

The Economics of Open Source Software Development
Jürgen Bitzer and Philipp J. H. Schröder (Editors)
© 2006 Published by Elsevier B.V.

11

Application of Social Network Analysis to the Study of Open Source Software

Jin Xu, Scott Christley, and Gregory Madey*

ABSTRACT

Social network analysis has been used in many research areas to discover the intrinsic mechanisms of social communities by examining the topological properties of the social network formed by relationships between the actors and the groups in those communities. In this chapter, we construct four social networks for the Open Source Software (OSS) development community at SourceForge (Sourceforge 1999). For each social network, we expand the number of people in the network by including the next set of peripheral users as defined by their role in the community, moving from the core project leaders, to the core developers, to the co-developers, and finally out to active users. Social network properties such as degree distribution, diameter, cluster size, and clustering coefficient are calculated and compared for each of the expanding social networks, and we elaborate on how the changing topological characteristics of the social networks may signify important capabilities for the diffusion of information, the ability to find collaborations, and the overall robustness of the OSS development community. We further find that all the social networks have scale-free properties, and the inclusion of the co-developers and active users triggers the emergence of the small world phenomenon for the social network. We examine how these topological network properties may potentially explain the success and efficiency of OSS development practices.

Keywords: Open Source Software, social network, data mining.

JEL Classification: C8.

* Corresponding author.

11.1 INTRODUCTION

There has been increasing application of social network analysis in many research areas (Burt 1992; Uzzi 1996; Gulati and Gargiulo 1999; White 2002; Powell et al. 2005). A social network is formed when there are social interactions between individuals, e.g. friendship, marriage, or communication. Social network analysis concentrates on mapping and measuring the relationships between people, groups, organizations, or other social entities.

An abundance of research has been done using network analysis. Newman (Newman 2001) constructed collaboration networks, linked by coauthorships. Network properties are identified and studied, such as typical distances between scientists. Furthermore, he suggests a measure of centrality to study the strength of collaborative ties. Albert et al. (Albert et al. 1999) studied the topology of the World Wide Web and calculated its diameter. Structures of several computer networks were explored to analyze virus and worm infections (Balthrop et al. 2004). They provided a dynamic mechanism to control contagion for different computer network types. Some researchers have used this method to analyze the Open Source Software (OSS) phenomenon. Gao et al. (Gao et al. 2003; Xu et al. 2003) analyzed the core developer network on SourceForge-hosted projects. They explored network properties and reported the presence of scale-free behavior in this network. Social network analysis was used to analyze the source code repositories of OSS projects (Lopez-Fernandez et al. 2004). Moreover, case studies are provided to show how to apply social network analysis to explore the structure, evolution, and internal process of software projects.

The OSS development practices promise enormous advantages such as reduced development cost, simplified team collaboration, and improved software quality. The OSS development practices differ greatly from commercial software development models in several aspects: commercial software companies prevent access to the source code of their products from outside developers and customers, while OSS allows source code to be freely modified and redistributed under “open source” licenses. More importantly, unlike closed-source software which is developed by centralized development teams, OSS projects are typically developed in a distributed and decentralized way (Raymond 1998; Feller and Fitzgerald 2002). The OSS projects are written, developed, and debugged largely by worldwide volunteers, who in most cases are connected and collaborate solely 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 has been attributed to effective project composition and efficient coordination mechanisms, which are due to the fact that the source code is open to inspection and participation is open to any interested individual (Crowston and Scozzi 2002). The OSS communities are one of the most successful high-performance collaboration communities on the Internet (Kim 2003).

Despite the growth in OSS research, the phenomenon is not yet fully understood (Bollier 1999; McConnell 1999; Feller and Fitzgerald 2002). The intrinsic mechanisms in the OSS development process still need to be investigated. For example, unlike closed-source 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-source software because they have closer contact with developers. Understanding

01 the collaboration among the OSS community may help solve the “software crisis”
02 (Raymond 1998). Moreover, the success of a project may be related to the underlying
03 social network topology of the OSS development community.

04 This chapter provides a comprehensive social network analysis of a large segment of
05 the OSS development community. Data is collected and extracted by mining a Source-
06 Forge 2003 data dump (Sourceforge 1999). By dividing the OSS development community
07 into four subsets, statistical and social network properties are explored to find the effect
08 of different roles in the OSS development community. Based on these social network
09 topological properties, we discover that there are special features in the OSS develop-
10 ment community. We give explanations to these topological and evolutionary features
11 and discuss their relationship to the success of OSS projects.

12 The rest of this chapter is organized as follows. Section 11.2 provides background on
13 the OSS social networks and topological network properties. Section 11.3 classifies roles
14 of developers by their activities in projects. Section 11.4 presents data extraction and
15 mining processes used for our analysis of the SourceForge community. In Section 11.5,
16 we perform statistical analysis on the collected data to find the member distribution in
17 OSS; furthermore, we study network properties to understand the topological features
18 of the OSS development community; In Section 11.6, conclusions and future work are
19 given.

20 21 22 **11.2 SOCIAL NETWORK PERSPECTIVES**

23
24 A social network consists of a group of actors connected through relations that hold them
25 together (Haythornthwaite 2001). An actor can be an individual or an aggregate unit,
26 e.g. organizations, groups, or other information/knowledge processing entities. Pairs of
27 actors maintain *ties* if they have one or more relations. The behavior of actors is affected
28 more by their ties and the social networks in which they are embedded than by the
29 attributes they possess (Wellman 1988). Thus, studying actors based on some attribute
30 data may not be enough to show the characteristics of their relations. Social network
31 analysis seeks to explore the network features which affect actors’ interactions.

32 A social network can be modeled as a graph with nodes representing people or
33 groups, and links representing relationships or information flows between nodes. Thus,
34 two people have a link between them if they have a relationship with each other.
35 The path between two nodes in the graph measures the closeness of those two nodes.
36 Social network analysis has been successfully applied in many scientific areas (Albert
37 and Barabasi 2002). For example, in a scientific collaboration network where nodes
38 represent scientists, a link between two nodes might indicate that those two scientists
39 have coauthored a paper together.

40 41 **11.2.1 Open Source Software Social Networks**

42
43 The OSS development movement is a classic example of a dynamic social network
44 (Lopez-Fernandez et al. 2004); it is also a prototype of a complex evolving network
45 (Gao et al. 2003; Xu et al. 2003; Madey et al. 2004). Developers collaborate with each
46 other through the Internet. We are interested in how these distant developers create

01 and sustain such a social network. The formation of the social network begins when
02 developers join a project, work with others, and form co-working relationships. This
03 social network bonding is important in maintaining developers, collaborations in the
04 OSS-decentralized development environment because developers can feel a sense of
05 belonging to a group, which is necessary to sustain the group as a whole entity rather
06 than as a set of separate individuals (Haythornthwaite 2001). Benefits of such a feeling
07 include greater performance, greater cooperation, and greater satisfaction (McGrath 1984;
08 Gabarro 1990).

09 The OSS social networks we constructed have two entities – developers and projects.
10 The social network is a bipartite (two-mode, affiliation) graph, with two kinds of nodes
11 to separately represent developers and projects. Links are formed between developers
12 and projects signifying the relationship that a developer performs some sort of activity
13 or has membership status for a project; as part of our analysis, we will expand the
14 range of activities represented as links to form several larger social networks which are
15 more inclusive of people in the community. The social network as a bipartite graph
16 can be transformed into two unipartite graphs, the developer network and the project
17 network, where all of the nodes are the same type of either developers or projects,
18 respectively. When transformed to the developer network, a link between developers
19 represents a shared project for those two developers. Most developer communication for
20 projects in OSS development occurs through forums, discussion boards, and mailings
21 lists which act as a form of broadcast allowing single individuals to communicate directly
22 to the whole group, so the developer network then represents a communication network
23 based upon the communication mediums provided by projects. When transformed to the
24 project network, a link between two projects indicates they have one or more common
25 developers. The developer network is used to study an individual's impact in the whole
26 network, while the project network is used to study a project's impact in the whole
27 network. Both networks can be used to study how information flows and diffuses either
28 from individual to individual through a project communication medium or from project to
29 project through the action of an individual. Both developer and project networks are used
30 to study the OSS phenomenon, but we primarily focus on the project network to explore
31 how individuals tie together many of the OSS projects into a larger social community.
32 Figure 11.1 shows a cluster (a collection of connected nodes) at SourceForge.net along
33 with the corresponding developer network generated from that cluster. A visualization
34 of a large group of projects, a sampling of the complete project network, can be seen in
35 Figure 11.7.

36 37 38 11.2.2 Topological Network Properties

39 The topology of a social network can be explored by studying its network properties.
40 Some special phenomena are already discovered to exist in the topology of many social
41 networks. In this section, we examine network properties in the OSS network.

42 Many properties are used to characterize the topology of the social network. The
43 following properties are used in our analysis of the OSS social networks:

- 44
45 • *Degree distribution*: The degree of a node, k , is the total number of links connected
46 to this node. The degree distribution represents the relative frequency of each value of

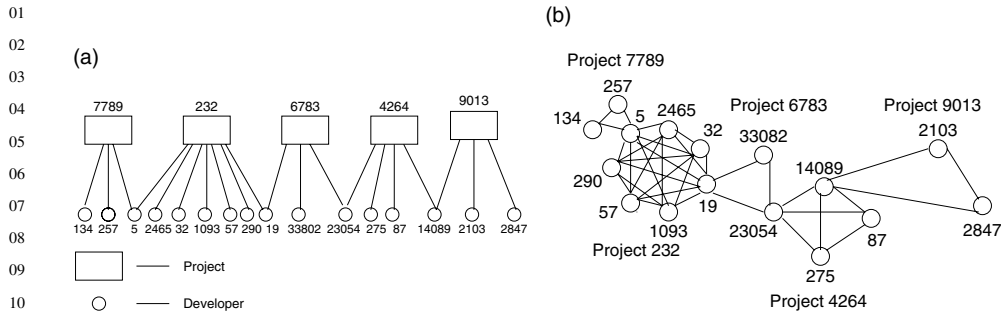


Figure 11.1 Modeling OSS as a social network: a cluster of 5 projects and 16 developers (projects and developers are anonymized to preserve privacy): (a) Project developer cluster (bipartite graph); (b) developer cluster (unipartite graph).

the node degree, k , in a given network. The degree distribution of social networks was generally believed to be the Poisson distribution meaning that links between nodes were formed randomly; however, recent research has shown that many real-world social networks actually have a degree distribution which is a power law (Albert et al. 1999), which is defined as follows:

$$y = bx^\alpha \tag{11.1}$$

where b, α are constants. The relationship between $\log(y)$ and $\log(x)$ is linear.

- **Diameter:** The shortest path between two nodes is defined as a sequence of links going from one node to the other node where the number of links, or length, of the path is minimized. The diameter of a social network is the longest of the shortest paths; that is, of all the shortest paths between all nodes, the path with the longest length signifies the diameter of the network, so it is the maximum number of links that have to be traversed for communication to flow from one node to another. If the network is disconnected, a path does not exist between all nodes; then, the diameter can be defined as infinity or the maximum diameter of its connected clusters. Calculating the longest shortest path is computationally expensive, so an alternative definition that is frequently used is the average shortest path. The average shortest path defines an average diameter which represents the average distance to traverse from one node to another; in this chapter, we use the average diameter because it can be approximated. Newman and Watts developed an approximate calculation of the average diameter by using a generating function (Newman et al. 2002). The approximate average diameter in a random network can be computed by

$$d = \frac{\log(N/z_1)}{\log(z_2/z_1)} + 1 \tag{11.2}$$

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

- 01 • *Cluster*: A *cluster* in a social network consists of the set of all nodes which are
02 connected together by some path; that is, there exists a sequence of links that can be
03 traversed to go from one node to any other node. A social network may have numerous
04 clusters, so nodes within a cluster are connected by a path, but no path exists between
05 two nodes in different clusters. Clusters are often considered to represent communities
06 within a social network.
- 07 • *Clustering coefficient*: The clustering coefficient of a node is defined as the ratio
08 of the number of links to the total possible number of links among its neighbors.
09 The clustering coefficient of a node is an indicator of the connectivity for that node,
10 and the clustering coefficient of a social network is the average of all the clustering
11 coefficients of the nodes. The generating function proposed by Newman and Watts
12 can be generalized to bipartite graphs to get the clustering coefficient (Albert and
13 Barabasi 2002):

$$C = \frac{1}{1 + \frac{(\mu_1 - \mu_2)(v_1 - v_2)^2}{\mu_1 v_1 (2v_1 - 3v_2 + v_3)}} \quad (11.3)$$

14
15
16
17 where $\mu_n = \sum_k k^n P_d(k)$ and $v_n = \sum_k k^n P_p(k)$. In the project-developer bipartite net-
18 work, $P_d(k)$ represents the fraction of developers who joined k projects, while $P_p(k)$
19 means the fraction of projects which have k developers.
20

21 22 11.2.3 Strong and Weak Ties

23
24 In a social network, an actor forms different kinds of connections to others; thus, ties
25 have different strength. Strong ties are formed between actors with similar background,
26 experience, and resources. For example, strong ties exist between friends and relatives.
27 Actors who are strongly tied usually have an intimate or special relationship, frequent
28 interactions, and common needs. Weak ties, by contrast, involve infrequent and limited
29 interactions. A sales lady working in a grocery store might have a weak tie with you.
30 You only have a relationship with her when you shop in her store.

31 Strong ties are important because we are more likely to exchange information with
32 our strong ties and get direct help from them. Strong ties reinforce our companionship
33 and provide us with support. People in a group maintain strong ties with other members
34 in the same group. They share the same goals and work closely. The study of strong ties
35 can help us understand the structure and local information of a social group.

36 Weak ties bring us more new opportunities and resources which cannot be obtained
37 from close relations (Granovetter 1973). For example, although counterintuitive, weak
38 ties may be more useful in helping a person to find a job than close friends. With weak
39 ties, information, ideas, and resources can be exchanged and flowed more broadly. Study
40 of weak ties gives us a global view of the whole community.
41

42 11.2.4 Small World Phenomenon and Scale-Free Network

43
44 The *small world phenomenon* is the principle that everyone in the world can be reached
45 through a short chain of acquaintances. In the 1960s, psychologist Stanley Milgram
46 performed a small world experiment to trace paths through the social network of residents

01 of the United States. He found that two random persons were connected by an average of
02 six acquaintances, which is called “six degrees of separation” (Milgram 1967). Duncan
03 Watts and Steve Strogatz provided evidence that the small world phenomenon exists in
04 many real networks (Watts and Strogatz 1998). They showed that the addition of a few
05 random links can turn a “large world” into a “small world” network. They defined a
06 small world network to include two features – a high clustering coefficient and a small
07 network diameter.

08 Barabasi and Albert found that some small world networks have another special
09 property. Such networks contain relatively few nodes (called “hubs”) which are highly
10 connected to other nodes, while the vast majority of nodes are only connected to a few
11 other nodes. These networks are called *scale-free* networks. According to Barabasi and
12 Albert (Albert and Barabasi 1999), such a network is generated by two rules. First, the
13 network grows by the sequential addition of new nodes; second, there exists *preferential*
14 *attachment* – the probability for a newly added node to be connected to an existing
15 node increases with the degree of the existing node. This property is sometimes called
16 the “rich gets richer” phenomenon. In scale-free networks, the degree distribution of
17 nodes follows a power law distribution. Scale-free networks have been found in many
18 networks such as power grids, the stock market, the Internet, and the spread of sexually
19 transmitted diseases (Albert et al. 1999; Matlis 2002).

20 Scale-free networks have different robustness characteristics compared to random net-
21 works in the presence of failures. In a random network, multiple node failures occurring
22 randomly will eventually trigger a collapse of the whole network, leaving most nodes
23 disconnected from each other. Scale-free networks have a large number of nodes with
24 only a few connections and only a very small number of highly connected node, so ran-
25 dom node failures will with high probability only disconnect one of the low connectivity
26 nodes; thus, such random failures do not effect the network and leave its structure intact.
27 However, if failures are not random and are targeted at the hubs, then the network can
28 collapse leaving it fragmented in smaller clusters. A highly connected hub becomes a
29 single point of failure, but knowledge of the scale-free structure of the network may
30 prove useful; for example, the spread of disease through a disease transmission network
31 can be halted by controlling and quarantining the hubs. The robustness properties of
32 scale-free networks is important for communication and collaboration networks because
33 information and resources can easily and quickly diffuse through the network even
34 though nodes are continuously joining and leaving the network.

35 36 37 **11.3 OSS DEVELOPMENT COMMUNITY**

38
39 Different member roles exist in an OSS project. Usually, an OSS project is initiated by
40 an individual or a small group of people with ideas they share for some intellectual,
41 personal, or business reasons (Godfrey and Tu 2000; von Hippel and von Krogh 2003).
42 Explanations of motivations for participation range from (1) the personal, including
43 intrinsic motivations, altruism, future rewards, and development of individual skills to
44 (2) the public, including the innovation and creation of public goods under the collective
45 action model (von Hippel and von Krogh 2003; Bitzer et al. 2004; Christley et al. 2004).
46 Then, the source code is made publicly available through the Internet. Interested people

01 download and use the code, report bugs, rewrite or modify code, suggest new features, and
02 submit patches. Adding new code is controlled by a small group of developers. Thus, we
03 can define member roles according to their contributions to the project. According to Xu
04 (2003), OSS members can be classified into either the user group or the developer group.
05 The user group includes passive users and active users. *Passive users* have no direct
06 contribution other than forming a larger user base. They just download code and use it for
07 their needs. *Active users* discover and report bugs, suggest new features, and exchange
08 other information by posting messages to forums or mailing lists. The developer group
09 can be further categorized into peripheral developers, central developers, core developers,
10 and project leaders. *Peripheral developers* irregularly fix bugs, add features, provide
11 support, write documents, and exchange other information. *Central developers* regularly
12 fix bugs, add features, submit patches, provide support, write documents and exchange
13 other information. *Core developers* extensively contribute to projects, manage CVS
14 releases and coordinate peripheral developers and central developers. *Project leaders*
15 guide the vision and direction of a project.

16 In this chapter, we assume that the OSS development community includes all of the
17 above members except passive users because passive users do not make direct software
18 development contributions and thus are not considered to be part of the development
19 community. Thus, as shown in Figure 11.2, our OSS development community includes
20 the following groups:

- 21
- 22 (1) *Project leaders*, who are also called project administrators;
 - 23 (2) *Core developers*, who regularly contribute to projects and manage CVS releases;
 - 24 (3) *Co-developers*, including both peripheral developers and central developers; and
 - 25 (4) *Active users*, who have some contributions except modifying code.
- 26

27 Because an individual may participate in multiple projects, that person can belong to
28 different groups in the development community (overlap in Figure 11.2). Our definition
29 of the OSS development community is considerably larger than the traditional closed-
30 source development team which approximately coincides with our project leader and
31 core developer groups. Our co-developer and active user groups are typically not part
32 of the development communities for traditional closed-source software, but are what has
33

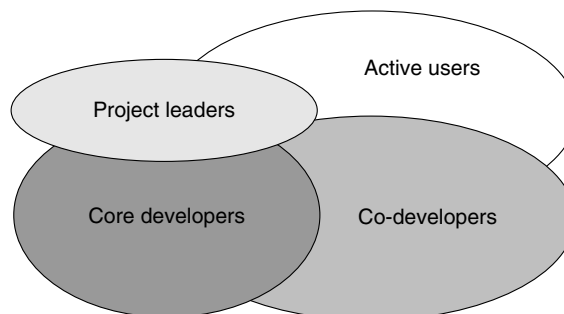


Figure 11.2 OSS development community classification.

01 been identified as end-user contribution to the product innovation process (von Hippel
02 and von Krogh 2003).

05 11.4 DATA COLLECTION AND EXTRACTION

07 One challenge in studying OSS development is to collect and extract data. Data col-
08 lection has proved to be tedious and time-consuming (Jensen and Scacchi 2004). Many
09 difficulties exist in collecting, cleaning, screening, and interpreting data (Howison and
10 Crowston 2004). In our previous studies (Madey et al. 2002; Gao et al. 2003; Xu et al.
11 2003), Web-bots were used to retrieve and extract data from Web pages at Source-
12 Forge.net. This process often took days, and frequently was plagued by incomplete and
13 missing data. In this section, we discuss a much improved data collection and extraction
14 process used in mining the SourceForge data.

15 There are many websites which host OSS projects. With over 100,000 projects and
16 over one million registered users as of middle 2005, SourceForge.net, sponsored by VA
17 Software, is the largest OSS development and collaboration site. It offers a centralized
18 place for OSS developers to control and manage OSS development by providing project
19 Web servers, trackers, mailing lists, discussion boards, software releases, etc. This site
20 provides highly detailed information about projects and developers, including project
21 characteristics, developers' activities, and "top ranked" developers. By studying these
22 websites, we can explore developers' behaviors and projects' growth.

23 We extracted data from a 2003 data dump obtained from SourceForge. The data dump
24 is derived from the "back-tier" relational database used to drive the SourceForge.net
25 website. The data dump contains information about the community, projects, and devel-
26 opers. We examined this data to characterize the entire SourceForge community, across
27 multiple numbers of projects, investigating behaviors and mechanisms at the project and
28 developer levels.

29 In the SourceForge data dump, information about the roles of developers on each
30 project is distributed over seven tables. Two roles, the project leader and core developer,
31 are explicitly defined and stored in a single table. The other two roles, co-developers and
32 active users, must be inferred and extracted from project activity data such as bug reports,
33 patch submissions, forum discussions, etc. The seven tables and their relationships are
34 shown in Figure 11.3. Table *groups* is the list of all projects. Tables *artifact_group_list*
35 and *artifact* contain members' activities such as bug tracking, patch submission, feature
36 requests, document writing, etc. Tables *forum_group_list* and *forum* reflect members'
37 participation in open discussion forums. Table *users* contains all users' information
38 including their identification numbers, the date they joined the community, etc. Table
39 *user_group* contains the relationships between projects and project leaders as well as
40 core developers.

41 By processing the above tables, we can identify members and their participation
42 activities for each project. The data extraction process is shown in Figure 11.4. We used
43 a three-step data integration and data reduction process on the data. Data integration
44 combines data from multiple sources into a coherent store. In the first step, we integrated
45 *artifact* and *forum* separately to create two tables – *artifact_activity* and *forum_activity*
46 to contain each member's activities in *artifacts* and *forums*; because some attributes

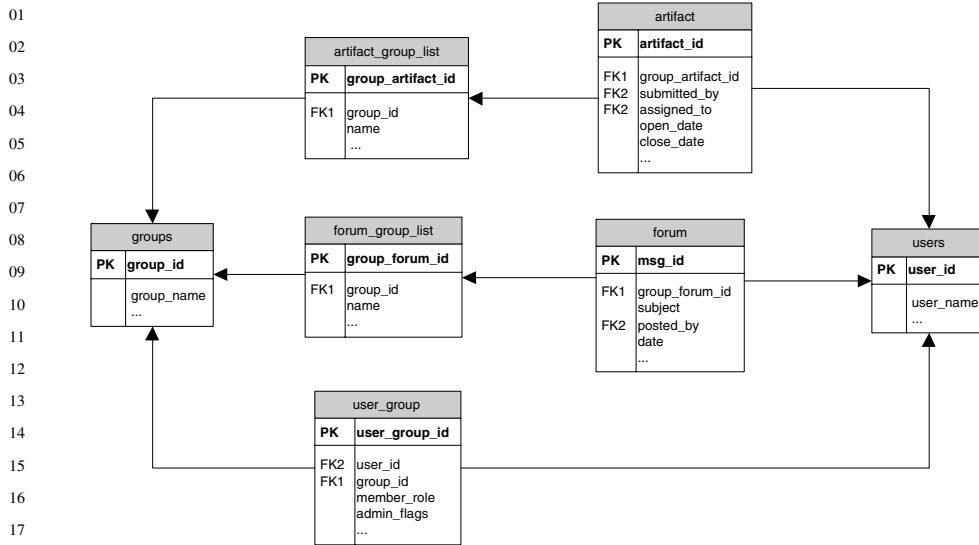


Figure 11.3 A subset of SourceForge database schema.

are different in *artifacts* and *forums*, we combined *artifact_activity* and *forum_activity* with *users* to get all members' activities for all projects, which is then recorded in *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.

11.5 DATA ANALYSIS

11.5.1 Analysis of OSS Member Distribution

As described in Section 11.3, we classified member roles as follows: project leaders are administrators in each project; core developers are developers listed in each project at SourceForge.net; co-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 every person into their highest ranked group. For example, if a person is listed as a project leader in one project and a core developer in another project, that person will be counted in project leader group. Figure 11.5 shows the distribution of member roles 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 the development community, which comprise 35% of the entire population, there are 10% project leaders, 5% core developers, 12% co-developers and

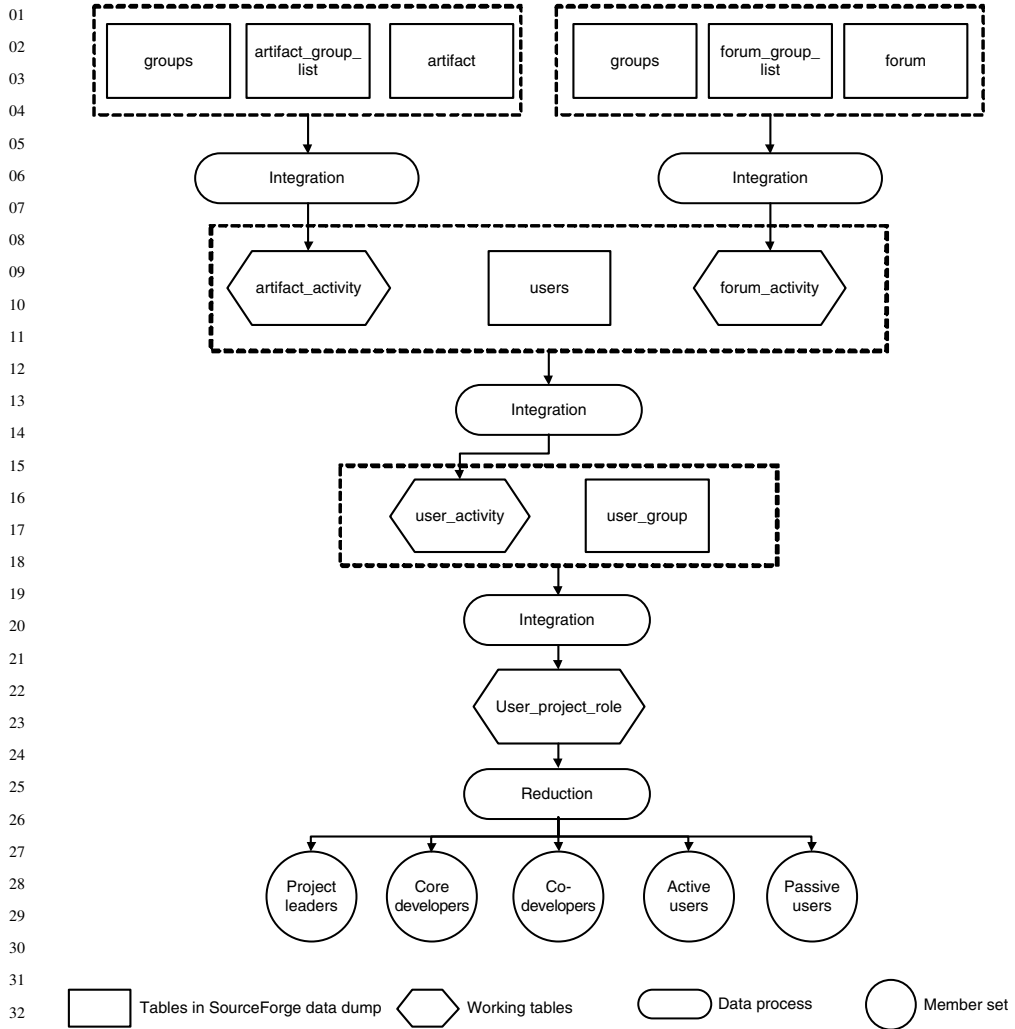


Figure 11.4 Data collection and extraction process.

8% active users. We observe 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.

If we count the number of developers on each project, including the people from our four member roles, and plot a histogram of the project size, we get a project size distribution with 261 different bins in the histogram. The project sizes range from 42,402 projects with one developer to a single project with 21,710 developers. To further investigate the relationship between the project size and the developer community, we divided the projects into three categories according to their size. Each category represents approximately one-third of those 261 histogram bins of project sizes: large projects,

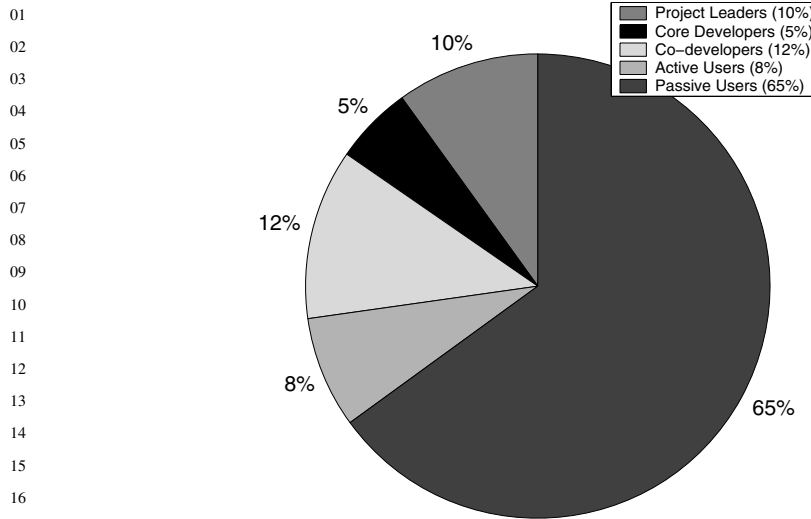


Figure 11.5 Distribution of SourceForge population.

middle projects, and small projects; then we analyzed the distribution of the four member roles for each project category. Large projects are those with more than 279 members; small projects contain less than 89 members, and middle projects have membership between 89 and 279. From a software engineering perspective, a small project team is usually composed of 1–7 developers, mid-sized teams go from 10 to 50 developers, and large projects teams have 50+ developers, so our project categorization appears to distort the notion of small, middle, and large project teams. However, because we take both co-developers and active users into account, our notion of a development community includes many more members than a traditional software engineering project team. Table 11.1 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, the share of co-developers and active users increases. 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%, respectively. 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.

11.5.2 OSS Network Topology

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};

Table 11.1 The distribution of large, middle and small projects

Community Size	Project Number	Project Leaders	Core Developers	Co-developers	Active Users
≤ 88	64,847	80,329 (47.8%)	34,659 (20.6%)	33,275 (19.8%)	19,941 (11.8%)
> 88 and ≤ 279	193	590 (2.1%)	1,703 (5.7%)	17,334 (60.3%)	9,124 (31.7%)
> 279	70	798 (0.9%)	2,576 (2.7%)	53,030 (55.8%)	38,593 (40.6%)

- Subset C = {project leaders} \cup {core developers} \cup {co-developers};
- Subset D = {project leaders} \cup {core developers} \cup {co-developers} \cup {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.

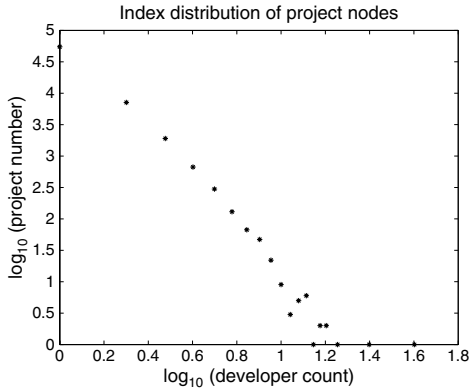
11.5.2.1 Degree distribution

We hypothesize that the OSS community is a scale-free network because it may be growing and self-organizing due to sequential growth and preferential attachment. 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 in projects which attract them. Some projects are more popular than others, and those projects tend to attract more developers and users; thus in this network, there exists a preferential attachment of developers to projects.

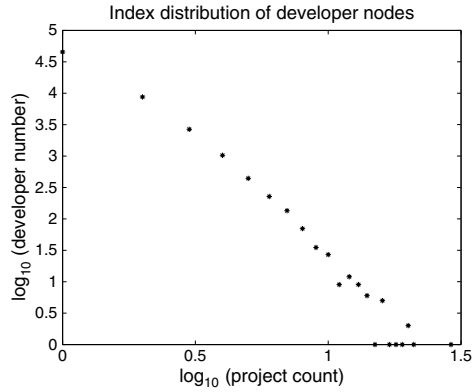
To support our hypothesis, we computed the degree distributions of SourceForge project and developer networks, shown in Figure 11.6. All left subgraphs represent the distributions of the log-log transformation of project size frequencies. The right subgraphs display the distributions of the log-log transformation 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 in one project (45,261 in subset A, 63,103 in subset B, 1,05,234 in subset C, and 1,13,999 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 11.2 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. We further use Pajek (2004) to draw a cluster of the project network. As shown in Figure 11.7, this cluster includes a random sampling of 2495 out of 65,110 total projects. The size of each node represents the logarithmic value of its developer count, and edges indicate common developers with

01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

(a)



(b)



(c)

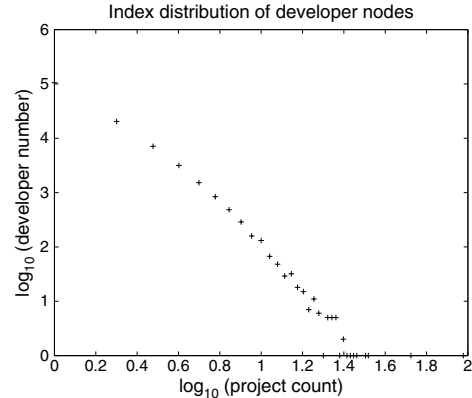
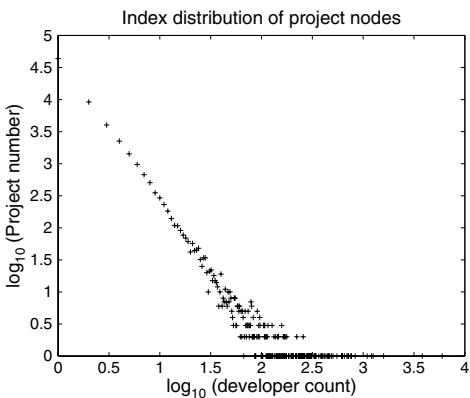
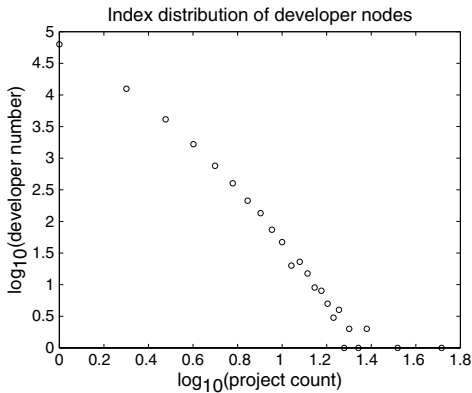
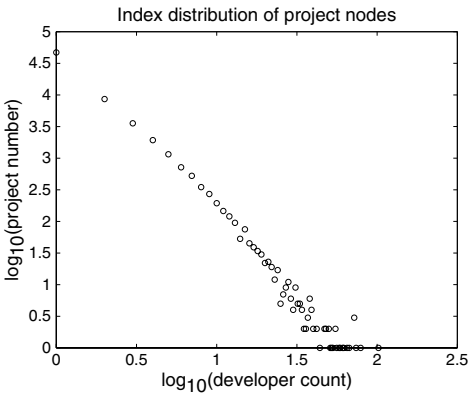


Figure 11.6 The SourceForge project and developer community scale free degree distributions: (a) subset A; (b) subset B; (c) subset c; (d) Subset D.

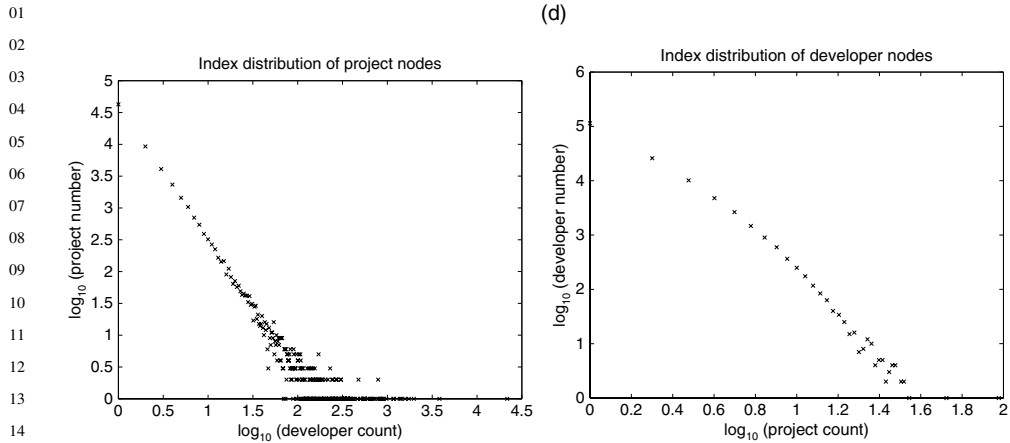


Figure 11.6 (Continued)

other projects in the random sampling. While visually deceiving, the figure shows the project network has a large number of small nodes, i.e. projects with few developers, compared to a small number of large nodes, projects with many developers; this is an observed property of scale-free networks.

11.5.2.2 Diameter

Using Equation 11.2, we calculated the average shortest path length of the four community subsets. As shown in Table 11.3, if the community is composed of only project leaders, the average shortest path length computed using Equation 11.2 in this network is infinity. This reflects the fact that the project leader network is highly disconnected. Perhaps project leaders are too busy to join other projects? In subset B, which includes both project leaders and core developers, the average shortest path shrinks to a length of about 10 for a total of 83,118 members; the introduction of core developers makes the social network much more connected, creating a large component of 15,091 members. Project leaders and core developers are important in OSS networks because there are more information exchange within them and they reinforce the cooperation among members in the same project group. Thus, they work as strong ties in OSS networks. Core developers are more willing (or able?) to participate on other projects. Co-developers and active users have a much greater likelihood to join multiple projects; they act as

Table 11.2 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

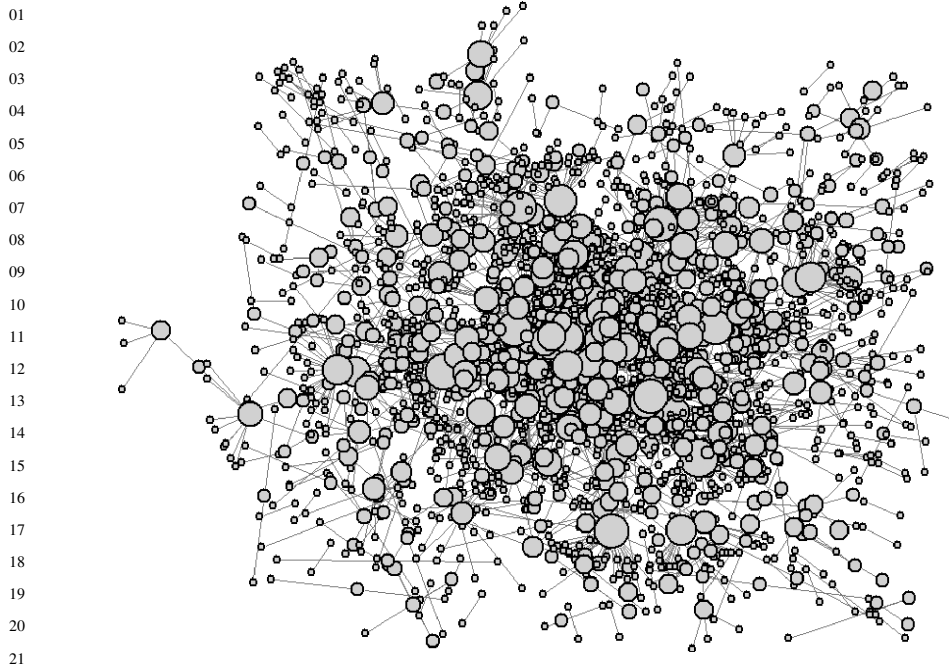


Figure 11.7 A scale-free project network (unipartite graph): actual graph of 2,495 randomly selected projects. Node size is logarithmically proportional to the number of developers, and edges represent a common developer between projects.

Table 11.3 The properties of the development community

Property	Subset A	Subset B	Subset C	Subset D
Size	58,651	83,118	1,39,570	1,61,691
z_1	1	6	508	3,241
z_2	1	17	13,398	31,998
Diameter	Inf.	10.2	2.7	2.7
Clustering coefficient	0.8406	0.8078	0.8867	0.8297
Largest project cluster	737	15,091	30,794	40,175
2nd Largest project cluster	197	34	20	20
# of Project clusters	43,826	34,280	27,983	21,659

the weak ties in the development community, and the result is that the social network becomes even more connected. Subset C and subset D both have an average shortest path of about 3; that means, on average, it takes only 3 links for any member in the SourceForge development community to reach another member in the community. With the participation of co-developers and active users, the degree of separation between any two members is significantly decreased. By understanding this fact, certain phenomenon can be understood in terms of the short distance between members in the OSS community; information such as ideas and discussions can spread fast in the OSS development

01 community because the information only has to travel a few links to reach anybody
02 in the network. Likewise, because many of the communication forums for projects are
03 broadcast mediums, information spread by members reach many other members with
04 just a single message post. This property is described in the following subsection.

05 06 *11.5.2.3 Clusters: Sizes and numbers*

07 Two projects are linked if they share a community member. All linked projects form a
08 project cluster (See Figure 11.1 for an example). Each project cluster has a corresponding
09 developer cluster, so we limit our discussion in this section to project clusters. All four
10 subsets of the SourceForge development community contain many separated clusters
11 (shown in Table 11.3). Also in Table 11.3, we see that the largest cluster is much
12 bigger than the second largest cluster and grows as we move from subset A to subset D.
13 However, the second largest cluster decreases with the increase of the community size.
14 The reason is that some clusters are linked to become part of the largest cluster. The
15 larger a cluster is, the more possible it will attach to the largest cluster as the community
16 size increases.

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

24 25 *11.5.2.4 Clustering coefficient*

26 We use Equation 11.3 to get the approximate clustering coefficients of the four subsets of
27 the SourceForge community. The clustering coefficient tells us how many of a member's
28 collaborators are collaborators with each other. As shown in Table 11.3, the clustering
29 coefficients of all four subsets are above 0.8. The high clustering coefficients are not
30 surprising because members are fully connected in each project.

31 32 33 **11.5.3 Discussion**

34 We found a small world phenomenon, the small diameter and the high clustering coeffi-
35 cient, in the SourceForge development community. The small distance results from the
36 fact that a member may participate in multiple projects. In this way, the member con-
37 nects separate clusters and creates a path among members in those clusters. Moreover,
38 a large percentage of members participate in one project (70.5% in subset D). This fact
39 explains the observed high clustering coefficient; the high clustering coefficient may also
40 be an artifact of how we construct our social network with all developers on the same
41 project connected to each other. Furthermore, the power law distribution found in the
42 SourceForge OSS network supports the claim that it is a scale-free network. The OSS
43 projects are often developed by collaborating volunteers, who sequentially join projects
44 based on their interests.

45 In previous studies (Gao et al. 2003; Madey et al. 2004), we observed the power law
46 distribution and the small world phenomenon in the SourceForge project leaders and

01 core developers community (classified as subset B in this chapter). In this chapter, we
02 observe that with the participation of co-developers and active users, while the cluster is
03 increasing, the diameter is much smaller (only about 3 links between pairs of community
04 members). Co-developers and active users are weak ties in the whole network. Weak ties
05 are often more important in spreading information or resources because they tend to serve
06 as bridges between otherwise disconnected social groups (Granovetter 1973). This fact
07 supports the assertion that co-developers and active users play a crucial role in connecting
08 the SourceForge development community. Their existence can make communication
09 flow faster throughout the whole OSS community. This fast communication may be
10 an example of a self-organizing, optimal reallocation of resources. For example, our
11 SourceForge data contains 676 text editor projects. Of all developers on those projects,
12 about 50% of developers are on the top six largest projects. One reason might be due to
13 the short communication path in the development community through which people can
14 find projects which attract them (e.g., are better organized, have more compatible goals,
15 or are making better progress). The feature of fast communication in OSS development
16 community may be a factor of its success because closed-source software development
17 does not typically have co-developers, and active users may not be in close contact with
18 developers.

11.6 CONCLUSIONS

21
22
23
24 In this chapter, we applied social network analysis to study the OSS development com-
25 munity at SourceForge. Based on a SourceForge 2003 data dump, we performed quanti-
26 tative analysis on the SourceForge OSS developer and project networks. Using statistical
27 analysis, we found that different sized projects have different member distributions.
28 Large projects consist mainly of co-developers and active users, while project leaders
29 and core-developers are the main part of small projects. Furthermore, we conducted
30 topological analysis on four subsets of the OSS development community. Properties
31 of the community network show that the SourceForge OSS development community
32 is a self-organizing system which obeys scale-free behaviors. Moreover, small world
33 phenomenon, the small average distance and the high clustering coefficient, exists in the
34 community. We identify the important effect of co-developers and active users because
35 users because their addition can turn the OSS community into a fast communication net-
36 work. Our research provides useful information of the underlying structure and evolution
37 of the OSS community, and the diffusion of information between projects.

38 The work in this chapter was part of an ongoing OSS study at the time of writing.
39 Future work will focus on the simulation of the OSS developer network based on the
40 data and analysis in this chapter. Future research includes testing our interpretations and
41 constructing fitness models. This work can be done by incorporating some social network
42 theories into an agent-based model so that we can perform computer experiments for
43 hypothesis testing. Furthermore, our analysis is based only on SourceForge.net. We will
44 explore some other OSS project sites (e.g. Savannah, Linux, Mozilla, and Apache) to
45 test if the development community characteristics found in SourceForge are presented
46 in those sites.

ACKNOWLEDGMENTS

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

REFERENCES

- Albert, R. and Barabasi, A.-L. 1999. Emergence of scaling in random networks. *Science*, 286(5439), 509–512.
- Albert, R. and Barabasi, A.-L. 2002. Statistical mechanics of complex networks. *Review of Modern Physics*, 74(1), 47–97.
- Albert, R., Barabasi, A.-L., and Jeong, H. 1999. Diameter of the world wide web. *Nature*, 401(9), 130–131.
- Balthrop, J., Forrest, S., Newman, M., and Williamson, M. 2004. Technological networks and the spread of computer viruses. *Science*, 304(5670), 527–529.
- Bitzer, J., Schröder, P. J. H., and Schrettl, W. 2004. Intrinsic Motivation in Open Source Software Development. Free University Discussion Paper No. 2004/19.
- Bollier, D. 1999. The Power of Openness: Why Citizens, Education, Government and Business should Care About the Coming Revolution in Open Source Code Software. Working Paper, available at: <http://cyber.law.harvard.edu/home/uploads/194/1999-02.pdf>.
- Burt, R. 1992. *Structural Holes: the Social Structure of Competition*. Cambridge, US: Harvard University Press.
- Christley, S., Gao, Y., Madey, G., and Xu, J. 2004. Public goods theory of the open source development community using agent-based simulation. In: *Agent 2004 Conference Proceedings*, Chicago, USA, pp. 633–640.
- Crowston, K. and Scozzi, B. 2002. Open source software projects as virtual organizations: Competency rallying for software development. *IEE Proceedings Software*, 49(1), 3–17.
- Feller, J. and Fitzgerald, B. 2002. *Understanding Open Source Software Development*. London, UK: Addison-Wesley.
- Gabarro, J. 1990. Intellectual teamwork: Social and technological foundations of cooperative work. *The Development of Working Relationships*. Edited by Galegher, J., Kraut, R. E. and Egido C., New Jersey Hove, US: Lawrence Erlbaum Associates, pp. 79–110.
- Godfrey, M. W. and Tu, Q. 2000. Evolution in open source software: A case study. In: *Proceedings of the 2000 International Conference on Software Maintenance (ICSM 2000)*, San Jose, US, pp. 131–142.
- Gao, Y. Q., Freeh, V., and Madey, G. 2003. Analysis and modeling of the open source software community. In: *Proceedings of North American Association for Computational Social and Organizational Science Conference (NAACSOS 2003)*, Pittsburgh, US.
- Granovetter, M. 1973. The strength of weak ties. *American Journal of Sociology*, 78(6), 1360–1380.
- Gulati, R. and Gargiulo, M. 1999. Where do interorganizational networks come from? *American Journal of Sociology*, 104(5), 1439–1493.
- Haythornthwaite, C. 2001. Tie strength and the impact of new media. In: *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS 2001)*, Maui, Hawaii, p. 1019.
- Howison, J. and Crowston, K. 2004. The perils and pitfalls of mining sourceforge. In: *Proceedings of the 1st Internatioal Workshop on Mining Software Repositories(MSR 2004)*, International Conference on Software Enginnering (ICSE 2004), Edinburgh, Scotland, pp. 7–11.
- Jensen, C. and Scacchi, W. 2004. Data mining for software process discovery in open source software development communities. In: *Proceedings of the 1st International Workshop on*

- 01 *Mining Software Repositories (MSR 2004)*, International Conference on Software Engineering
02 (ICSE 2004), Edinburgh, Scotland, pp. 96–100.
- 03 Kim, E. E. 2003. An Introduction to Open Source Communities. Blue Oxon Group, available at:
04 <http://www.blueoxen.com/research/00007/BOA-00007.pdf>.
- 05 Lopez-Fernandez, L., Robles, G., and Gonzalez-Barahona, J. 2004. *Applying Social Network*
06 *Analysis to the Information In CVS Repositories*, Edinburgh, UK, pp. 101–105.
- 07 Madey, G., Freeh, V., and Tynan, R. 2002. The open source software development phenomenon: An
08 analysis based on social network theory. In: *Proceedings of Americas Conference on Information*
09 *Systems (AMCIS 2002)*, Dallas, US, pp. 1806–1813.
- 10 Madey, G., Freeh, V., and Tynan, R. 2004. *Modeling the F/OSS Community: A Quantitative*
11 *Investigation, Free/Open Source Software Development*, Edited by Koch, S., Hershey: Idea
12 Publishing, pp. 203–220.
- 13 Matlis, J. 2002. Scale-free networks. *ComputerWorld*, 4 November.
- 14 McConnell, S. 1999. Open-source methodology: Ready for prime time? *IEEE Software*, 16(4),
15 6–8.
- 16 McGrath, J. 1984. *Groups: Interaction and Performance*. Upper Saddle River, US: Prentice Hall.
- 17 Milgram, S. 1967. The small world problem. *Psychology Today*, 1(1), 60–67.
- 18 Newman, M. E. J. 2001. Scientific collaboration networks: II. shortest paths, weighted networks,
19 and centrality. *Physical Review E*, 64(1), 016132.
- 20 Newman, M. E. J., Watts, D. J., and Strogatz, S. H. 2002. Random graph models of social
21 networks. In: *Proceedings of the National Academy of Sciences of the United States of America*
22 *2002*, 99 (19 February), pp. 2566–2572.
- 23 Pajek 2004. <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>.
- 24 Powell, W., White, D., Koput, K., and Oweb, S. J. 2005. Network dynamics and field evolution:
25 The growth of interorganizational collaboration in life sciences. *American Journal of Sociology*,
26 110(4), 1132–1205.
- 27 Raymond, E. 1998. The cathedral and the bazaar. *First Monday*, 3(3).
- 28 Sourceforge 1999. <http://www.sourceforge.net>.
- 29 Uzzi, B. 1996. The sources and consequences of embeddedness for the economic performance of
30 organizations: The network effect. *American Sociological Review*, 61(4), 674–698.
- 31 von Hippel, E. and von Krogh, G. 2003. Open source software and the “private-collective”
32 innovation model: Issues for organization science. *Organization Science*, 14(2), 208–223.
- 33 Watts, D. and Strogatz, S. 1998. Collective dynamics of “small-world” networks. *Nature*, 393(6),
34 440–442.
- 35 Wellman, B. 1988. Social structures: A network approach. *Structural Analysis: From Method and*
36 *Metaphor to Theory and Substance*, Edited by Wellman, B. and Berkowitz, S. D., Cambridge,
37 UK: Cambridge University Press, pp. 19–61.
- 38 White, H. 2002. *Markets from Networks: Socioeconomic Models of Production*. Princeton,
39 NJ: Princeton University Press.
- 40 Xu, N. 2003. An Exploratory Study of Open Source Software based on Public Project Archives.
41 Master’s thesis, John Molson School of Business, Concordia University.
- 42 Xu, J., Gao, Y., and Madey, G. 2003. A Docking Experiment: Swarm and Repast for
43 Social Network Modeling. Working Paper, available at: [http://www.cse.nd.edu/~oss/Papers/](http://www.cse.nd.edu/~oss/Papers/DockingSwarmPaper.pdf)
44 [DockingSwarmPaper.pdf](http://www.cse.nd.edu/~oss/Papers/DockingSwarmPaper.pdf).
- 45 Xu, J., Huang, Y., and Madey, G. 2003. A research support system framework for web data
46 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, pp. 37–41.
- 47 Xu, J., Gao, Y., Goett, J., and Madey, G. 2003. A multi-model docking experiment of
48 dynamic social network simulations. In: *Agent 2003 Conference Proceedings*, Chicago, USA,
49 pp. 129–140.