A TOPOLOGICAL ANALYSIS OF THE OPEN SOURCE SOFTWARE DEVELOPMENT COMMUNITY

Jin Xu *Yongqin Gao* *Scott Christley* *Gregory Madey*

Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
Email: {jxu1,ygao1,schristl,gmadey}@nd.edu
Tel: (574)631-7596 Fax: (574)631-9260

ABSTRACT

The fast growth of OSS has increased the interest in studying the composition of the OSS community and its collaboration mechanisms. Moreover, the success of a project may be related to the underlying social structure of the OSS development community. In this paper, we perform a quantitative analysis of Open Source Software developers by studying the entire development community at SourceForge [26]. Statistics and social network properties are explored to find collaborations and the effects of different members in the OSS development community. Small world phenomenon and scale free behaviors are found in the SourceForge development network. These topological properties may potentially explain the success and efficiency of OSS development practices. We also infer from our analysis that weakly associated but contributing co-developers and active users may be an important factor in OSS development.

1. INTRODUCTION

Open Source Software (OSS) has brought us enormous advantages such as reduced development cost, simplified team collaboration and improved software quality. Unlike closed-source software, OSS projects are typically developed in a distributed and decentralized way [8, 25]. OSS projects are written, developed, and debugged largely by worldwide volunteers, who in most cases are connected and collaborate through the Internet. The OSS development community has developed a large number of outstanding software products, including Apache, Perl, Linux, etc. The success of OSS increases our interest in studying the composition of the OSS community and its collaboration mechanisms, both in the business and IT communities. Despite the growth in OSS research, the phenomenon is not yet fully understood [4, 8, 20]. The intrinsic mechanisms in the OSS development process still need to be investigated. For example, unlike closed software, the OSS development process involves developers who irregularly participate in projects. Moreover, the roles of users in OSS may be different from those in closed software because they have more closer contact with developers. Understanding the collaboration among the OSS community may help solve the "software crisis" [25]. Moreover, the success of a project may be related to the underlying social network topology of the OSS development community.

Many researchers have begun to study the OSS development community. For example, Nakakoji et al. [22] classify OSS community members into eight different roles and divide OSS projects into three different types – Exploration-Oriented, Utility-Oriented, and Service-Oriented. Furthermore, they identify two types of OSS evolution patterns: evolution of developer roles, and co-evolution of systems and the roles of contributing members. A modified classification is presented by Xu [33] that reduces the number of OSS member roles from eight to five. (In this paper, we further reduce the number of roles from five to four.) Crowston et al. [6, 15] study developer roles and the structure of OSS development teams for success factors in distributed work teams. Hars et al. report results from a survey to the participants in various OSS projects and categorize motivations of OSS developers [11]. By studying Linux Software Maps (LSMs), Dempsey et al. [7] analyze the body of all extant LSMs at a Linux site to obtain information on the nature of Linux contributors. Data mining techniques were used by Xu et al. to find patterns in the OSS development community [32]. Gao et al. [9, 30, 31] analyze activities of core developers on SourceForge hosted projects and report the scale free behavior of the OSS community. These previous studies are either qualitative classifications, are performed on a small set of sample projects, or the effect of the entire community was not measured.

In this paper, we perform a quantitative analysis of Open

This research was funded in part by the NSF Award-0222829, from the Digital Society & Technologies Program, CISE/IIS.

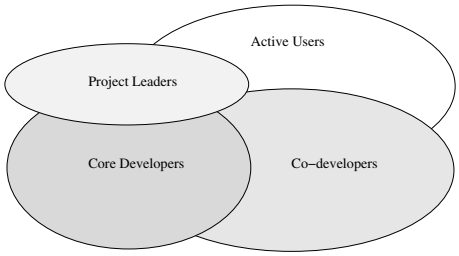


Fig. 1. OSS Development Community Classification

Source Software developers by studying the development community at SourceForge.net [26]. Data is collected and extracted by mining a SourceForge 2003 data dump. By dividing the OSS development community into four subsets, statistics and social network properties are explored to find the effect of different roles in the OSS development community. Based on these social network topological properties, we discover that there are special features in the OSS development community. We give explanations to those features and discuss their relationship to the success of OSS.

The rest of this paper is organized as follows: the next section classifies roles of developers by their activities in projects; the third section gives a brief introduction to the OSS social network; the fourth section describes the small world phenomenon and the scale free network; social network and presents properties we analyze in this paper; then, we describe data extraction and mining process of the SourceForge community; statistical and social network analysis is performed on the collected data to understand the SourceForge development community; the seventh section discusses the discoveries based on the analysis; finally, conclusions and future work are given.

2. OSS DEVELOPMENT COMMUNITY CLASSIFICATION

An OSS development community is composed of groups of loosely-connected contributors with central coordinators and decision makers. According to Xu [33], OSS members can be classified into either the user group or the developer group. The user group includes passive users and active users. *Passive users* have no direct contribution other than forming a larger user base. They just download code and use it for their needs. *Active users* discover and report bugs, suggest new features, and exchange other information by posting messages to forums or mailing lists. The developer group can be further categorized into peripheral developers, central developers, core developers and project leaders. *Peripheral developers* irregularly fix bugs, add features, provide support, write documents, and exchange other information. *Central developers* regularly fix bugs, add features, submit patches, provide support, write documents and exchange other information. *Core developers* extensively contribute to projects, manage CVS releases and coordinate peripheral developers and central developers. *Project leaders* guide the vision and direction of a project.

In this paper, we define that the OSS development community includes all of the above members except passive users, because passive users do not make direct software development contributions and thus are not considered to be part of the development team. Thus, as shown in Fig.1, our OSS development community includes the following groups:

1. Project leaders who are also called project administrators,
2. Core developers who regularly contribute on projects and manage CVS releases,
3. Co-developers including both peripheral developers and central developers, and
4. Active users who have some contributions except modifying code.

Because an individual may participate in multiple projects, he/she can belong to different groups in the development community (overlap in Figure 1). Our definition of the OSS development community is considerably larger than the traditional closed source development team which approximately coincides with our project leader and core developer groups. Our co-developer and active user groups are typically not part of the development communities for traditional closed source software, but are what has been identified as end-user contribution to the product innovation process [12].

3. OSS SOCIAL NETWORK

The social network perspective has been explored to explain an organization's behavior in terms of their embeddedness in social networks [5,10,24,27,29]. Social network analysis seeks to understand the relationships and information flows between people, groups, organizations, or other social entities. A social network can be modeled as a graph with nodes representing people or groups, and links representing relationships or information flows between nodes. Thus, two persons are directly connected if they have a relationship with each other. The path between two nodes in the graph measures the closeness of these two nodes. Social network analysis has been successfully applied in many scientific areas [1]. For example, in a scientific collaboration network, nodes represent scientists, and two nodes have a link if two scientists have coauthored a paper together.

The Open Source Software (OSS) development movement is a classic example of a dynamic social network [16]; it is also a prototype of a complex evolving network [9, 19, 30–32]. In this network, we focus on two entities – developers and projects. Thus, the OSS development community can be represented as a bipartite graph, the project-developer network, with two kinds of nodes to represent developers and projects. Developers in a project are connected to that project. This bipartite graph can be transformed to two unipartite graphs, the developer network and the project network. Fig.2 shows a cluster (a collection of connected nodes) at SourceForge.net. In this paper, we study all three networks to determine the role and influence of different members in the OSS community.

Many properties are used to characterize the topology of the social network. The properties we used in our analysis include:

Degree Distribution: The degree of a node, k, is the total number of links connected to this node. The degree distribution represents the relative frequency of each index value, k, in a given network. The degree distribution of social networks was believed to be the Poisson distribution. Recently, it has been found to often have a power law distribution [2], which is defined as follows:

$$y = x^{\alpha} \tag{1}$$

where α is a constant. The relationship between $log(y)$ and $log(x)$ is linear.

Diameter: The diameter of a social network is the longest shortest-path between all pairs of nodes. In a disconnected network, the diameter can be defined as the maximum diameter of its clusters. An alternative definition of the diameter is the average shortest path between all pairs of nodes. We use the second definition in this paper because the first diameter definition can be hard to compute for large networks. The average diameter indicates the average separation of pairs of nodes in a network, which represents the average distance to connect two nodes together. Newman and Watts developed an approximate calculation of the diameter by using a generating function [23]. The approximate diameter in a random network can be computed by

$$d = \frac{log(N/z_1)}{log(z_2/z_1)} + 1 \tag{2}$$

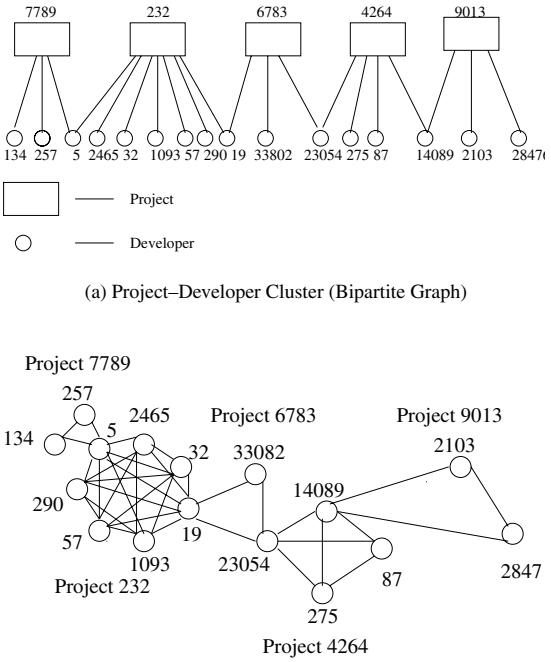
where d is the diameter, N is the total number of nodes, z_1 is the average number of neighbors 1 link away, and z_2 is the average number of neighbors 2 links away. Thus the diameter of a network is increasing logarithmically with N.

Cluster: A *cluster* in a social network consists of connected nodes. Each cluster represents a related community in the network.

Clustering Coefficient: The clustering coefficient of a node is defined as the ratio of the number of links to the total possible number of links among its neighbors. The clustering coefficient is an indicator of the connectivity of a node. The clustering coefficient of a social network is the average of all the clustering coefficients of nodes. The generating function proposed by Newman and Watts can be generalized to bipartite graphs to get the clustering coefficient [1]:

$$C = \frac{1}{1 + \frac{(\mu_1 - \mu_2)(\nu_1 - \nu_2)^2}{\mu_1 \nu_1 (2\nu_1 - 3\nu_2 + \nu_3)}} \tag{3}$$

where $\mu_n = \sum_k k^n P_d(k)$ and $\nu_n = \sum_k k^n P_p(k)$. In the project-developer bipartite network, $P_d(k)$ represents the fraction of developers who joined k projects, while $P_p(k)$ means the fraction of projects which have k developers.



(a) Project–Developer Cluster (Bipartite Graph)

(b) Developer Cluster (Unipartite Graph)

Fig. 2. Modeling OSS as a Social Network: a Cluster of 5 Projects and 16 Developers (Projects and Developers are Anonymized

4. SMALL WORLD PHENOMENON AND SCALE FREE NETWORK

The *small world phenomenon* is the principle that everyone in the world can be reached through a short chain of acquaintances. In 1960s, psychologist Stanley Milgram performed a small world experiment to trace paths through the

social network of residents of the United States. He found that two random persons were connected by an average of six acquaintances, which is called "six degrees of separation" [21]. Duncan Watts and Steve Strogatz provided evidence that the small world phenomenon exists in many real networks [28]. They showed that the addition of a few random links can turn a "large world" into a "small world" network. They defined a small world network to include two features – a high clustering coefficient and a small network diameter.

Barabasi and Albert found that some small world networks have another special property. Such networks contain relatively few nodes that are highly connected to other nodes, while the vast majority of nodes are only connected to a few other nodes. These networks are called *scale-free* networks. According to Barabasi and Albert [3], such a network is generated by two rules. First, the network grows by the sequential addition of new nodes; second, there exists *preferential attachment* – the probability for a newly added node to be connected to an existing node depends on the degree of the existing node. This phenomenon is sometimes called the "rich gets richer" phenomenon. In scale free networks, the degree distribution of index values of nodes follows the power law distribution.

5. DATA COLLECTION AND EXTRACTION

One challenge in studying OSS development is to collect and extract data. Data collection has proven to be tedious and time consuming [14]. Many difficulties exist in collecting, cleaning, screening and interpreting data [13]. In our previous studies [9, 17–19, 32], web-bots were used to retrieve and extract data from web pages at SourceForge.net. This process often took days, and frequently was plagued by incomplete and missing data. In this section, we discuss a much improved data collection and extraction process used in mining the SourceForge data.

There are many web sites which host OSS projects. With around $87,000$ projects and $912,000$ registered users as of mid-2004, SourceForge.net, sponsored by VA Software is the largest OSS development and collaboration site. It offers a centralized place for OSS developers to control and manage OSS development by providing project web servers, trackers, mailing lists, discussion boards, and software releases, etc. This site provides highly detailed information about projects and developers, including project characteristics, developers' activities, and "top ranked" developers. By studying these web sites, we can explore developers' behaviors and projects' growth.

We extracted data from a 2003 data dump obtained from SourceForge. The data dump is derived from a PostgreSQL relational database used as the "back-tier" database for the SourceForge.net web site. The data dump contains information about the community, projects, and developers. We examined these data to characterize the entire SourceForge community, across multiple numbers of projects, investigating behaviors and mechanisms at the project and developer levels.

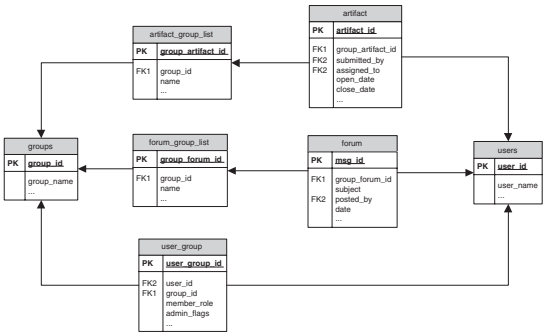


Fig. 3. A Subset of SourceForge Database Schema

In the SourceForge data dump, information about the roles of developers on each project is distributed over seven tables. Two roles, the project leader and core developer roles, are explicitly defined and stored in a single table. The other two roles, co-developers and active users, must be inferred and extracted from project activity data, such as bug reports, patch submission, forum discussions, etc. The seven tables and their relationships are shown in Fig.3. Table *groups* is the list of all projects. Table *artifact_group_list* and *artifact* contain members' activities such as bug tracking, patch submission, feature requests, and document writing, etc. Table *forum_group_list* and *forum* reflect members' participation in open discussion forums. Table *users* lists all users' information including their names and contact information, etc. Table *user_group* contains the relationships between projects and project leaders as well as core developers.

By processing the above tables, we can identify members and their participation activities for each project. The data extraction process is shown in Fig.4. We used a three-step data integration and data reduction process on the data. Data integration combines data from multiple sources into a coherent store. In the first step, we integrate *artifact* and *forum* separately to create two tables – *artifact_activity* and *forum_activity* to contain each member's activities in *artifacts* and *forums*; because some attributes are different in *artifacts* and *forums*, we combine *artifact_activity* and *forum_activity* with *users* table to get a member's activities for all projects he/she participates, which are recorded in Table *user_activity*; by integrating this table with *user_group*, we can identify each member's role in a project. We put this information into a table called *user_project_role*; lastly, data reduction is used to reduce the huge data set to a smaller representative subset according to members' role in a project.

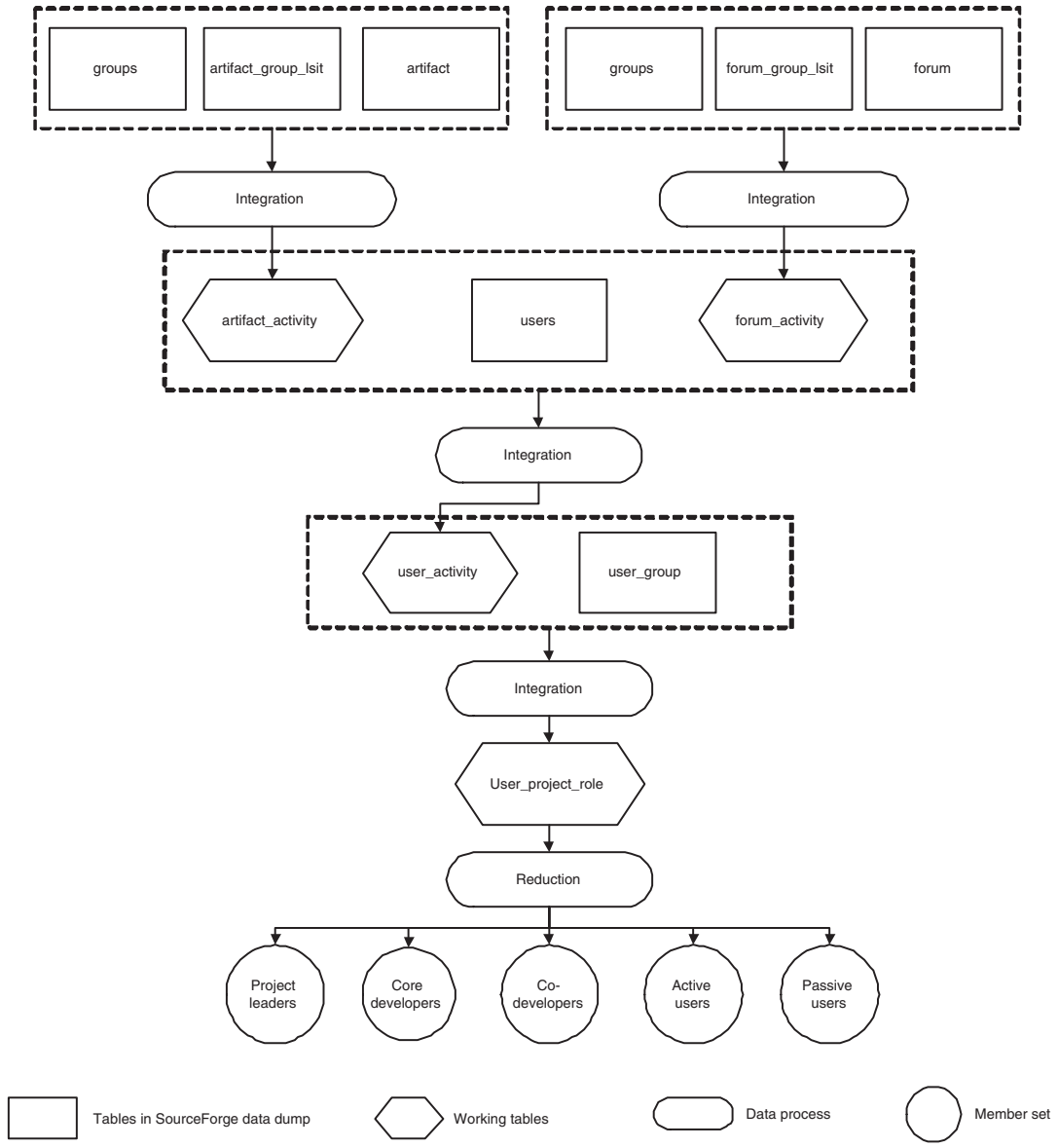


Fig. 4. Data Collection and Extraction Process

Table 1. The Community Distribution for Large, Middle and Small Projects

Developer Size	Project Number	Project Leaders		Core Developers		Co-developers		Active Users	
≤ 88	64847	80329	(47.8%)	34659	(20.6%)	33275	(19.8%)	19941	(11.8%)
> 88 and ≤ 279	193	590	(2.1%)	1703	(5.7%)	17334	(60.3%)	9124	(31.7%)
> 279	70	798	(0.9%)	2576	(2.7%)	53030	(55.8%)	38593	(40.6%)

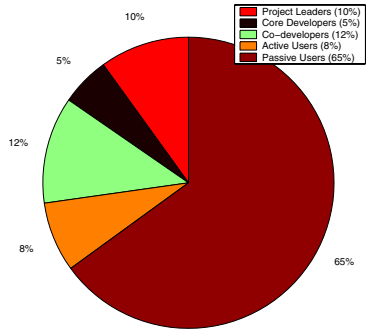


Fig. 5. Distribution of SourceForge Population

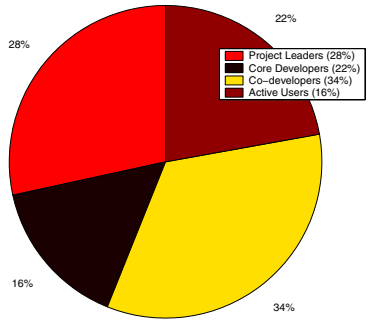


Fig. 6. Distribution of SourceForge Development Community

6. ANALYSIS OF SOURCEFORGE DEVELOPMENT COMMUNITY

In this section, we examine the community distribution of the entire SourceForge population; then, we analyze the topological properties of the SourceForge OSS development community.

6.1. Member Distribution

As described in more detail in Section 2, we classified member groups as follows: project leaders are administrators in each project; core developers are developers listed in each project at SourceForge.net; co-developers (central and peripheral developers) are people who are assigned to tasks such as bug fixing and document writing, but are not listed as project leaders and core developers; active users are those who submit requests and post messages, but are not included in the project leaders, core developers and co-developers groups; passive users are obtained by excluding all developers from all users. Because a person can have different roles in different projects, we put him/her into his/her highest ranked group. For example, if a person is listed as a project leader in one project and a core developer in another project, he/she will be counted as in the project leader group. Fig.5 shows the distribution of members in the entire SourceForge population. About 65% of the SourceForge population are passive users who make no observable contributions to the development of projects. Among developers (shown in Fig.6), which comprise 35% of the entire population, there are 28% project leaders, 16% core developers, 34% co-developers and 22% active users. We observed that co-developers have almost the same percentage as the sum of project leaders and core developers. This is because a large portion of projects on SourceForge are small and most developers on them are also initiators of the projects.

To further investigate the relationship between the project size and the development community, we divide projects into three categories, each representing approximately one-third of the range from smallest to largest community sizes: large projects, middle projects and small projects, and analyzed their community distributions. Large projects are those with members greater than 279. Small projects contain less than or equal to 88 members. Middle size projects have members between 88 and 279. We note that from a software engineering perspective, a small project team is typically composed of 1–10 developers and mid-size goes from 10–50. However, because we include both co-developers and active users in the open source development community, the project sizes should be much larger than those of traditional closed source software development teams. Table 1 gives the development community distribution of these groups. In small projects, the main part of the community are project leaders (47.8%) and core developers (20.6%). As the size increases, more and more co-developers and active users join the project. In large projects, project leaders and core developers consist only 3.6% of the whole community, while co-developers and active users are 55.8% and 40.6%, separately. The fact that co-developers and active users comprise a large part in those more popular projects implies that they may play a crucial role in OSS projects.

6.2. Topological Analysis

To understand how OSS development members (especially co-developers and active users) collaborate and their effect on the whole community, we divide the OSS development community into four nested subsets. Each subset grows by including more individuals based on their roles in the SourceForge OSS community:

- Subset A $= \{$project leaders$\}$;
- Subset B $= \{$project leaders$\} \cup \{$core developers$\}$;
- Subset C $= \{$project leaders$\} \cup \{$core developers$\} \cup \{$co-developers$\}$;

Table 2. Regression Parameters of Degree Distributions

	Parameters	Subset A	Subset B	Subset C	Subset D
Project-side	R^2	0.9396	0.9704	0.6905	0.7221
	Slope	-3.5841	-2.6968	-1.3020	-1.2220
Developer-side	R^2	0.9870	0.9846	0.9469	0.9830
	Slope	-3.3747	-3.4676	-3.7793	-3.2743

- Subset D = {project leaders} ∪ {core developers} ∪ {co-developers} ∪ {active users} ;

We analyze the topological properties of these four subsets to help identify the different characteristics of each subset and determine the effect of different community members.

6.2.1. Degree Distribution

Degree distribution is the relative frequency of the index values (number of edges on a node) throughout the network. Degree distribution was recently found to fit a power law distribution in many real networks [2]. A power law distribution is an important characteristic of a scale free network. As we introduced in Section 4, such networks can be generated using two rules: sequential addition of new nodes and preferential attachment.

We hypothesize that the OSS community is a scale free network because it may be growing and self-organizing using the above two rules. In the OSS community network, the number of community members and projects grow over time. With the evolution of projects, community members sequentially join projects. Furthermore, the OSS development community is highly decentralized. Developers freely participate on projects which attract them. Some projects are more popular than others. Such projects tend to attract more developers and users. Thus, in this network, there exists a preferential attachment of developers to projects.

To support of our hypothesis, we compute the degree distributions of SourceForge project and developer networks, shown in Fig.7. All left subgraphs represent the distributions of the log-log transformation of project size frequencies. The right subgraphs display the distributions of the log-log transformation of the project membership of developers. All eight subgraphs show degree distributions which are highly skewed. For example, on the right subgraphs, a large number of members only participate on one project (45261 in subset A, 63103 in subset B, 105234 in subset C, 113999 in subset D). But some members join multiple projects. When comparing subset A to subset D, the largest number of projects a member joins increases from 29 to 95. These members are linchpin nodes in the community network, who link projects together into clusters. We can observe that all distributions display the power law relationship. Table 2 shows the linear regressions of all eight subgraphs. Such a power law relationship suggest that the SourceForge development network is a scale free network. In this network, a successful project can attract more developers, while many projects will stagnate after a short while. The number of members in a project may be an important positive feedback factor in determining the attractiveness of a project.

Table 3. The Properties of the Development Community

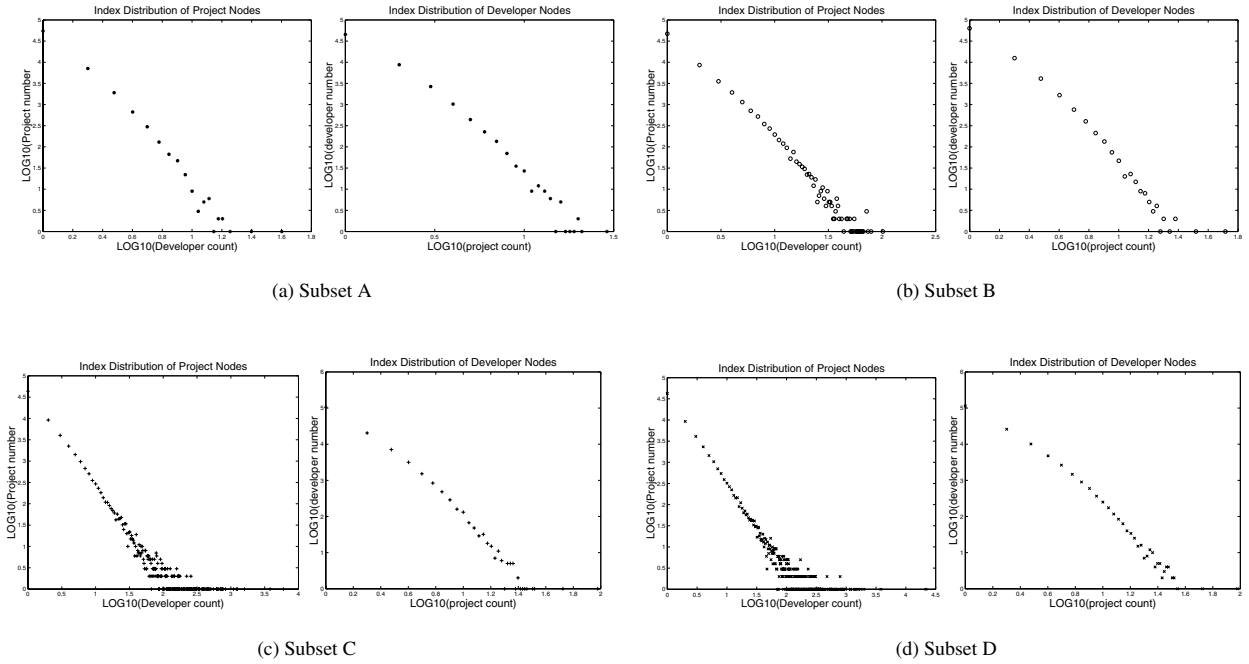
Property	Subset A	Subset B	Subset C	Subset D
Size	58651	83118	139570	161691
$\mathbf{z_1}$	1	6	508	3241
$\mathbf{z_2}$	1	17	13398	31998
Diameter	Inf.	10.2	2.7	2.7
Clustering Coefficient	0.8406	0.8078	0.8867	0.8297
Largest Project Cluster	737	15091	30794	40175
$\mathbf{2^{nd}}$ **Largest Project Cluster**	197	34	20	20
# of Project Clusters	43826	34280	27983	21659

6.2.2. Diameter

Using Equation 2, we calculated the average shortest path length of the four community subsets. As shown in Table 3, if the community is composed of only project leaders, the average shortest path length computed using Equation 2 in this network is infinity. This reflects the fact that the project leader network is highly disconnected. Perhaps project leaders are to busy to join other projects? Core developers are more willing (or able?) to participate on other projects. In the subset B, which includes both project leaders and core developers, the average shortest path is 10 out of 83118 members. Co-developers and active users are likely to join multiple projects. Subset C and Subset D both have the average shortest path as 3. It takes about 3 steps to reach a randomly chosen member in the SourceForge development community. With the participation of developers and active users, the degree of separation is significantly decreased. In the OSS development community, members can be closely related to each other although they are working in a distributed environment. Information such as ideas and discussions can spread fast in the OSS development community because of the short distance. By understanding this fact, certain phenomenon can be understood in terms of the short communication path available in the OSS community. Also, the number of separated clusters influences the spread of ideas throughout a network. This property is described in the following subsection.

6.2.3. Clusters – Sizes and Numbers

Two projects are linked if they share a member. All linked projects form a project cluster (See Fig.2 for an example). Each project cluster has a corresponding developer cluster. Thus, we limit our discussion in this section to project clusters. All four subsets of the SourceForge development com-



(a) Subset A

(b) Subset B

(c) Subset C

(d) Subset D

Fig. 7. The SourceForge Project and Developer Community Scale Free Degree Distributions

munity contain many separated clusters (shown in Table 3). Also in Table 3, we see that the largest cluster is much bigger than the second largest cluster and grows as we move from subset A to subset D. However, the second largest cluster decreases with the increase of the community size. The reason is that some clusters are linked to become part of the largest cluster. The larger a cluster is, the more possible it will attach to the largest cluster as the community size increases.

This type of cluster analysis may be used to identify groups of related projects or developers with similar interests. This discovery may reflect that some projects are more similar than others. They may have some common characteristics. This may be applied to project management, for example, by identifying similar projects, we can study the life cycle of an old project to predict the future of a young project. Alternatively, recommender systems such as those found at online shopping sites may be developed to assist users and developers looking for software or projects to participate on.

6.2.4. Clustering Coefficient

We use Equation 3 to get the approximate clustering coefficients of the four subsets of the SourceForge community. The clustering coefficient tells us how many of a member's collaborators are collaborators with each other. As shown in Table 3, the clustering coefficients of all four subsets are above 0.8. The high clustering coefficients are not surprising because members are fully connected in each project.

7. DISCUSSION

We found a small world phenomenon, the small diameter and the high clustering coefficient, in the SourceForge development community. The small distance results from the fact that a member may participate in multiple projects. In this way, the member connects separate clusters together and creates a path among members in those clusters. However, a large percentage of members participate on one project (70.5% in subset D). This fact explains the observed high clustering coefficient. Furthermore, the power law distribution found in the SourceForge OSS network supports the claim that it is a scale-free network. OSS projects are often developed by collaborating volunteers, who sequentially join projects based on their interest.

In previous studies [9, 19], we observed the power law distribution and the small world phenomenon in the SourceForge project leaders and core developers community (classified as subset B in this paper). In this paper, we observe that with the participation of co-developers and active users,

while the cluster is increasing, the diameter is much smaller (only about 3 links between pairs of community members). This fact supports the assertion that co-developers and core developers play a crucial role in connecting the SourceForge development community. Their existence can make communication flow faster throughout the whole OSS community. This fast communication may be an example of a self-organized optimized resource reallocation. For example, our SourceForge data contains 676 text editor projects. Of all developers on those projects, about 50% of developers are on the top 6 largest projects. One reason might be due to the short communication path in the development community through which people can find projects which attract them. The feature of fast communication in OSS development community may be a factor of its success because closed source software development does not typically have co-developers, and active users may not be in close contact with developers.

8. CONCLUSIONS

In this paper, we studied the Open Source Software development community at SourceForge. Based on a SourceForge 2003 data dump, we performed quantitative analysis on the SourceForge OSS development community. Using statistical analysis, we found that large and small projects have different community distributions. Large projects consist mainly of co-developers and active users, while project leaders and core-developers are the main part of small projects. Furthermore, we conducted topological analysis on four subsets of the OSS development community. Properties of the community network show that the SourceForge OSS development community is a self-organizing system which obeys scale-free behaviors. Moreover, small-world phenomenon, the small average distance and the high clustering coefficient, exists in the community. We identify the important effect of co-developers and active users because their addition can turn the OSS community into a fast communication network. Our research provides useful information of the underlying structure and components of the OSS community. The information in this paper is important in studying the evolution of the OSS community.

The work in this paper is part of an ongoing OSS study. Future work will focus on the simulation of OSS developer network based on the data and analysis in this paper. What still needs to be done is to provide motivation for the individual users in terms of a social theory. We want to expand upon this work by incorporating some social network theories into an agent-based model so that we can perform computer experiments for hypothesis testing. Furthermore, our analysis is based only on SourceForge.net. We will explore some other OSS project sites, (e.g. Savannah, Linux, Mozilla, and Apache) to test if the development community characteristics found in SourceForge are presented in those sites.

9. REFERENCES

[1] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Review of Modern Physics*, v. 74:47–97, 2002.

[2] R. Albert, H. Jeong, and A. L. Barabasi. Diameter of the world wide web. *Nature*, sv. 401:130–131, 1999.

[3] A. L. Barabasi and R Albert. Emergence of scaling in random network. *Science*, v. 286:509–512, 1999.

[4] D. Bollier. The power of openness: Why citizens, education, government and business should care about the coming revolution in open source code software. http://eon.law.harvard.edy/opencode/h20, 1999.

[5] R.S Burt. *Structural holes: The social structure of competition*. Harvard University Press, Cambridge, MA, 1992.

[6] K. Crowston, B. Scozzi, and S. Buonocore. An exploratory study of open source software development team structure. http://floss.syr.edu/tiki-index.php, 2002.

[7] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg. Who is an open source software developer? *Communications of the ACM*, v. 45(2):67–72, 2002.

[8] J. Feller and B. Fitzgerald. *Understanding Open Source Software Development*. Addison-Wesley, London, UK, 2002.

[9] Y. Q. Gao, V. Freeh, and G. Madey. Analysis and modeling of the open source software community. In *North American Association for Computational Social and Organizational Science (NAACSOS 2003)*, Pittsburgh, PA, 2003.

[10] R. Gulati and M. Gargiulo. Where do interorganizaiotnal networks come from? *American Journal of Sociology*, v. 104:1439–1493, 1999.

[11] A. Hars and S. Ou. Working for free? – motivations for participating in open source projects. In *Proceedings 34th HICSS Conference*, Maui, 2001.

[12] Eric von Hippel and Georg von Krogh. Open source software and the "private-collective" innovation model: issues for organization science. *Organization Science*, v. 14(2):209–223, 2003.

[13] J. Howison and K. Crowston. The perils and pitfalls of mining sourceforge. In *Proceedings of Mining Software Repositories Workshop, International Conference on Software Enginnering (ICSE 2004)*, Edinburgh, Scotland, 2004.

[14] C. Jensen and W. Scacchi. Data mining for software process discovery in open source software development communities. In *Proc. Workshop on Mining Software Repositories*, Edinburgh, Scotland, 2004.

[15] Crowston K., Annabi H., and Howison J. Defining open source software project success. In *Proc. of International Conference on Information Systems (ICIS)*, Seattle, Washington, 2003.

[16] L. Lopez-Fernandez, G. Robles, and J.M. Gonzalez-Barahona. Applying social network analysis to the information in cvs repositories. In *Proceedings of the First International Workshop on Mining Software Repositories (MSR 2004)*, Edinburgh, UK, 2004.

[17] G. Madey, V. Freeh, and R. Tynan. The open source software development phenomenon: An analysis based on social network theory. In *Americas Conference on Information Systems (AMCIS2002)*, Dallas, TX, 2002.

[18] G. Madey, V. Freeh, and R. Tynan. Understanding oss as a self-organizing process. In *The 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002)*, Orlando, FL, 2002.

[19] G. Madey, V. Freeh, and R. Tynan. *Modeling the F/OSS Community: A Quantitative Investigation, Free/Open Source Software Development, Edited by Koch S.* Idea Publishing, 2004.

[20] S. McConnell. Open-source methodology: Ready for prime time? *IEEE Software*, v. 16(4):6–8, 1999.

[21] S. Milgram. The small world problem. *Psychology Today*, v. 2:60–67, 1967.

[22] K. Nakakoji, Y. Yamamoto, K. Kishida, and Y. Ye. Evolution patterns of open-source software systems and communities. In *The International Workshop on Principles of Software Evolution*, Orlando Florida, 2002.

[23] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. In *Proc. Natl. Acad. Sci.*, pages pages 2566–2572, 2002.

[24] W.W. Powell, D.R. White, K.W. Koput, and Smith J. Oweb. Network dynamics and field evolution: the growth of interorganizational collavoration in life sciences, 2002. Forthcoming, American Journal of Sociology.

[25] E. Raymond. The catherdral and the bazaar. *First Monday, http://www.firstmonday.dk/*, v. 3, 1998.

[26] Sourceforge. http://www.sourceforge.net, 1999.

[27] B. Uzzi. The sources and consequences of embeddedness for the economic performance of organizations: The network effect. *American Sociological Review*, v. 61:674–698, 1996.

[28] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, v. 393:440–442, 1998.

[29] H.C. White. *Markets from networks: socioeconomic models of production*. Princeton University Press, Princeton, NJ, 2002.

[30] J. Xu, Y. Gao, J. Goett, and G. Madey. A multi-model docking experiment of dynamic social network simulations. In *Agents 2003*, Chicago, IL, 2003.

[31] J. Xu, Y.Q. Gao, and G. Madey. A docking experiment: Swarm and repast for social network modeling. In *Seventh Annual Swarm Researchers Meeting (Swarm2003)*, Notre Dame, IN, 2003.

[32] J. Xu, Y. Huang, and G. Madey. A research support system framework for web data mining. In *Workshop on Applications, Products and Services of Web-based Support Systems at the Joint International Conference on Web Intelligence (2003 IEEE/WIC) and Intelligent Agent Technology*, Halifax, Canada, 2003.

[33] N. Xu. An exploratory study of open source software based on public project archives. Master's thesis, the John Molson School of Business, Concordia University, Canada, 2003.