Modeling the Free/Open Source Software Community: A Quantitative Investigation

Gregory Madey
Computer Science & Engineering
University of Notre Dame
Phone: 574-631-8752
Fax: 574-631-9260
gmadey@nd.edu

Vincent Freeh
Computer Science
North Carolina State University
Phone: 919-513-7196
Fax: 919-515-7925
vin@cs.ncsu.edu

Renee Tynan
Department of Management
University of Notre Dame
Phone: 574-631-6764
Fax: 574-6316764
rtynan@nd.edu

Modeling the Free/Open Source Software Community: A Quantitative Investigation

ABSTRACT

This chapter reports the latest results from an ongoing study of Free/Open Source Software (F/OSS) Development at the community level. Publicly available data about F/OSS projects, developers, processes, and their relationships have been collected from F/OSS hosting sites, including SourceForge and others. Numerous descriptive statistics, including the existence of many power-law relationships, are presented. The F/OSS community is modeled as a collection of ad hoc, social networks consisting of heterogeneous agents, self-organizing into projects and clusters of projects. A computer simulation of the F/OSS community model is developed using SWARM, an agent-based simulation toolkit. Empirical data is used to parameterize the simulation, which in turn is used to investigate hypotheses about processes and mechanisms leading to F/OSS community formation. The quantitative data, the model, and the simulation offer insight into F/OSS project coordination and communication.

Keywords: free/open source software, social networks, self-organization, agent-simulation

INTRODUCTION

Our investigation aims to increase the understanding of the Free/Open Source Software (F/OSS) movement by providing a quantitative investigation of the network properties of the community. In some ways the F/OSS movement is a prototypical example of a self-organizing system (R. Axelrod, M. Cohen, 1999; A.-L. Barabasi, 2002; A. L. Barabasi, Albert, R., Jeong, H, 2000; Faloutsos, 1999; Holland, 1998; Huberman, 1999; Kuwabara, 2000; Madey, 2002a, 2002b, 2002c), but it also possesses some unique properties that may affect the development of the network.

The lack of central planning or control in F/OSS projects challenges conventional economic assumptions, turns conventional software engineering and project management principles inside out, threatens traditional proprietary software business strategies, and presents new legal and government policy questions regarding software licensing and intellectual property. Understanding F/OSS is a far from an academic enterprise—F/OSS is a major component of the IT infrastructure enabling global e-commerce. Free/Open Source Software includes BIND, sendmail, Apache, Linux, INN, GNU utilities, MySQL, PostgreSQL, and Perl, all critical elements of the internet.

In this paper we describe a social network investigation of almost 60,000 F/OSS projects at SourceForge (2003), a web-based project support site sponsored by VA Software. With permission, we collected data on developers and projects over time at SourceForge. We analyzed the data using cluster analysis to learn more about the structure of the developer-project network, and then used the data to create a model of the network for agent-based simulations. We ran simulations of the network using the model to validate the model and to discover emergent properties of the network that can only be observed by studying the network growth over time.

We find that both project size and the number of projects developers are working on can be modeled with the power law relationship, providing empirical evidence for the claim the F/OSS community is a self-organizing system. We also find that the cluster size of connected developers fits the power law, if the largest and most connected cluster, comprising almost 35% of the developers, is removed, and we discuss the possible causes behind this dual structural nature of the network. Finally, extending Barabasi's construct of a network fitness component (A.-L. Barabasi, 2002), we find that a dynamic lifecycle fitness parameter for projects is necessary to best model the project data at SourceForge.

We begin with a discussion of social network theory and the utility of using simulation modeling to understand self-organizing systems. We then describe our data collection, cluster analysis results, model development, and simulation results, followed by a discussion of the theoretical and practical ramifications of our results and directions for future research.

MODELING SOCIAL NETWORKS

Why should we invest the effort to do a quantitative simulation of the F/OSS network? The rationale behind such an investigation is that the F/OSS community is a social network that possesses several prototypical features of complex systems, systems that prior investigations have shown possess temporal and emergent properties that can be discovered only through modeling the system as a whole over time. For example, Axelrod (1984) found that certain types of "guarded cooperation" emerged as the most effective strategies for maximizing long term joint outcome of dyads in a community, a result that could not have been obtained without simulation modeling.

Social network theory seeks to understand the network properties of people in relation to one another. Social network theory models persons as vertices or nodes of a graph and their relationships as edges or links of the graph (A.-L. Barabasi, 2002; Jin, 2001; Wasserman, 1994; D. Watts, 1999; D. Watts, Strogatz, S. H., 1998). Thus two persons are directly connected if they have a relationship (e.g., friendship) with each other; they then are one edge away from one another, a distance of one. More distant relationships are modeled as paths through the graph; a "friend of a friend" is two edges away. For example, displayed in Figure 1 is a diagram of a social network composed of four individuals, labeled A through D and represented as vertices of the graph. In this social network, individuals A through C are fully connected by edges, representing a "circle of friends" or a clique. Vertices C and D are connected by one edge and since this is the only path between C and D, that edge is called a weak tie. The number of edges attached to a vertex is called its index value. Vertex C, called a hub (or later in this chapter, a linchpin), is critical in social networks because its removal would break the network into two disconnected components. Hubs are also important because they often share the edges that weakly tie cliques together, and those weak ties have been shown to be most important in the spread of information through a network (A.-L. Barabasi, 2002; Granovetter, 1973; D. Watts, 1999). Several studies reveal an interesting phenomenon present in many of these social networks; most persons are very few links from any other person – the Small World Phenomenon (D. Watts, 1999; D. Watts, Strogatz, S. H., 1998). This idea was popularized in the play (and movie) *Six Degrees of Separation* (Guare, 1990) which claims that all persons in the world are at most six friendship links away.

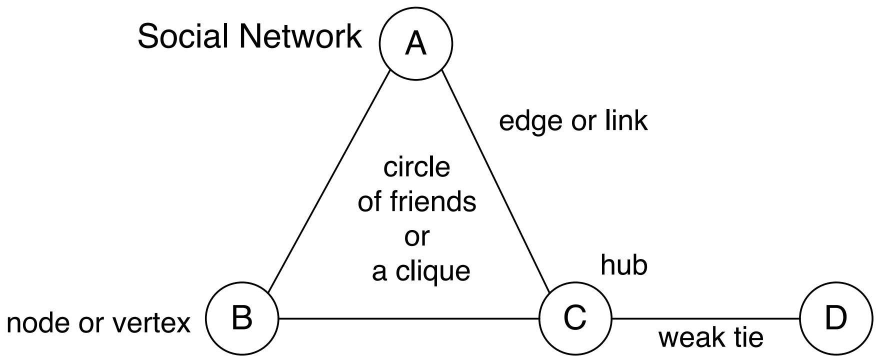

Figure 1: Components of a social network

Collaboration networks are variations of social networks, where the relationships are collaborations, e.g., actors in movies (Tjaden, 1996; D. Watts, 1999), or co-authors on research papers (A. L. Barabasi, Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Viscek, T., 2001; Newman, 2001). Often entire populations are connected into one large cluster with high characteristic clustering coefficients (D. Watts, 1999). Highly prolific actors or authors are linchpins in collaborative networks. Linchpin actors or researchers play key roles in bridging disparate groups into one large cluster.

Social networks and collaboration networks have another interesting property in common; they are often self-organizing systems, forming patterns of connectivity that emerge in a bottom-up process based on local interactions. Such systems displaying self-organizing behaviors and emergence based primarily on local interactions are the subject of study called complex systems (Cowan, 1999; Johnson, 2001; Kelly, 1994).

Social networks, collaborative networks, and other self-organizing systems (e.g., the Internet, corporation sizes, cities, economic systems, word usage in languages, ecosystems, lines-of-code in programs) often have another interesting property; they have highly skewed distributions, which under a log-log transformation results in a linear relationship. This is called a power-law relationship. Power-law relationships have been reported for the Internet (Albert, 1999; A. L. Barabasi, Albert, R., 1999; A. L. Barabasi, Albert, R., Jeong, H, 2000; Faloutsos, 1999; Huberman, 1999), sizes of U.S. firms (Axtell, 2001), city size distributions (Pumain, 1997), ecosystems (Jorgensen, 1998), word rank in languages and writing (Schroeder, 1991) and other systems for which emergent properties are of interest. One major hypothesis about the source of power law relationships is nonrandom attachment of nodes (called preferential attachment (A.-L. Barabasi, 2002; A. L. Barabasi, Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Viscek, T., 2001;

Newman, 2001)). Preferential attachment refers to the fact that some vertices tend to attract new edges over time in an evolving network with greater probability than other vertices. Unlike random networks, where all vertices have equal probability of attracting new edges as the network grows, in collaboration networks the probabilities tend to be in part proportional to the index of the vertex, resulting in a "rich-get-rich" phenomenon. They also may have different intrinsic "fitness" that can outweigh the attractiveness for new edges caused by index value of the graphs' vertices (A.-L. Barabasi, 2002). This property helps model the fact that "young upstarts" can attract new edges with greater probability that older vertices which typically have greater index values because they have been part of the network for a longer time.

We analyze the Free/Open Source Software phenomenon by modeling it as a collaborative social network. The developers are vertices of a network and joint membership on an open source project is a collaborative link between the developers as shown in Figure 2.

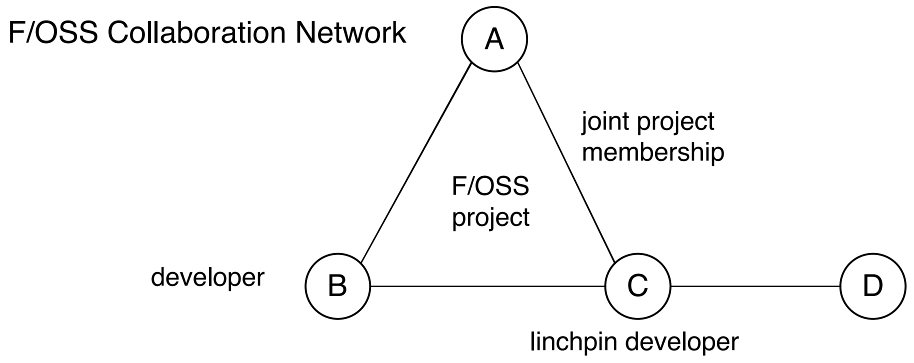

Figure 2: The Free/Open Source Software (F/OSS) community modeled as a collaborative social network.

The F/OSS development community is highly decentralized and is a volunteer effort where developers freely join projects that they find appealing – all attributes of typical self-organizing systems. We hypothesize that the open source community displays power-law relationships in its structure and grows with a preferential attachment process modified by a fitness property. Our empirical analysis of structural data collected from SourceForge suggests that this is the case. These results, along with additional unique properties of the F/OSS community discovered using agent-based simulation and data-mining techniques are described in the next section.

DATA COLLECTION

Because Free/Open Source Software is typically developed by global virtual teams, most projects use the Web and the Internet to facilitate their work. This provides research opportunities for the acquisition of data directly from online data sources as has been used by several prior studies, including Mockus et al (2002), Ghosh and Prakask (2000), and Krishnamurthy (2002). We collected data on F/OSS developers and projects from January, 2001 to March, 2003 at SourceForge.net, a large project management website supporting F/OSS development with project management tools, bug tracking software, mail list services, discussion forums, and version control software (SourceForge, 2003). We believe that SourceForge is representative of a wide cross-section of the F/OSS community. While SourceForge does not support many large high profile projects that maintain their own developer sites (e.g., Apache, Perl, sendmail, and Linux), some large projects have moved to SourceForge, including Samba, and there are many smaller projects that have not joined SourceForge.

Each project in SourceForge has a unique project number, and each developer is assigned a unique ID when they register with SourceForge. We collected data linking developers to the projects they are working on, over time. The data thus consisted of a table of records with two fields, project number and developer ID. Because projects can have many developers and developers can be on many projects, neither field is a unique primary key, and thus, the composite key composed of both attributes serves as the primary key. After the data collection, the data was completely anonymized by an algorithm that shuffles the ID values.

To collect the data, we created a web crawler to traverse the SourceForge website monthly, with the permission of SourceForge. All project home pages in SourceForge have a similar top-level design, with most of them dynamically generated from a database. A simple shell script fetches each project's developer page. It parses the HTML, and extracts the names of the developers. A python program parses the HTML source, giving as output one line for each developer, containing the project number and the developer's ID. For March 2003, the table generated with this method exceeded 110,000 records in the form shown in Table 1. Thus, in the example data displayed in Table 1, there are 6 projects ranging is size from 1 developer to 3 developers. All the data was stored in a relational database by month. The combined data for all months of collected data exceeds 2 million records. Identification of clusters, or connected groups of developers, was implemented using a version of Prim's spanning tree algorithm (Corfmen, 2001).

```
9001|dev378
9001|dev8975
9001|dev9972
9002|dev27650
9005|dev31351
9006|dev12509
9007|dev19395
9007|dev4622
9007|dev35611
9008|dev7698
```

Table 1: Typical data (anonymized) retrieved from SourceForge associating projects and developers

RESULTS AND ANALYSIS

We submitted the data to three types of analyses. First, we examined the cluster properties of the network using developers as nodes, with an edge existing between nodes if both developers are on the same project. This representation is analogous to using research paper authors as nodes and joint authorship as a link in a collaboration network (A. L. Barabasi, Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Viscek, T., 2001; Newman, 2001). As can be seen in Figure 3, a randomly selected small cluster from one monthly data set from Sourcforge, linchpin developers (developer 58 and developer 46), or developers who link two projects together, play an important role in linking clusters together (Granovetter, 1973). Our analysis showed that the developer

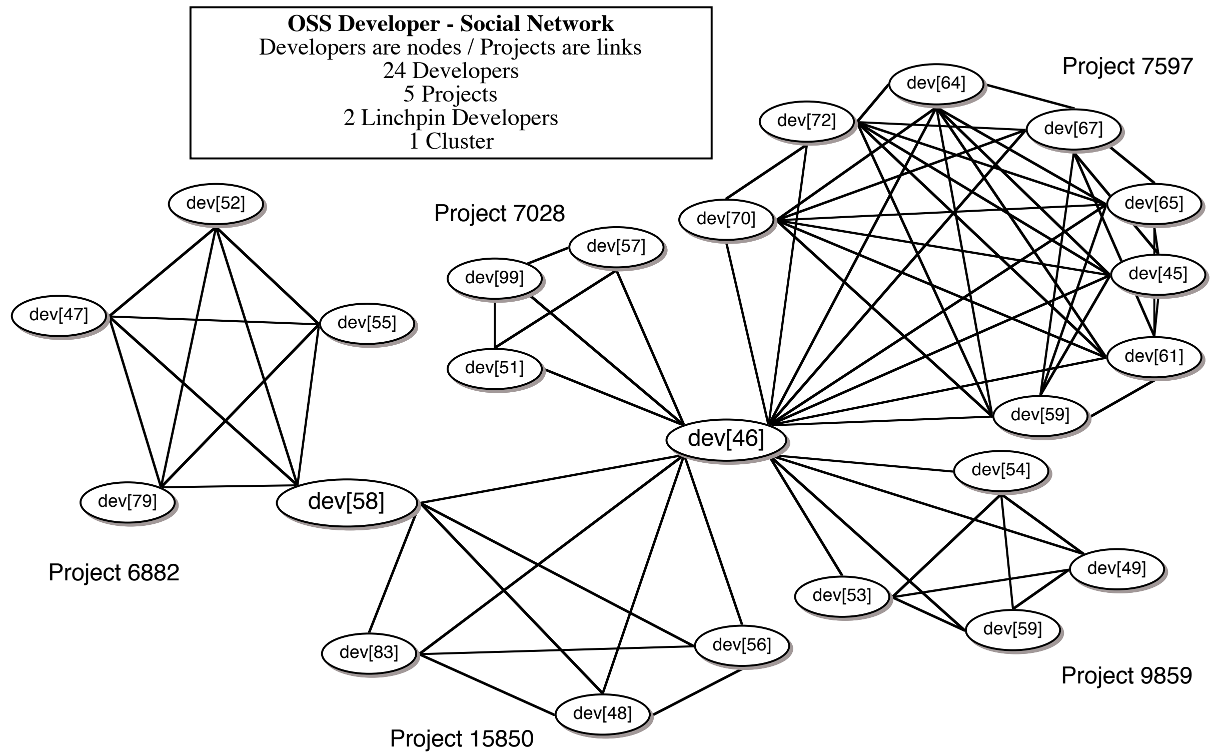

Figure 3: Social network consisting of one connected cluster, 24 developers, 5 projects, and 2 linchpin developers (dev[58] and dev[46]).

collaboration network at SourceForge fits the power-law model, as determined by ordinary least squares (OLS) regression in log-log coordinates. As shown in Figure 4, the distribution of the number of projects per developer has a power-law distribution, with the solid line the OLS regression line through the data (adjusted R-squared greater than .95). The data displayed in Figure 4 was collected from SourceForge in March 2003. Equivalent results are seen in all monthly data for distributions of the number of projects developers join.

The second analysis used projects as nodes and developers as edges, with two projects linked together if they share a developer in common. Project size frequency showed equivalent results to the number of projects developers join, fitting the power-law and with R-squared greater than .95 displayed in Figure 5.

We also conducted an analysis on the size of clusters, with a cluster defined as a connected component with a path between all developers. Cluster analysis of the SourgeForge data from March 2003 identified the presence of one large cluster consisting of 28,394 developers, about 34.6% of the developers at SourceForge. The next largest cluster was of size 75, with sizes ranging down to one (i.e., those developers on single-member projects). Projects with only one

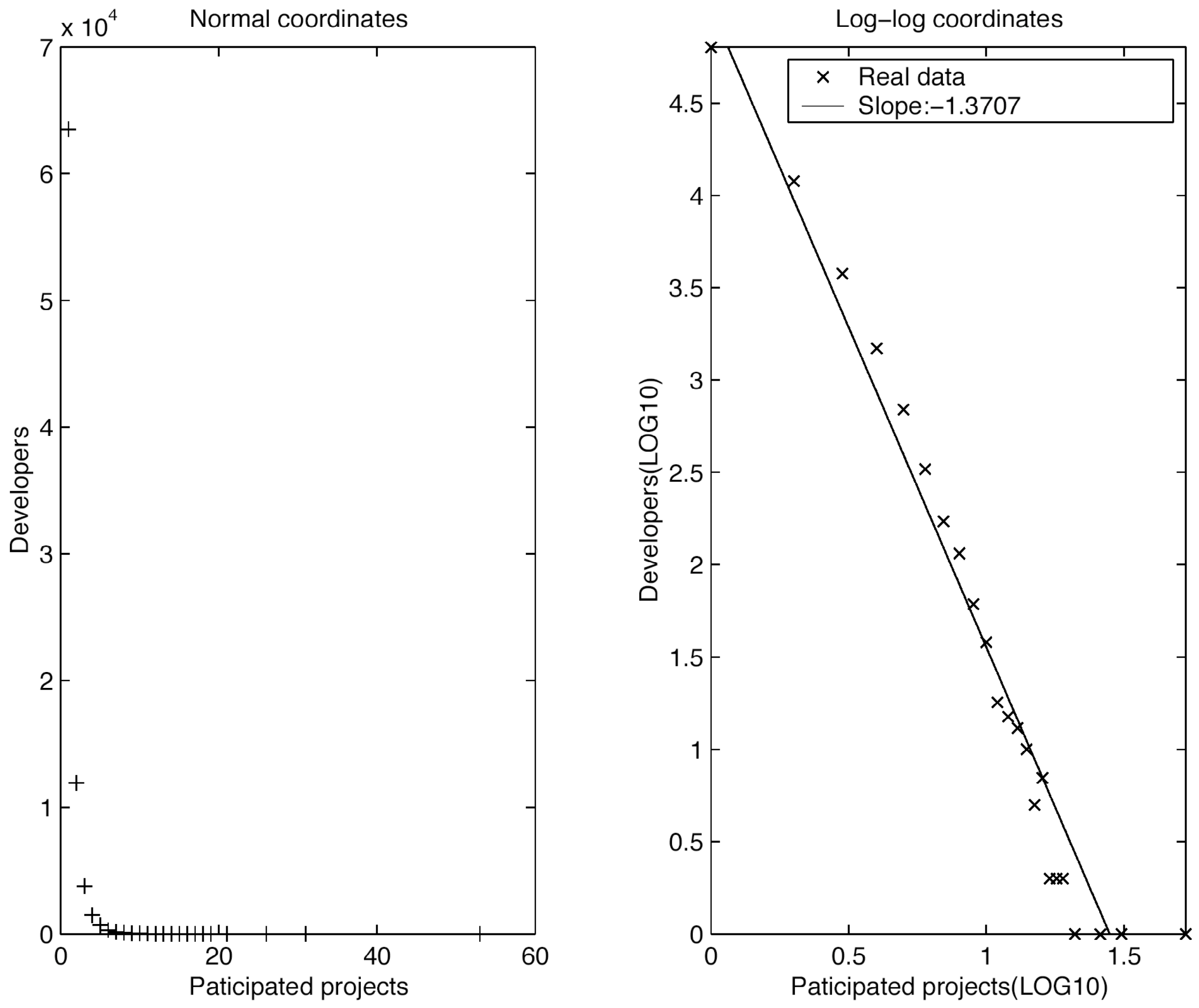

Figure 4: Distribution of developer index values on linear axes (left) and log axes (right)

developer occurred most frequently (in addition, developers on only one project was the most frequent value for the number of projects joined by a developer, with a large intersection between the two groups).

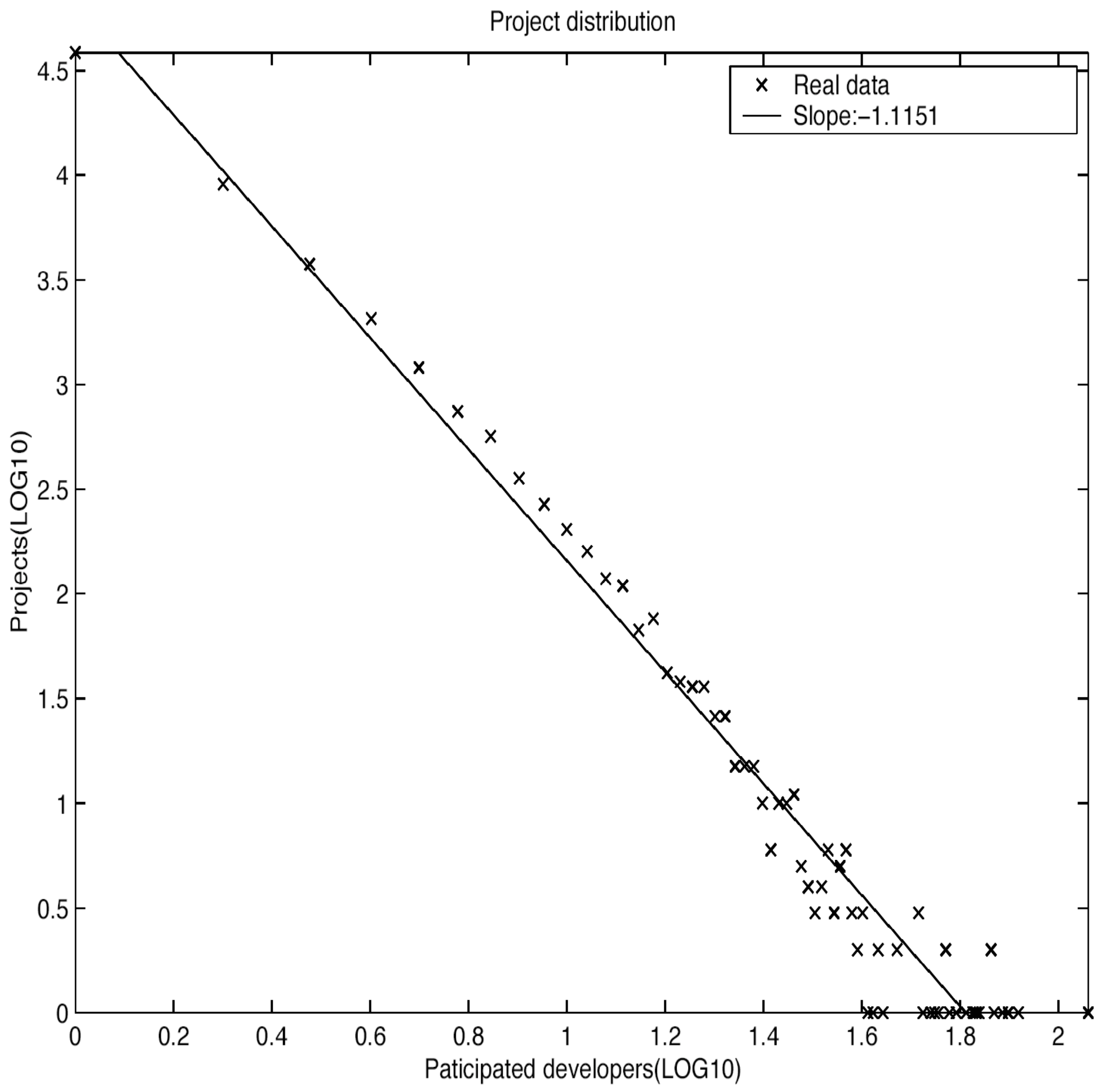

Figure 5: Power law distribution on size of projects determined by number of participating developers

The large super-cluster was an outlier relative to the rest of the data points. If this cluster is removed, the cluster components fit the power-law, with a log-log transformation giving a good fit using regression (adjusted R-squared = .93) shown in Figure 6.

In Figure 7, we display the relative sizes over the months of gathered data, measured in numbers of developers, of the major cluster (the largest cluster), the isolated clusters (consisting of a single project with a single developer), and all other clusters. Note that the major cluster has grown over time but is slowing relative to the rest of the social network.

DISCUSSION OF THE ANALYSIS

The analysis provides support for the contention that the F/OSS community is a self-organizing system, and it also yielded an unexpected finding regarding the structure of the community. Several different types of analyses on the F/OSS data obey the power-law, which gives support to the hypothesis advanced by many qualitative researchers that the F/OSS community operates as a self-organizing system. When one examines the size of connected projects and developers, however, a second phenomenon emerges. It appears that there may be a dual nature to the structure of the F/OSS community, at least at this point in time. While the less well-connected clusters fit the power law, suggesting that part of the network is operating as a self-organizing system, there is a substantial percentage of the network (34.6% in March 2003) that is behaving differently, and that cluster does not fit the power-law pattern of the rest of the network data.

To our knowledge this type of phenomenon has not been reported for other self-organizing systems, which may be due to the fact that our data set is large enough for us to be able to do a cluster analysis on cluster size. There are many possible explanations for this dual nature to the network structure that we are observing in the network. The F/OSS community does not operate in a vacuum, and in fact it may operate in some respects as a “shadow” network, with its structure influenced by the structure of the outside networks some of the developers and projects belong to. For example, the commercial software industry can be expected to exert an influence on the structure of the F/OSS network by influencing the training and background of the developers, as well as influencing the reward and incentive structure for working on a particular type of project. Many F/OSS projects have developers that are full time employees of commercial software firms assigned to the projects, e.g., IBM is reported to have as many as 350 employees on various F/OSS projects including Apache, Linux, Jikes, and Samba, and has invested over $1 billion USD in Linux related development (Hochmuth, 2002).

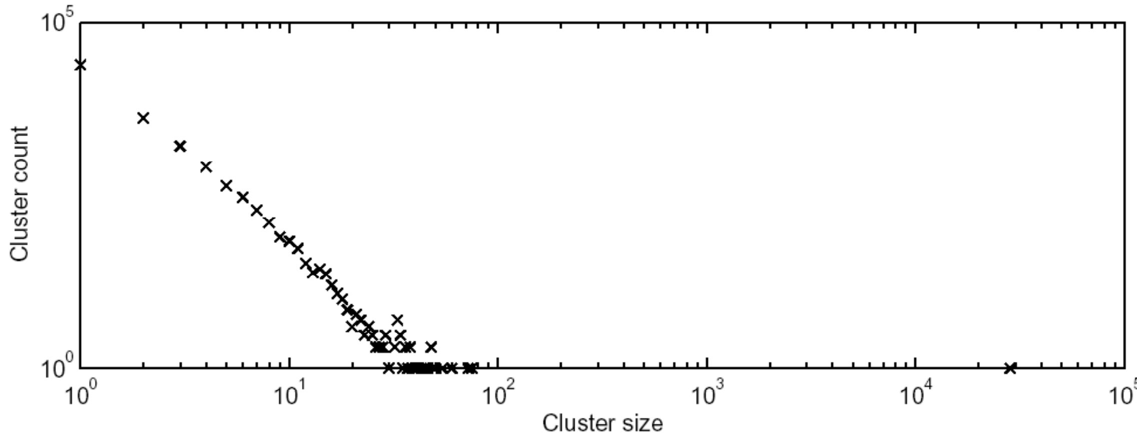

Figure 6: Cluster analysis on SourceForge developer data – one large cluster (far right) and remaining clusters with a power distribution of sizes

Several follow-on research questions suggest themselves to help better understand the meaning and significance of this dual structure. Do projects and developers in the large cluster differ in observable ways from projects and developers that fit the power-law? At what point in the development of the network does the super-cluster emerge and what percentage of the whole will it converge to? What relationship does this network structure have to threshold effects that have been observed in real-world network phenomenon such as the spread of computer viruses or vulnerability to denial-of-service network attacks? If there really is a dual nature to the F/OSS network, we need to recognize and understand it as such, as the network and behavioral dynamics observed in the super-cluster may differ significantly from that part of the network that is organizing itself as a power law distribution. While we think this phenomenon is likely to be

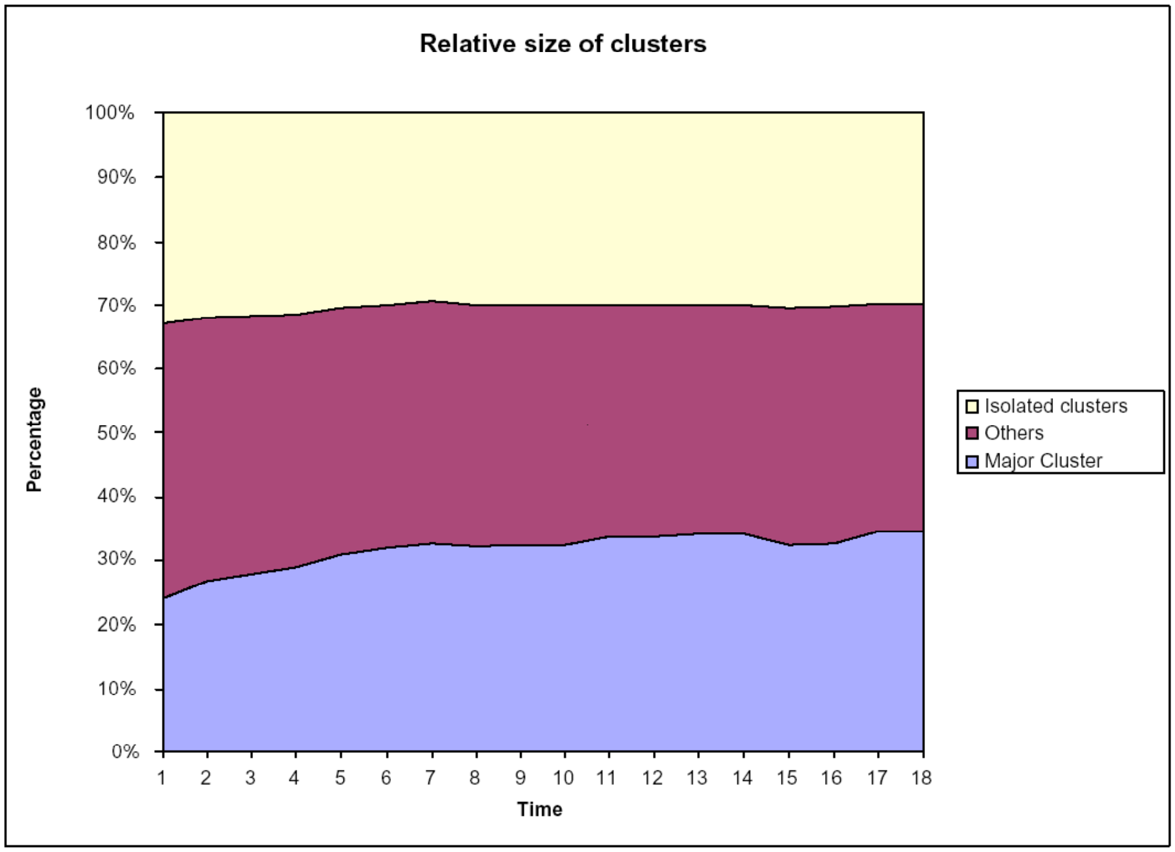

Figure 7: Relative cluster sizes. Major cluster comprises about 34.6% of all developers.

due to idiosyncratic properties of the F/OSS community and the fact that it is a shadow network and subject to strong external pressures, another possibility which must be examined is whether data in other self-organizing systems shows this dual nature when it becomes large enough, or at a certain point in its development.

AGENT-BASED MODELING OF F/OSS

Agent-based modeling is a technique for understanding the temporal dynamics and emergent properties of self-organizing system by simulating the behavior of individual components of the network over time (R. Axelrod, 1984; Epstein, 1996; Resnick, 1994; Schelling, 1978). We use empirically collected data to generate models of the F/OSS using social network theory (see Figure 8). We then use that model to provide specifications for an agent-based simulation in which we grow an artificial SourceForge over time (Epstein, 1996; Goldspink, 2000, 2002). We can then compare the outcome of the simulation to the data to assess and improve the model. In addition, using multiple iterations of the simulation given different starting conditions, we can discover invariant properties of the network/simulation that would not otherwise be observable.

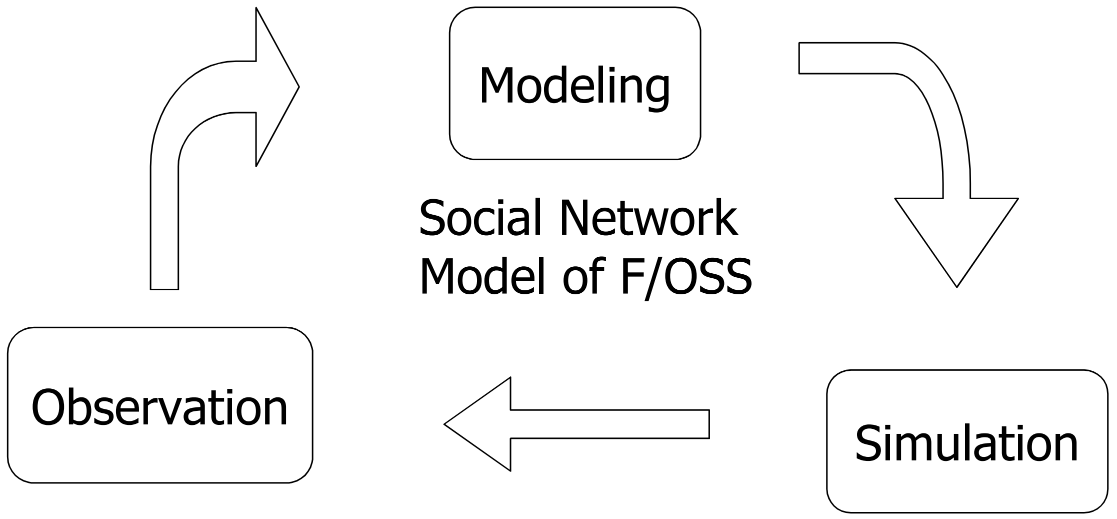

Figure 8: Coupled agent-based modeling and simulation help gain a better understanding of the F/OSS phenomenon.

Although the rules describing the local interactions of the components or agents of a self-organizing system (in our case, F/OSS projects and developers) may be few and simple, often unexpected and difficult to predict global properties emerge. Many investigators of such systems have found that they can only be understood through modeling, and specifically through agent-based simulation (also called iconological, individual-based, and structural modeling) (Eve, 1997; Harvey, 1997; Kiel, 1997; Smith, 1997).

An understanding of a complex system can be obtained by discovering the rules and mechanisms that control agent interactions, by discovering invariant global properties, and conditions that lead to stability, periodic behavior, or chaotic behavior. Prediction may not be possible because of sensitive dependence to initial conditions, dynamic coupling, and feedback. Rather, the goal of these simulations is to develop an understanding of how and why the elements of the system are able to produce emergent behavior (R. Axelrod, 1997a, 1997b; Harvey, 1997; Holland, 1998). Possible emergent behaviors in these complex systems might include adaptation, learning, memory, cooperation, and the persistence of self-sustaining temporal patterns.

To conduct an agent-based simulation researchers specify types of agents. In our simulations, we used only one type of agent for each simulation, F/OSS developers. Future research could, however, specify different types of developers and different types of projects (e.g., game projects, programming language projects, Internet projects, etc.). Next the researchers specify behavior rules for the agents. These rules can be either deterministic (e.g., "If X happens, Y type of agent will do Z") or probabilistic (e.g., "If X happens, Y type of agent has a 20% chance of doing X"). The model can also contain information about the environment the agents are interacting in.

In our simulation we model at every time period (1 day) whether the developers in our artificial SourceForge community will 1) start a new project, 2) join another existing project, or 3) quit a project. In addition, at every time period we model whether a new developer will join the community, either by starting a new project or joining an existing project. We use the data we collected at SourceForge to provide parameters for our simulation (e.g., the growth of the number of developers and projects at SourceForge over time, the ratio of new project creation between current developers and new developers, the evolution of projects as measured by number of developers, downloads, lines-of-code, bug reports, and posting on discussion lists, rates at which developers leave projects, project termination rates, etc.)

The simulations are built using the Java programming language and the agent-based modeling library Swarm (Minar, 1996; Swarm_Development_Group, 2000; Terna, 1998). The Swarm library enables discrete event, multi-agent simulations. In these simulations, agents are organized into "swarms". Several swarms can be nested into another swarm in a hierarchical manner. Thus, OSS developers can be grouped into a "project swarm", which in turn can be grouped into a "cluster swarm", which would be part of the entire "OSS development swarm". Both the Swarm library and the Java language are object-oriented, providing the object-oriented programming benefits of attribute and behavior encapsulation, information hiding, and inheritance (Epstein, 1996; Minar, 1996; Swarm_Development_Group, 2000; Terna, 1998). We are using modeling and simulation in concert with empirical data collection and analysis as shown in Figure 8.

We use the empirically collected data to generate models of F/OSS using social network theory. Next, we use that model as a specification for a Java/Swarm agent-based simulation in which we grow an artificial SourceForge. Using multiple iterations of the simulation given different starting conditions (e.g., different random seeds), we discover invariant properties of the simulation.

These properties, along with other simulation output are compared against the data we have on the “real-world’ SourceForge. Agreement between the real and artificial suggests that the model may be correct, although it does not prove it. Disagreement between the real and artificial suggests that the model (or simulation) may be flawed or incomplete and provides clues on where to look in the empirical data for hints on how to correct or refine the model. This process of coupling model and simulation to real data has provided us with addition insights into the F/OSS community. We conducted our agent-based simulations of SourceForge in an iterative manner, starting with the simplest model, checking its fit against the data, and improving the model. Using this method we obtained the following results:

1) Preferential Attachment Improves the Model. Our first simulation used random attachment of developers to projects, which fails to replicate the power law observed in the real data. Adding preferential attachment does improve the fit of the model. By experimenting with different simulation implementations we discovered that preferential attachment can be implemented independently for a) developers choosing a project yielding a power distribution on project sizes, and b) developers preferentially choosing to join an additional project, yielding a power distribution of developer index size. Figures 9 displays a comparison between: 1) the actual values of SourceForge’s average degree and diameter over time, and 2) the average degree and diameter of the network created in our simulation under the assumption of random joining (attachment) of

developers to projects. (In our analysis we identify networks that grow with random attachment as Erdos-Renyi graphs or ER for short). Likewise, Figure 10 displays a comparison between: 1) the actual values of SourceForge's average degree and diameter over time, and 2) the average degree and diameter of the network created in our simulation under the assumption of preferential joining (attachment) of developers to projects based on project size (or degree in our network representation). (In our analysis we identify networks that grow with preferential attachment as Barabasi-Albert graphs or BA for short). By comparing Figures 9 and 10, we see a much better fit for these network parameters (average degree and average diameter) in the simulations that assume preferential attachment.

2) Simply adding preferential attachment did not provide a model that could properly model the "young-upstart" phenomenon, where a new project or new developer quickly passes older projects or developers. Barabasi (A.-L. Barabasi, 2002) found that to model links to Internet sites a "fitness factor" was necessary. Fitness modifies the probability of a new edge, meaning that some websites are born more fit than others and are more likely to attract new links. Adding a fitness factor to our model did improve the fit of the model, although there were discrepancies.

3) In our search to understand these discrepancies we made the discovery that the best fitness factor for the SourceForge data is not static, but rather dynamic as a function of project age and life-cycle. In other words, these data are better modeled with a fitness factor that changes over time. For example, a new project may start with an attractive fitness factor, but its fitness levels over time, and then decreases.

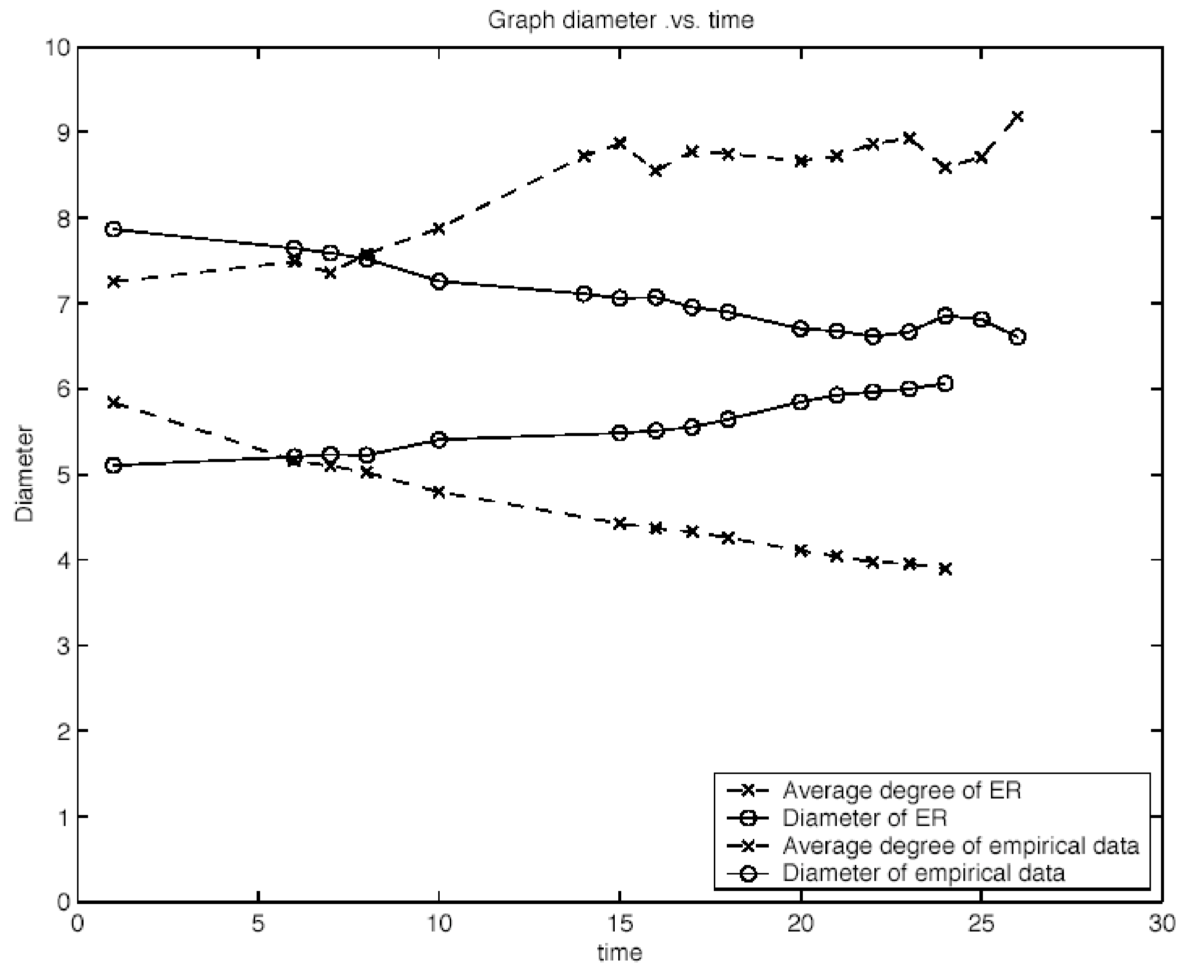

Figure 9: Comparison between: 1) the empirical values of SourceForge's average degree and diameter over time, and 2) the average degree and diameter of the network created in our simulation under the assumption of random attachment (ER) of developers to projects

CONCLUSION

We investigated the potential for learning more about the F/OSS community by 1) using SourceForge to learn more about developer and project evolution over time, 2) modeling the F/OSS community using social network theory, 3) discovering that the frequency of developer index, project sizes, and cluster sizes (excluding the one large component) all have a power-law

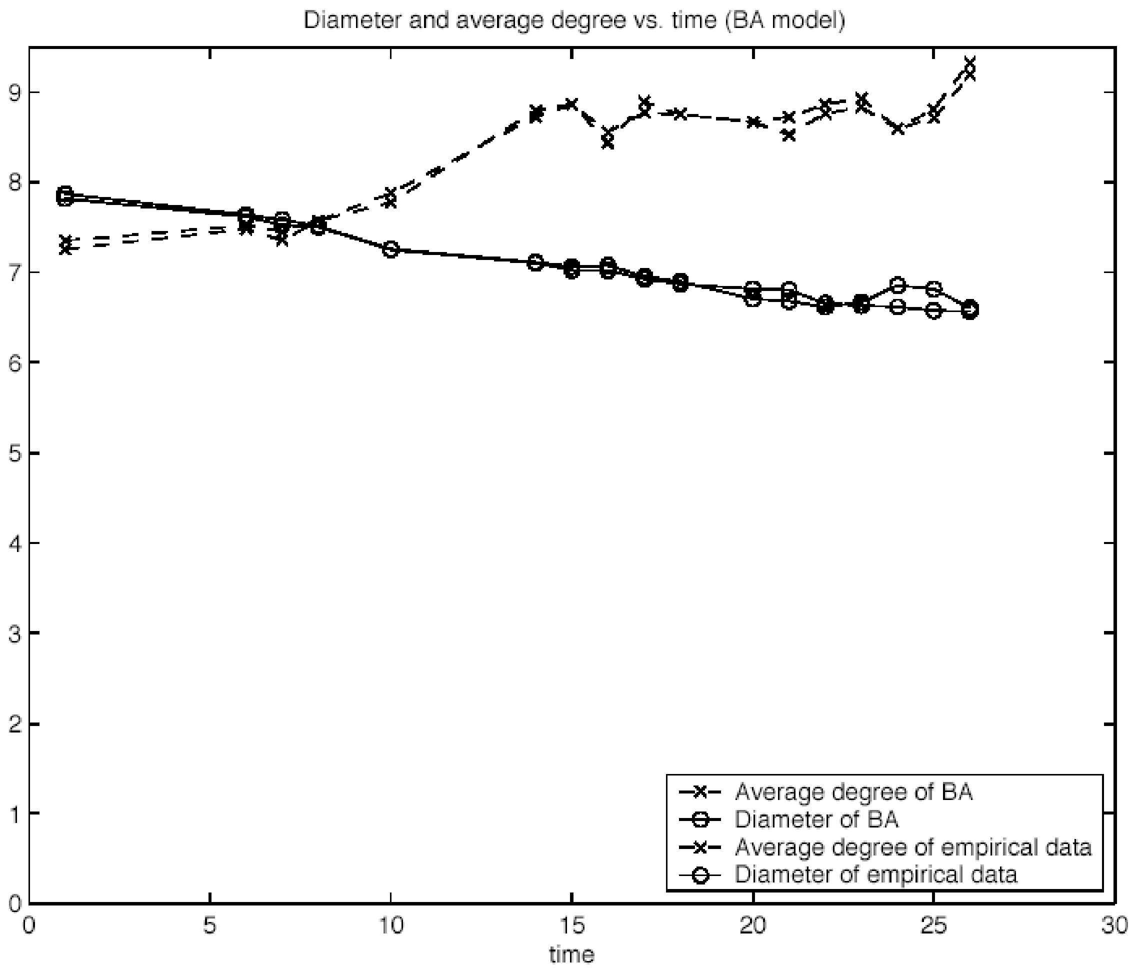

Figure 10: Comparison between: 1) the empirical values of SourceForge's average degree and diameter over time, and 2) the average degree and diameter of the network created in our simulation under the assumption of preferential attachment (BA) of developers to projects

distributions, 4) simulating the evolution of "artificial" F/OSS communities using agent-based modeling, and 5) iterating a coupled observation-modeling-simulation cycle to discover properties of the F/OSS community. Using the projects at SourceForge, the F/OSS community appears to be a highly fragmented social network, with the largest connected component comprising approximately 35% of all developers. Also, supporting this observation is the observed power law distribution in the SourceForce community which has as the most frequent value, projects with only one developer and developers on only one project with a relatively large intersection of the two. Future work includes: 1) studying Savannah and other F/OSS sites (under way), 2) compute and analyze additional social network metrics such as clustering coefficients, network diameter, and average index values, and 3) continue to refine the model and simulation using the process displayed in Figure 8.

ACKNOWLEDGMENTS

This research was partially supported by the US National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829. We acknowledge the assistance of Patrick McGovern, Director of SourceForge.net, and the contributions of University of Notre Dame students Yongqin Gao, Chris Hoffman, Carlos Siu and Nadir Kiyanclar with their assistance with data collection, data analysis, and programming on this project. Finally, we wish to thank the two anonymous reviewers of this chapter for their most helpful suggestions.

REFERENCES

Albert, R., Jeong, H., Barabasi, A. L. (1999). Diameter of the World Wide Web. *Nature, 401*, 130-131.

Axelrod, R. (1984). *The Evolution of Cooperation*. New York: Basic Books.

Axelrod, R. (1997a). Advancing the Art of Simulation in the Social Sciences. *Complexity, 3*(2), 16-22.

Axelrod, R. (1997b). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton: Princeton University Press.

Axelrod, R., M. Cohen. (1999). *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. New York: The Free Press.

Axtell, R. L. (2001). Zipf Distribution of U.S. Firm Sizes. *Science, 293*(5536), 1818-1820.

Barabasi, A.-L. (2002). *Linked: The New Science of Networks*. Boston: Perseus.

Barabasi, A. L., Albert, R. (1999). Emergence of Scaling in Random Networks. *Science, 286*, 509-512.

Barabasi, A. L., Albert, R., Jeong, H. (2000). Scale-free Characteristics of Random Networks: The Topology of the World Wide Web. *Physica A*, 69-77.

Barabasi, A. L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., Viscek, T. (2001). *Evolution of the Social Network of Scientific Collaborations*. Retrieved April 10, 2001, xxx.lanl.gov/arXiv:cond-mat/0104162v1, from xxx.lanl.gov/arXiv:cond-mat/0104162v1

Corfmen, T., Leiserson, C., Rivest, R., Stein, C. (2001). *Introduction to Algorithms* (2nd ed.): The MIT Press.

Cowan, G., Pines, D., and Melzer, D. (Ed.). (1999). *Complexity*: Westview.

Epstein, J. M., R. Axtell. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Cambridge, MA: The MIT Press.

Eve, R., S. Horsfall, M. Lee (Ed.). (1997). *Chaos, Complexity and Sociology*. Thousand Oaks: Sage Publications.

Faloutsos, M., Faloutsos, P., Faloutsos, C. (1999). *On Power-Law Relationships of the Internet Topology*. Paper presented at the SIGCOMM'99, Cambridge, MA.

Ghosh, R., and Prakask, V. V. (2000). Orbiten Free Software Survey. *First Monday, 5*(7).

Goldspink, C. (2000). Modelling social systems as complex: Towards a social simulation meta-model. *Journal of Artificial Societies and Social Simulation, 3*(2).

Goldspink, C. (2002). Methodological Implications of Complex Systems Approaches to Sociality: Simulation as a Foundation for Knowledge. *Journal of Artificial Societies and Social Simulation, 5*(1), 1-19.

Granovetter, M. (1973). The Strength of Weak ties. *American Journal of Sociology, 78*, 1360-1380.

Guare, J. (1990). *Six Degrees of Separation*. New York: Vintage Books.

Harvey, D., M. Reed. (1997). Social Science as the Study of Complex Systems. In L. D. Kiel, E. Elliot (Ed.), *Chaos theory in the Social Sciences: Foundations and Applications* (pp. 295-323). An Arbor: The University of Michigan Press.

Hochmuth, P. (2002, December 12). IBM's open source advocate. *Network World*, http://www.nwfusion.com/power/2002/frye.html.

Holland, J. (1998). *Emergence: From Chaos to Order*. Reading, MA: Addison-Wesley Publishing Company.

Huberman, B. A., Adamic, L. A. (1999). Growth Dynamics of the World Wide Web. *Nature, 401*, 131.

Jin, E. M., Girvan, M., Newman, M. E. J. (2001). *The Structure of Growing Social Networks*.Unpublished manuscript, Santa Fe.

Johnson, S. (2001). *Emergence*. New York: Scribner.

Jorgensen, S. E., Mejer, H., Nielsen, S. N. (1998). Ecosystem as Self-organizing Critical Systems. *Ecological Modeling*, 261-268.

Kelly, K. (1994). *Out of Conrol*. Reading: Addison-Wesley.

Kiel, L. D., E. Elliot. (1997). *Chaos Theory in the Social Sciences: Foundations and Applications*. Ann Arbor: The University of Michigan Press.

Krishnamurthy, S. (2002). An Empirical Examination of 100 Mature Open Source Projects. *First Monday, 7*(6).

Kuwabara, K. (2000). Linux: A Bazaar at the Edge of Chaos. *First Monday, 5*(3), 1-68.

Madey, G., Freeh, V., and Tynan, R. (2002a). *Agent-Based Modeling of Open Source using Swarm*. Paper presented at the Americas Conference on Information Systems (AMCIS2002), Dallas, TX.

Madey, G., Freeh, V., and Tynan, R. (2002b). *The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory*. Paper presented at the Americas Conference on Information Systems (AMCIS2002), Dallas, TX.

Madey, G., Freeh, V., and Tynan, R. (2002c). *Understanding OSS as a Self-Organizing Process*. Paper presented at the The 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002), Orlando, FL.

Minar, N., R. Burkhart, C. Langton, M. Askenzi. (1996, June 21, 1996). *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*, from http://www.santafe.edu/sfi/publications/96wplist.html

Mockus, A., R.T. Fielding, J. Herbsleb. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309-346.

Newman, M. E. J. (2001). *Clustering and Preferential Attachment in Growing Networks*.Unpublished manuscript, Santa Fe.

Pumain, D., Moriconi-Ebrard, F. (1997). City Size Distributions and Metropolisation. *GeoJournal, 43*(4), 307-314.

Resnick, M. (1994). *Turtles, Termites, and Traffic Jams*. Cambridge, MA: The MIT Press.

Schelling, T. (1978). *Micromotives and Macrobehavior*. New York: W. W. Norton.

Schroeder, M. R. (1991). *Fractals, Chaos, Power Laws*. New York: W. H. Freeman and Company.

Smith, T. (1997). Nonlinear Dynamics and the Micro-Macro Bridge. In R. Eve, S. Horsfall, M. Lee (Ed.), *Chaos, Complexity and Sociology* (pp. 52-78). Thousand Oaks: Sage Publications.

SourceForge. (2003). *http://sourceforge.net/*. Retrieved March 2003

Swarm_Development_Group. (2000). *Brief Overview of Swarm*, from http://www.santafe.edu/projects/swarm/swarmdocs/set/book149.html

Terna, P. (1998). Simulation Tools for Social Scientists: Building Agent Based Models with SWARM. *Journal of Artificial Societies and Social Simulation, 1*(2), 1-12.

Tjaden, B. (1996). *The Kevin Bacon Game*. Retrieved July, 2001, from http://www.cs.virginia.edu/oracle/

Wasserman, S., K. Faust. (1994). *Social Network Analysis: Methods and Applications*. Cambridge, UK: Cambridge University Press.

Watts, D. (1999). *Small Worlds*. Princeton: Princeton University Press.

Watts, D., Strogatz, S. H. (1998). Collective Dynamics of Small-World Networks. *Nature, 393*, 440-442.