeman 1979). We extract interaction data from SourceForge’s bug tracking system to create a sociomatrix for each project. A sociomatrix is a standard data representation for a network analysis (Wasserman and Faust 1994). The sociomatrix has a row and a column for each individual, and the cells of the matrix count the number of interactions from one individual to another. To obtain the score of project centrality, we analyze each sociomatrix using a popular social network analysis tool: Ucinet 6.0. Density measures the readiness of the group to respond to changes. It is defined as the percentage of ties that exist in a network out of all possible ties. A density of 1 implies that every actor is connected to every other actor. A density of 0 implies that no actor knows any other actor. Project density is also obtained through analyzing sociomatrix using Ucinet.

There are several project-specific factors that may also influence the success of the project. We measure types of licenses according to license categories classified by Lerner and Tirole
A value of 1 indicates the restrictive license; 0 indicates the nonrestrictive license. In addition, a dummy variable is used to specify whether the project has the donors. Project maturity is captured by the age of project which is defined as the difference between the data collection date and the project registration date (in months). Project complexity is measured by number of packages. A dummy variable is included to control for whether the project uses C/C++, which are one of the largest and most popular programming language categories on SourceForge.net. Finally, a dummy variable is employed to indicate whether the project is targeted at end users.

**Results**

Data analysis shows that the average project centrality is around 54.13% and it ranges from 0 to 100%; and the average density is around 20.63% and it ranges from 1.84% to 100%. To directly understand the communication pattern of the project team, we provide the communication pattern graphs of two example projects by using Ucinet draw tool (See Figure 2). Project “Gaim” has the lowest density in the sample projects (centrality: 46.98%; density: 1.84%). Project “FileZilla” has the highest centrality (centrality 100%; density: 22.22%).

![Figure 2: Communication Pattern Graphs of Example Projects](image)

We measure success from both supply and demand side. It is known that the supply and demand would influence each other simultaneously. Thus, we employ three-stage-least-square (3SLS) method to analyze the data by Eview. The results are shown in Table 1.

From results by 3SLS, we find that the effect of communication pattern on development activity is significant. The model shows the significant effect for both project centrality and project density on development activity, which support hypothesis 1 and 3. However, the model explains the insignificant effect of communication pattern on project popularity. Hypothesis 5 (A, B, C, D) suggests the relationship between project-specific characters and project development activity. The model shows the significant effects of project complexity, financial support, and programming language, and insignificant effect of types of license on development activity. Hypothesis 6 (A, B, C, D) suggests the relationship between project-specific characters and project popularity. The model shows the significant effects of project maturity, types of license, target audience, and insignificant effect of programming language on project popularity.
Discussion, Limitation and Future Research

The main purpose of this study is to investigate the effect of communication pattern on the success of open source project. Communication pattern of the project team is captured by project centrality and project density; while the project success is viewed in terms of development activity and popularity. Results, shown in Table 1, are generally supportive of the hypotheses posited in this paper. Generally speaking, from our study, we find that factors which contribute to demand and supply side success may be different. This implies that different definitions of OSS success require different set of predictors. For example the communication patterns are more related to the activity level on the developer side; the target audience is more related to the popularity of the product, etc.

Table 1: Results by 3SLS

<table>
<thead>
<tr>
<th>Estimation Method: Three-Stage Least Squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample: 156</td>
</tr>
<tr>
<td>Total system (balanced) observations 112</td>
</tr>
<tr>
<td>Linear estimation after one-step weighting matrix</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development Activity</th>
<th>Popularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient</td>
<td>Std. Error</td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>Development Activity</td>
<td>NA</td>
</tr>
<tr>
<td>Popularity</td>
<td>-0.732**</td>
</tr>
<tr>
<td>Project Centrality</td>
<td>0.446**</td>
</tr>
<tr>
<td>Project Density</td>
<td>-1.849**</td>
</tr>
<tr>
<td>Types of License</td>
<td>0.259</td>
</tr>
<tr>
<td>Programming Language</td>
<td>-0.707*</td>
</tr>
<tr>
<td>Financial Support</td>
<td>0.648*</td>
</tr>
<tr>
<td>Project Complexity</td>
<td>0.420*</td>
</tr>
<tr>
<td>Project Maturity</td>
<td>NA</td>
</tr>
<tr>
<td>Target Audience</td>
<td>NA</td>
</tr>
</tbody>
</table>

** significant at 5% confidence interval. * significant at 10% confidence interval.

As one of attributes of communication pattern, project centrality expresses the inequality or variance of contribution in communication among the contributors. The significant positive effects on project development activity are reported from 3SLS analysis. These significant effects indicate that substantial inequality of developers’ contributions to the projects is important for project success. Those who contribute more serve as the core group of the project. Such core-periphery structures hold the potential to improve the speed and flexibility with which information diffuses within a group (Cummings and Cross 2003). Thus, centralized projects will be with higher communication efficiency and thereby related to better performance. Another attribute of communication pattern discussed in this paper is project density. Project density measures the readiness of the group to respond to changes, and how close a network is to realize its potential. In high density projects, the passes of information dissemination are crowded since information needs to travel through the extended hierarchies, although this kind of structure is good to respond to changes. The communication efficiency, which is the key to project development activity, is indeed affected. Thus, our result suggests that project density negatively influences the development activity of project. In a nutshell, the impact of communication pattern of open source development team on project development activity is significant from this study. The impact of communication pattern on project popularity is insignificant, which may be due to the sample size is not large
enough. In our future research, we will use larger sample size data to see whether there is the significant effect of communication pattern on project popularity. In addition, project specific characteristics also show the significant effects on project success from both supply and demand side. For example, the more complex project, the higher possibility to achieve high development activity; the project receiving financial support will be more likely to achieve high development activity; the more mature project, the higher possibility to attract more users and to be more popular.

Our study has limitations that provide avenues for future research. First, some limitations are related to the operationalization and measurement of dependent variable. For example, using number of closed bugs in a fixed period as a measure of development activity may underestimate the supply side of success. Some other measures such as number of file releases are also the indicators of development activity. Multiple measures of development activity will be considered in the future work. Second, we examined only a small proportion of projects in SourceForge.net that use the bug tracking system and that make bug reports public. We are comfortable that our selection criteria are appropriate for identifying truly active projects, but to go beyond these limits, we have started to examine communication pattern of more projects from other interaction data, such as patch reports, supporting requests and feature requests, to see if they display similar patterns of interactions, and if they influence the success of the projects. Finally, we have limited our analysis to project centrality and density to characterize the communication pattern of the projects. It would be of interest to explore the impact of other social structure measures on the success of the projects.

At last, in current study, the communication pattern and success are treated as fixed in this empirical study. However, as we known, communication pattern can change and success could be temporary. Therefore, it is important to see the dynamic impact of communication pattern on project success, through which we can predict the long term sustainability of the project. Future research will examine the dynamic impact of communication pattern on project success through the panel data analysis.

References


Scacchi, W. “Understanding the requirements for developing open source software systems,” 
Sen. “Open source software development projects: determinants of project popularity,” 
Wheeler, D. A. “Why open source software/free software (OSS/FS)? Look at the number!” 