Modeling and Simulation of the Open Source Software Community

Yongqin Gao, Greg Madey
Department of Computer Science and Engineering
University of Notre Dame
ygao1, gmadey@nd.edu

Vince Freeh
Department of Computer Science
North Carolina State University
vin@cse.ncsu.edu

Abstract

The Open Source Software (OSS) development movement is a classic example of a social network; it is also a prototype of a complex evolving network. In this research, we used simulation to help us with understanding the OSS development movement. First we generate a model to depict the evolution of the SourceForge community, one of the biggest OSS communities, as a network. And then we simulate the evolution of the network using Java/Swarm and calibrate the simulation with the empirical data we collected from SourceForge. In simulation, we used several different models to find the best fit model for the empirical data in an iterative fashion. These models include a random network, a scale-free network and a scale-free network with constant fitness. Also we proposed and verified the scale-free network with dynamic fitness model at the end of the simulation iterations.

1 Introduction

Open source[1] software, usually created by volunteer programmers dispersed worldwide, now competes with that developed by commercial software firms. This is due in part because closed-source proprietary software has associated risks to users. These risks may be summarized as systems taking too long to develop, costing too much and not working very well when delivered. Open source software offers an opportunity to surmount those disadvantage of proprietary software. SourceForge[2] is one of the most popular OSS hosting web sites, which offers features like bug tracking, project management, forum service, mailing list distribution, CVS and more. In October 2004 it had almost 950,000 registered users and almost 90,000 registered projects. It is a good sample of the OSS community to study the underlying mechanisms of OSS communities.

The collaboration relationships between the developers in the OSS community are the focus of our study. These relationships can be studied if we model them as a complex network. Scientists and mathematicians have been studying such networks for some time. Different from the traditional networks computer scientists study, randomness is dominant in this kind of network, since there is no centralized control of the construction and evolution of the network, every node in the network will determine what nodes to connect to itself based on the local knowledge it has. In this paper, we will use different network models for the complex networks, including "random networks" [10], "Small world networks" [17], and "Scale-free networks" [8]. Finally, we will propose a modified scale-free network

[1]"Open source" is a certification mark owned by the Open Source Initiative(http://www.opensource.org)

[2]http://sourceforge.net

model to fit the OSS community.

The remainder of this paper is structured as follows. Section 2 reviews related work. In Section 3, we discuss the model we generated to depict the SourceForge community. The result and analysis of the simulation we build for the SourceForge community and the verification and validation of the simulation are provided in Section 4. Finally, we conclude in Section 5 with directions for future work.

2 Related Works

Recently, collaboration networks have gained more and more attention. Many of these networks can be precisely described by bipartite graphs [13] and there are many papers about these collaboration networks [17, 19, 14, 15, 16, 7]. In this research, we also study The SourceForge community as a collaboration network. But in addition to the statistical analysis used in these papers[3], we use agent-based simulation to help us understand the network better in this paper.

Previous research involving the simulation of complex networks is based on snapshots [8, 17, 13], which focused on the topology of the network at a given point in time. Since we collected data over a two year period, we were able to look into the evolution of the network. This helped us understand the evolution of the network topology, the development patterns of any single object in the network and the impact of the interaction among objects on the evolution of the overall network system. Preliminary reports on this research have been published in proceedings of NAACSOS 2003 [1, 2]

3 Modeling

Our model is based on a bipartite graph (the collaboration network of SourceForge) so we would be able to describe the attachment and the detachment between project and developer more straightforward.

In our simulation, there were two kinds of entities – developer and project. Developers, which are the active agents in the real SourceForge network, will also be the active entities (agents) in our model. Our simulation is a time-step simulation, which means that every agent in the simulation acted at every time step. We define the time step in our model as one day in the real world.

At every time step, there was a given number of new developers, $N(ND)$, added into the simulation. In our model, the number of new developers at every time step is a random number generated by a uniform distribution. Then every existing developer (existing and new developers) in the simulation is triggered to act one by one. The simulation of a new developer and an existing developer are different, which we discuss separately.

- *New developer.* Every new developer is triggered to act right after it was generated. There are two possible actions for a new developer – creating or joining. Creating meant that the developer would create a new project and thus add a link from this developer to a new project. Joining meant that the developer would select an existing project according to some rules, and thus a link from this developer to the project was added. $P(CN)$ and $P(JN)$ are the two parameters that controlled the probability of creating and joining. The one exception was that the very first developer in the simulation had to create a new project because there were no existing projects in the system yet.

- *Existing developer.* Every existing developer was triggered to act at every time step. There were four possible actions for an existing developer – creating a project, joining a project, abandoning a project and no action. Creating and joining were similar to the definitions for the new developer. Abandoning meant that the developer would abandon a randomly selected project in which he had participated. No action

[3]Related study can be found at [1]

meant that the developer did nothing, which is normally the most frequent action a developer will take. $P(CO)$, $P(JO)$, $P(AO)$ and $P(IO)$ are the four parameters to control the probability of creating, joining, abandoning and no action. Details on how a developer chooses a certain action are different for the different models we simulated.

The different simulation models may have different details in their definitions. These differences focused on the rules a developer used to choose an action and the rules a developer used to decide what project to join. Also the related parameters may also be different for different simulation iterations. We will explain these different details and related parameters when we discuss each simulation model.

4 Simulation Results

4.1 Experiment Setup

We ran our simulations on a computer cluster, which includes three simulation servers, three database servers and one gateway machine. These machines are connected by a 100M(bps) switched LAN (Local Area Network). The gateway is the only connection between the cluster LAN and the campus network.

The simulation servers, which are dedicated to simulations, are all dual PIII 650MHz with 1G of memory. The database servers, which are dedicated to database management, are also dual PIII 650MHz with 1G of memory and 160G Storage.

The operating systems of the cluster is as follows: The simulation servers are Linux with 2.4.18 kernel. And the database servers are Windows 2000 Server with Oracle 9i. The simulation toolkit we used was Swarm and the Java programming language.

The metrics we will use in the simulation iterations include degree distribution, diameter, clustering coefficient and cluster distribution [6].

4.2 Simulation Models

We ran the simulations in an iterative fashion. This means we started from a basic model, running a simulation based on that model, and then verified and validated it using the empirical data. After adjusting the model and related simulation parameters, we ran the simulations again. We repeated the iterations until we got a good fit with the empirical data. Now we will discuss the details of our simulation models one by one.

4.2.1 Model one: random network (adapted ER model)

We started with the ER model since it is a well-known complex network model and is the foundation of the others. This model can be studied using nonlinear dynamics and displays some extraordinary properties caused by phase transitions. One of them is the existence of well-defined boundaries that separate order from disorder [5].

The model we use is growing, which is different from the traditional ER model. So we call it an adapted ER model. Randomness is the dominant factor in the procedures of the adapted ER model. The rules a developer used to choose an action and the rules a developer used to decide what project to join are all based on randomness.

For developers, the randomness is the uncertainty in action selection decision, which means that the developer randomly chooses the actions according to given probabilities. Thus, the randomness for a developer is implemented by random action selection based on fixed probabilities given by six simulation parameters. For the project, the randomness is in the project selection. When a developer decided to join a project, he had no preference for one project over any other and every project had the same probability of being chosen.

According to the model description in the last section, we have six simulation parameters, which are used to define the probabilities of each candidate action, $P(CN)$, $P(JN)$, $P(CO)$, $P(JO)$, $P(AO)$ and $P(IO)$. For a new developer, there are no other possibilities except creating and joining, thus $P(CN) + P(JN) = 1$. In order to match the simulation with the empirical data, we use the statistical average percentages of these two actions among new developers from the empirical data as initial values for $P(CN)$ and $P(JN)$. The value of $P(CN)$ is set at 0.67331, and the value of $P(JN)$ is set at 0.32668. For a existing developer, there are four possible actions, to create, join, quit or idle, and their relation is restricted by the equation $P(CO) + P(JO) + P(AO) + P(IO) = 1$. We also used the statistical average percentages of these action selections as the initial values. The results are as follows: $P(CO) = 0.018028$, $P(JO) = 0.005974$, $P(AO) = 0.0076881$ and $P(IO) = 0.96831$.

After running the simulation for the ER model, we validated it by comparing the simulation output with existing theoretical results and verified it by comparing the simulation output with the empirical data. The first property we investigated was the clustering coefficient. The clustering coefficient (CC) of the simulated ER model is shown in Figure 1. The clustering coefficient of ER model is below 0.24 (presented as 'o' in the figure), which is much smaller than that of the empirical data of SourceForge developer network (where $CC \sim 0.75$, which are presented as 'x' in the figure). Also it diminished quickly as the overall system size increased. The relation between clustering coefficient CC and system size N (between 21,200 and 35,320) is approximately

$$CC \sim N^{-0.14} \tag{1}$$

This observation is identical to the mathematical result for the clustering coefficient of the ER model [18], which was demonstrated to be $CC \sim N^{-\beta}$, β is a constant smaller than 1.

Average degree and diameter are two other attributes we investigated in the ER model. The average degree and the diameter of the simulated ER model are shown in Figure 2. The average degree of the simulated data is decreasing while the diameter of the simulated data is increasing ('x' in Figure 2). In theoretical ER models, the average degree should

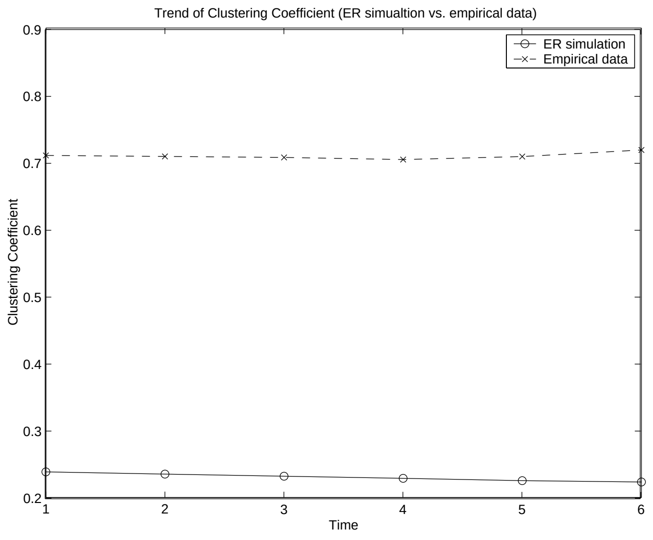

Figure 1: Clustering coefficient in ER: unit for X coordinate is month, and 1 represent July 2002

decrease while the size of the network increases and the mathematical expression for the diameter of an ER model [18] is:

$$D \sim \frac{log(N)}{log(<k>)}, \tag{2}$$

so the average degree and diameter of simulated data fit the mathematical results. This is different from the observations we get in the empirical data that the average degree should increase and the diameter of the network should decrease as the network grows based on SourceForge developer collaboration network data ('+' in Figure 2).

Next we examined the cluster distribution. The cluster distribution of the simulated ER model is shown in Figure 3. The upper figure is the cluster distribution with linear coordinates. There is one big cluster present, which is expected to appear in ER models at some time during evolution [11]. The lower figure is the cluster distribution with log-log coordinates. The data points do not fit the linear regressions in both the empirical data and the simulated data (R^2 for empirical data is 0.5556 and R^2 for simulated data is 0.4445) due to the major clus-

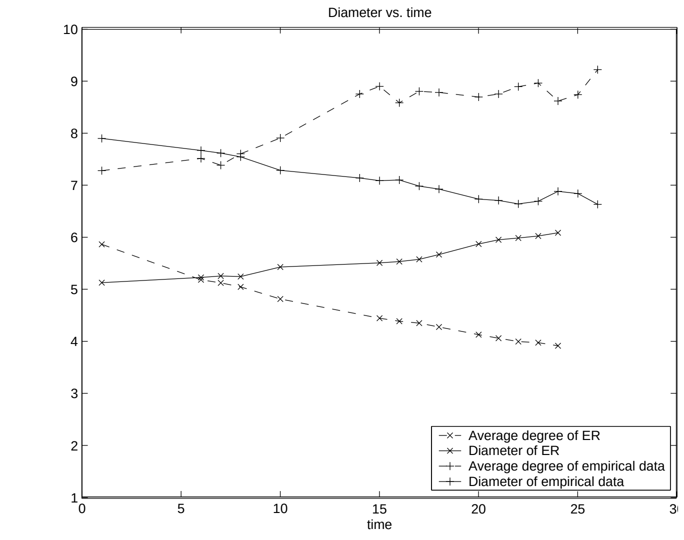

Figure 2: Average degree and diameter in ER simulation and the empirical data: unit of X is month, 0 represents December 2000 and simulated time 0

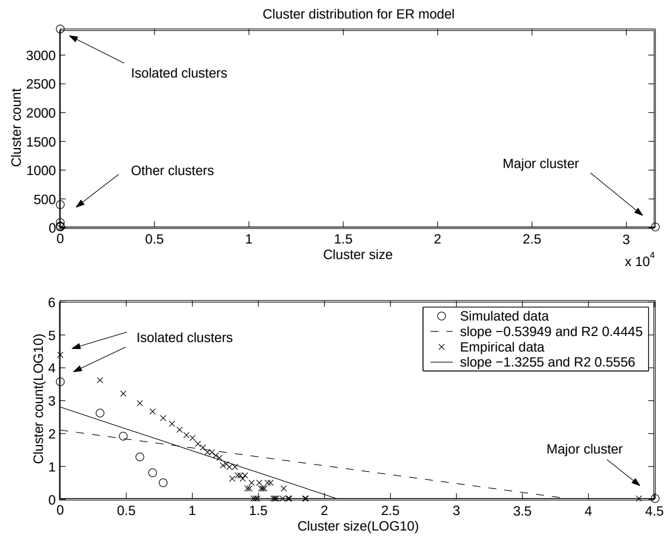

Figure 3: Cluster distribution in ER simulation and the empirical data

ter. Without considering the major cluster, the linear regression of cluster distribution of ER model has a similar slope to that of the empirical data and the R^2 values are quite good (R^2 for empirical data is 0.9598 and R^2 for simulated data is 0.9906). But the actual data points are different. The following simulated models all have the similar slopes of linear regressions in cluster distribution with more similar data points to the empirical data.

Finally, we investigated the degree distribution of the ER model. The simulation results of the ER model are shown in Figure 4. The upper two figures are developer distribution and project distribution in normal coordinates. There is no power law in the distribution. The distributions look more like the mathematically predicted Poisson distribution. Also, we compared them with the distributions of empirical data in log log coordinates, which are shown in lower figures. For empirical data, the developer and project distributions fit a straight line well, characterizing the power law distribution. In detail, we found a linear regression with a slope of -3.5311 and a R^2 of 0.9533 for the developer de-

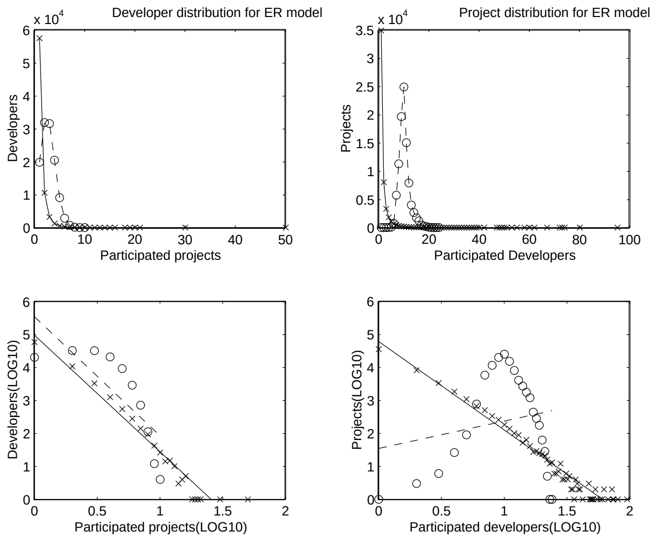

Figure 4: Developer and project distributions in ER simulation and the empirical data: ‘x’ represents empirical data and ‘o’ represents simulation data of ER model

gree distribution; we also found a linear regression with a slope of -2.6781 and a R^2 of 0.9633 for the project degree distribution. The ER model has distributions that are significantly different from that of our SourceForge collaboration network. In detail, we found the developer degree distribution has a R^2 as 0.619 and the project degree distribution has a R^2 as 0.0407. Also, we can see that in project distribution, the maximum number of developers in a single project is only 27, which is much less than the result of empirical data, which are 94. This is because the projects shared the same probability of being chosen, and no project had a preferential advantage to obtain significantly more developers. Until now, we validated the simulation of ER model with mathematical results, and we showed by simulation that the ER model is a poor model for the SourceForge developer collaboration network.

4.2.2 Model two: scale-free network (BA model)

The scale-free network model is the second model in our simulations. There are two major improvements made in the scale-free network: growth and preferential attachment. The growing property is already included in our adapted ER model, so we just need to add preferential attachment to our simulation.

In the model procedure definition, preference will influence both the rules that a developer used to choose an action, which is called developer preference later, and the rules that a developer used to choose what project to join, which is called project preference later. For project preference, the probability that a project would be selected was proportional to its degree. The probability of a project i being selected is defined as

$$P(i) = \frac{k_i}{\sum_{j \in AllProjects} k_j} \tag{3}$$

For developer preference, the more projects in which the developer participated, the higher the probability he would take an action (create, join or abandon) as opposed to no action. So in our simulation, the degree the developer already had was also one of the factors that determined the developer's action probabilities.

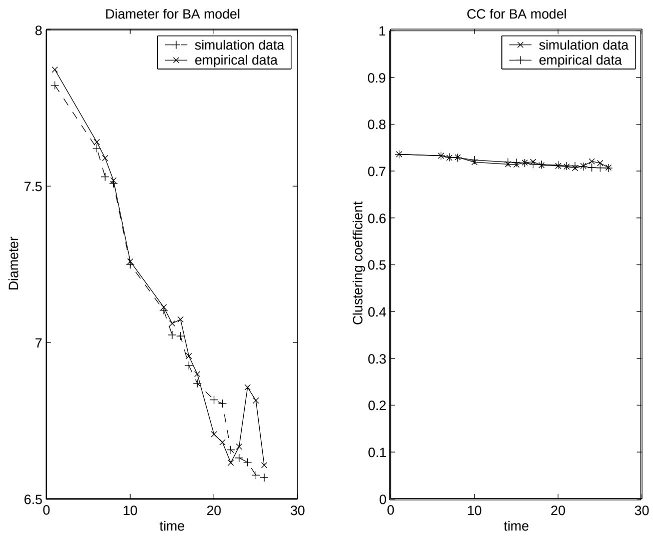

Figure 5: Diameter and Clustering coefficient in BA simulation and the empirical data: unit of X coordinate is month and 0 represent December 2000

Next, we explain the parameters used in the remaining models. These parameters include the daily new developer rate $N(ND)$ and the six action probabilities we discussed in section 3. The initial values of these parameters are chosen by the statistical average of the empirical data, which are the some parameter values used in the adapted ER model.

First, we investigated the diameter and the clustering coefficient of simulation data based on the BA model. The results are shown in Figure 5. The diameter is around 7 and decreasing. The clustering coefficient is around 0.7 and decreasing. They both match the empirical data well.

Then we investigate the degree distributions of this simulation iteration – the BA model. The developer and project distributions of our simulation for the BA model are shown in Figure 6. In the figure, the upper two figures are the developer distribution and project distribution in normal coordinates and

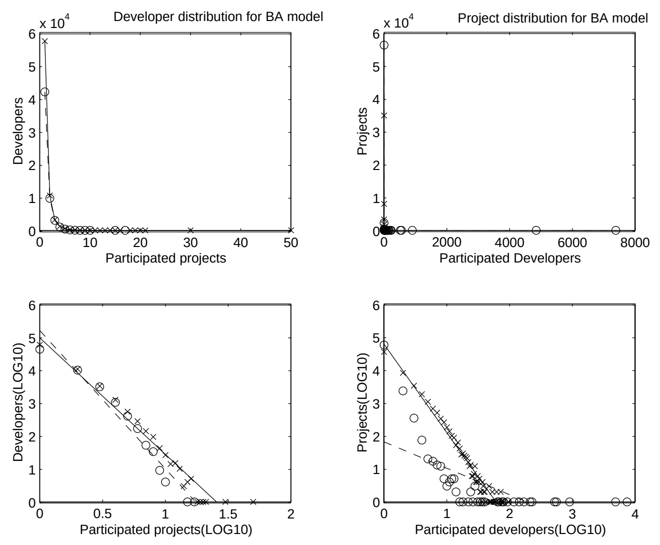

Figure 6: Developer and project distributions in BA simulation and the empirical data: 'o' represents simulation data of BA model and 'x' represent empirical data

the lower two figures are their distributions in log-log coordinates. In the lower figures, the "x" points are the distribution of empirical data and the solid line is its linear fit; the "o" points are the distribution of simulated data and the dash line is its linear fit. We observe that for the developer distribution, simulation data fits the empirical data quite well in both data points and the linear regression (the function of the linear regression for simulated data is $y = -4.1803x + 5.1833$ and R^2 is 0.96); for the project distribution, the data points and the linear regression does not fit well (the function of linear regression for simulated data is $y = -1.2454x + 2.5823$ and R^2 is 0.442). The discrepancy between the linear regressions of the simulation data and empirical data occurs because of the tails of the distributions; especially the largest project. We can find that the largest project in simulated data includes almost 8000 developers while the largest project in empirical data just includes about 200 developers.

Also there exists the phenomenon of young nodes possibly out-running old ones (for example, Google search engine overran the Yahoo! search engine in just a few years) in the real network. But BA model is not capable of reproducing this phenomenon.

So this model can reproduce most of the topological statistics we observed in empirical data, including the power law in both developer distribution and project distribution. But the project degree distribution does not 'fit' the empirical data and it can not reproduce the "young upcomer" phenomenon.

4.2.3 Model three: scale-free network with constant fitness

The scale-free network with constant fitness is the third model in our iterations. In this model we introduced a new attribute – fitness. Fitness is an appended attribute proposed by Barab$\acute{a}$si for the Scale-free network model to depict the "young upcomer" phenomenon. Considering the above deficiency, we studied the BA model with constant fitness as our third model. We defined the fitness as a constant parameter η_i for every project. Thus every time a new project (say, project j) is created, a fitness η_j was added to the system, where η_j is chosen from a distribution $\rho(\bullet)$. The probability of participating in a project i is proportional to the degree and the fitness of the project i,

$$P_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{4}$$

Theoretically the fitness distribution $\rho(\bullet)$ may be any distribution (e.g., normal distribution or exponential distribution). In our models (BA with constant fitness and BA with dynamic fitness), we used exponential distribution as the fitness distribution. The function of the distribution is

$$P(\eta) = \lambda e^{-\lambda\eta} \tag{5}$$

where λ is equal to 2.

The degree distribution of this model is mathematically proved to be

$$P(k) \sim \frac{k^{-C-1}}{ln(k)} \tag{6}$$

which is a power law with logarithmic correction [18] (C is a constant parameter). We also demonstrated that this model has power law distributions for developers and projects in simulation and the results are shown in Figure 7. The upper figures are the comparison of developer and project degree distributions between empirical data and simulated data in linear coordinates. We also applied log-log transformation on the data points and the results are shown in the lower figures. The R^2 for developer degree distribution in simulated data is 0.9559 and the R^2 for project degree distribution in simulated data is 0.5261. So the simulated data fit the empirical data much better on the developer degree distribution than on the project degree distribution. One of the significant differences is still the size of the largest project, which is around 6000 in the simulated data while it is around 200 in the empirical data.

Then we investigate the diameter and clustering coefficient of the simulated data. We demonstrated that the actual values and evolutions of the diameter and the clustering coefficient of this model match those of the empirical data[4].

Thus, although the BA model with constant fitness not only matches most of the statistics of the empirical data but is also capable of explaining the "young upcomer" phenomenon, it still has some deficiency in reproducing the same result as the empirical data especially for the project distribution. Further iteration is needed.

4.2.4 Model four: scale-free network with dynamic fitness

In the previous iterations, we discovered several phenomena that scale-free networks with constant fitness were not able to explain, like the project degree distribution.

We propose a dynamic fitness factor in the scale-free network, which allows the fitness for every project to decay with respect to time. In our model, we made the fitness decay linearly by month, which

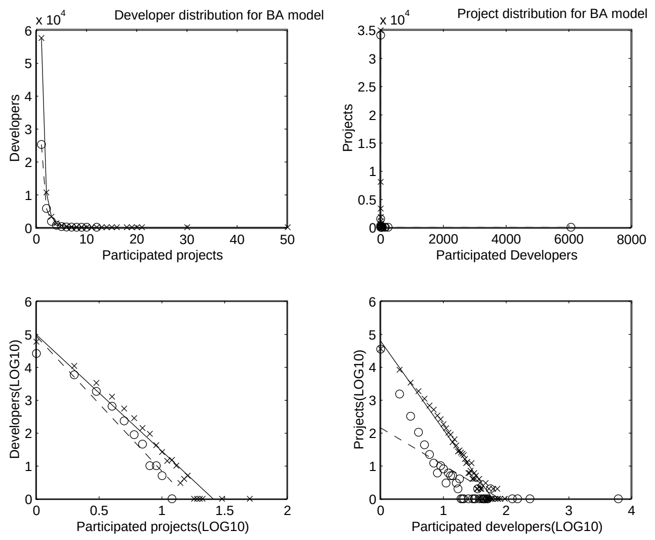

Figure 7: Degree distributions in BA simulation with constant fitness and the empirical data: for developer distribution, function of linear regression is $y = -4.1860x + 5.1876$ and R^2 is 0.9559; for project distribution, function of linear regression is $y = -0.8008x + 1.8169$ and R^2 is 0.5261

[4]The related results can be found at [4]

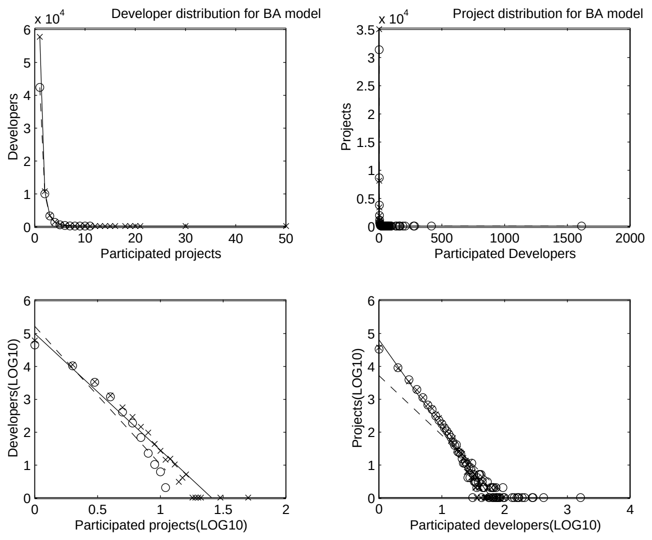

Figure 8: Degree distributions of BA simulation with dynamic fitness and the empirical data: ’o’ represents the data points from the simulated data and ’x’ represents the data points from the empirical data

is a non-negative integer. Every project possessed a random fitness when it was created by a developer.

The results of the degree distributions of BA model with dynamic fitness are shown in Figure 8. The upper figures are the comparison of developer and project degree distributions in linear coordinates. The lower figures are the comparison of developer and project degree distributions in log-log coordinates. The R^2 of developer degree distribution from the simulated data in lower figure is 0.959 and the R^2 of project degree distribution from the simulated data in lower figure is 0.7657. Also the largest project size of the simulated data is just 1500. We can further lower this value by tuning the fitness parameter.

Also we proved that this model could match the other statistics (diameter and clustering coefficient) of the empirical data[5].

After analyzing the simulation output of the BA model with dynamic fitness, we found that our model reproduced the network similar to and even better the BA model with constant fitness did, in relation to the attributes that we investigated (degree distribution, diameter and clustering coefficient).

[5]The related result can be found at [4]

5 Conclusion

5.1 Summary

Open Source Software (OSS) development is a classic example and prototype of collaborative social networks. Instead of the network snapshots used in the previous research, we used 24 months of network data on projects and developers that were collected monthly starting in January 2001 in this study. And in additional to the statistical analysis used in the previous research, We used simulation to verify and validate the model, which we proposed to depict the network evolution of the collaboration network, by comparing the simulation output with the properties we obtained by statistical study of the empirical data.

In simulation study of the model of SourceForge collaboration network, we proposed a new modified BA model - BA model with dynamic fitness. We iterated from ER model, BA model, BA model with constant fitness to BA model with dynamic fitness in the simulation. Through simulation, we verified all the properties in these models that are derived by mathematical methods and derived other properties that are mathematically difficult to derive. As a result, we get the best “fit” model and related simulation parameters, which can reproduce all the investigated properties in this paper.

5.2 Limitation and future work

Our work was based on the empirical data we obtained from SourceForge. However this empirical data is incomplete for several reasons: 1) the SourceForge site continues to grow and evolve, and 2) our data collection has a few monthly gaps. This ”incompleteness” may bias the reported results, so future analysis of additional data will be needed.

Although SourceForge is one of the biggest and most popular hosting sites in the OSS community, this is only one of many OSS sites. Results reported here may not generalize to all of those sites. So we may analyze other OSS hosting sites in the future.

In the simulation iterations, we found that the parameters we get from the statistics of the empirical data do not exactly fit the simulations. Thus we need adjust these parameters accordingly. Unfortunately, these parameters are interdependent and our "guess and try" approach mostly likely did not discover the best setting for those simulation parameters. An improved approach might couple mathematical optimization and our simulation approach.

References

[1] Y. Gao, V. Freeh and G. Madey, *Analysis and modeling of open source software community*, NAACSOS 2003, Pittsburgh.

[2] Y. Gao, V. Freeh and G. Madey, *Conceptual framework for agent-based modeling and simulation*, NAACSOS 2003, Pittsburgh.

[3] G. Madey, V. Freeh and R. Tynan, *Agent-based modeling of open source using swarm*, 8th Americas Conf. of Information Systems, 2002.

[4] Y. Gao, *Topology and Evolution of the Open Source Software Community*, Thesis, University of Notre Dame, 2003.

[5] Stuart Kauffman, *Origins of Order: Self-organization and Selection in Evolution*, Oxford University Press, Oxford, 1993.

[6] Z. Neda, E. Ravasz, A. Schubert and A.L. Barabási, *Evolution of the social network of scientific collaborations*, PhysicA, 311(590), 2002.

[7] R. Albert and A.L. Barabási, *Dynamics of complex systems: scaling laws for the period of boolean networks*, Physics Reviews.

[8] R. Albert and A.L. Barabási, *Emergence of scaling in random networks*, Science, 286:509-512, 1999.

[9] G. Drummond, *Open source software and documents: a literature and online resource review*, 1999.

[10] P. Erdös and A. Rényi, *On random graphs*, Publications mathematicae, 6:290-297, 1959.

[11] A. Bunde and S. Havlin, *Fractals and disordered systems*, 2nd Edition, Springer, Berlin, 1996.

[12] P. J. Kiviat, *Simulation, technology, and the decision process*, ACM Tran. on modeling and computer simulation, 1(2):89, 1991.

[13] D. J. Watts, M.E.J. Newman, *Random graph models of social networks*, Physics reviews, 64(026118), 2001.

[14] M.E.J. Newman, *Scientific collaboration networks: I. network construction and fundamental results*, Physics reviews, 64(016131), 2001.

[15] M.E.J. Newman, *Scientific collaboration networks: II. shortest paths, weighted networks, and centrality*, Physics reviews, 64(016131), 2001.

[16] M.E.J. Newman, *Clustering and preferential attachment in growing networks*, Physics reviews, 64(025102), 2001.

[17] D. J. Watts and S. H. Strogatz, *Collective dynamics of small-world networks*, Nature 393(440), 1998.

[18] R. Albert, and A. Barabási, *Statistical mechanics of complex networks*, Reviews of Modern Physics, 74(47), 2002.

[19] B. Bollobás, *Random graphs*, London:academic, 1985.