Modeling and simulation of Open Source Software community

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

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

Greg Madey
Department of Computer Science and Engineering
University of Notre Dame
gmadey@nd.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. By continuously collecting developers and projects information from SourceForge for two years, we have sufficiant data to infer the dynamic and the structural mechanisms that govern the evolution and topology of this complex system. We use this information in three steps. First, we analyse the empirical data we get from SourceForge to get statistics of the OSS developer social network. Thus we can get the evolution parameters about the network. Second, we generate a model to depict the evolution of this network. Finally we simulate the evolution of the OSS developer complex network using Java/Swarm and tune the simulation to match the real data. In simulation, we used several different model to find the best fit model for real data in an iterative fashion. These models include random network, scale-free network and scale-free network with constant fitness. Also we verified the model we proposed in the simulation iterations.

1 Introduction

Open source[1] software, usually created by volunteer programmers dispersed worldwide, now competes with that developed by commercial software fi rms. 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

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

to surmount those disadvantage of proprietary software. SourceForge[2] is one of the most famous OSS hosting web sites, which offer features like bug tracking, project management, forum service, mailing list distribution, CVS and more. It has more than eighty thousands developers and fi fty thousands projects right now. 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 focus of our study. These relationships can be studied if we look them as a network. Scientists and mathematicians have been dealing with these complex networks for some time. Different from the traditional network computer scientists studied, 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 by itself based on the local knowledge it has. In this paper, we will discuss these network models for the complex networks, including "random network theory"(proposed by Erd$\ddot{o}$s and R$\acute{e}$nyi), "Small world networks"(proposed by Watts and Strongatz) to "Scale-free networks"(proposed by Albert and Barab$\acute{a}$si).

The remainder of this thesis is structured as follows. Section II will discuss the model we generated to depict the SourceForge community. Section III will discuss the simulation we build for the SourceForge community and the verifi cation and validation of the simulation. Finally, we will give a conclusion of our work in section IV.

2 Modeling

Our collaboration network of SourceForge is different from others considering the existence of detachment, which means the edge is not permenant and can be removed some time. The traditional models are all based on permenant links and can not depict our collaboration network. So we will propose a new model for SF collaboration network. We knew that the colloaboration network is rooted in the bipartite graph with two kinds of entities – developer and project. And if we model the network evolution from bipartite graph, we are able to include the detachment in the model easily. So we proposed a model for the collaboration network based on the bipartite graph.

The model for simulation is like a procedure defi nition. In our simulation, there will be two kinds of entities – developer and project. developer as the active role in real SourceForge network will also be the active entity in our model. Our simulation is time based simulation, which means that every agents in the simulation is triggered to act every time step. We defi ne the time step in our model as one day in real world.

Every time step, there are given number of new developers, $P(ND)$, added into the simulation. In our model, the actual number of new developers every time step is random number generated by a normal distribution averaged at $P(ND)$. Then every existed developers ("old" and new developers) in the simulation

[2]http://sourceforge.net

will be triggered to act one by one. The procedure definitions for new developer and "old" developer are different, which we will discuss separately.

- *New developer.* Every new developer will be triggered to act right after it is generated. There are two possible actions for a new developer – creating or joining. Creating means that the developer will create a new project and thus add a link from this developer to the new project. Joining means that the developer will select an existed project according to preferential attachment, and thus a link from this developer to the project will be added. $P(CN)$ and $P(JN)$ are the two parameters to control the probability of action creating and action joining. The probability is fixed for new developer, which means no developer preferential attachment for new developer. Since new developers are all same with zero paticipated project. There is one exemption that the very first developer in the simulation have to create a new project because there is no existed project in the system yet.

- *"Old" developer.* Every "old" developer will be triggered to act at every time step. There are four possible actions for a "old" developer – creating, joining, abandoning and idling. Creating and joining is similar to the definitions in the new developer. Abandoning means that the developer will abandon a random selected project the developer paticipated. In real work, the abandoning should also have preference, but we just use random selection in our simulation. Idling means that the developer will do nothing, which is the major action a developer will take. $P(CO)$, $P(JO)$, $P(AO)$ and $P(IO)$ are the four parameters to control the probability of action creating, joining, abandoning and idling. The real probabilities a developer will choose a certain action is a function of the relative parameter and the degree of the developer. This function depicts the presence of the developer preference.

3 Simulation

3.1 Experience Environment

out simulations were tested in the NOM cluster, which includes three simulation servers, three database servers and one gateway machine. These machines are connected by a 100M LAN(Local Area Network). The gateway is the only connection between the cluster LAN and the campus network and it will take the responsibility of all the accesses from campus to the cluster. The simulation servers, which are designed to be dedicated to simulations, are all dual PIII 650 with 1G memory. The database servers, which are designed to be dedicated to database management, are also dual PIII 650 with 1G memory and 160G Harddisk. The software environment of the NOM 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 toolkits we used is Swarm and the programming language is Java.

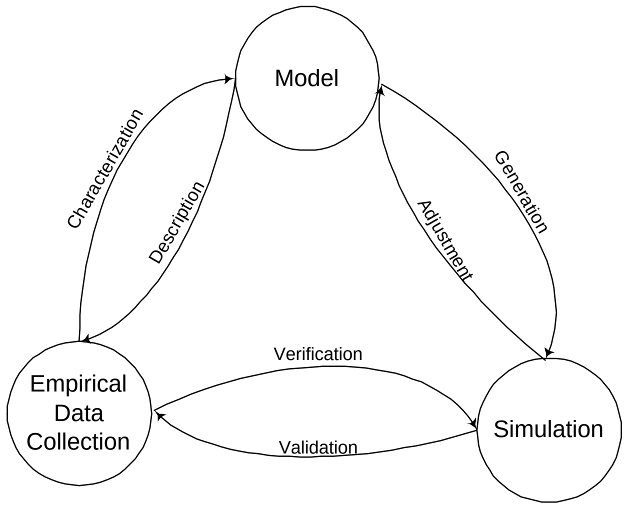

Figure 1: Conceptual framework for agent-based modeling and simulation

3.2 Framework for simulation study

We proposed a conceptual framework for agent-based modeling and simulation, as shown in Figure 1. We have three entities in the framework: empirical data collection, model and simulation. The model is the process description that is implemented in simulation, and by which we can reproduce the evolution of empirical data. The simulation is a method to verify and validate the model. We also have six relations in the framework, each represents one kind of process in the framework. Characterization is a process to abstract the characteristics of empirical data and to generate the rules and attributes in the model. Description is a process to manifest the underlying mechanisms of the evolution of empirical data. Generation is a process to build a simulation based on the given model. Adjustment is a process to modify the model according to the feedback from verifi cation process. Verifi cation is a process testing the simulation's behaviors by comparing the simulation output with the empirical data and the designed model behaviors. Validation is a process of interpolating or extrapolating the simulation output and comparing with the empirical data and its attributes not used to defi ne the model. Validation may add more rules or attributes into the model for prospective improvement. The main goal of this kind of study is to get a "fi t" model to describe the evolution of a collaborative network by simulation iterations.

According to this framework, we will run the simulation in an iterative fashion. This means we will start from a basic model and run a simulation based on this model, verify and validate this model according to the empirical data, then adjust the model and related simulation parameters and run simulation again. We will repeat the iterations until we get the "fi t" model for the empirical data. In our study of SourceForge, we picked ER model as the basic model to start our iterations. Now we will explain the details of our simulation iteration steps according to the model it used.

- *Random network*

  We started from the ER model since it is the fi rst well-known complex network model and the base of the others. This model fi rstly included nonlinear dynamics displaying some surprising properties derived from the presence of phase transitions. One of them is the existence of well-defi ned boundaries separating order from disorder [**?**].

  Randomness is the dominant factor in the ER model. In our simulation, the collaboration network is based on the bipartite graph, where developer and project is relatively independent. This means that randomness in our simulation will be two parts, developer randomness and project randomness.

  For developer, the randomness is the randomness in action selection, which means that the action selections by developer are just random according to given probabilities. According to the model description in last chapter, we have six simulation parameters, which are used to defi ne the probabilities of each possible action. These are $P(CN)$, $P(JN)$, $P(CO)$, $P(JO)$, $P(AO)$ and $P(IO)$. $P(CN)$ and $P(JN)$ are used as the probabilities of action selections by new developers. For a new developer, there will be no other possible action except for creation and join, thus $P(CN) + P(JN) = 1$. In order to match the simulation with the empirical data, we used the statistical average percentages of these two actions among new developers from the empirical data for $P(CN)$ and $P(JN)$. These two probabilities are initial values, we adjust them in latter iterations. The value of $P(CN)$ is set to be 0.67331, and the value of $P(JN)$ is set to be 0.32668. For a "old" developer, there will be four possible actions, which are creation, join, quit and idle. There also exists equation like $P(CO) + P(JO) + P(AO) + P(IO) = 1$. By the same method, we will use the statistical average percentages of these action selections as the initial value. The results are as follows: $P(CO) = 0.018028$, $P(JO) = 0.005974$, $P(AO) = 0.0076881$ and $P(IO) = 0.96831$. Thus, the randomness in developer is implemented by random action selection based on fi xed probabilities given by these six simulation parameters.

  For project, the randomness is in project selection. When a developer decides to pick a project to join, there will be no preference, every project has the same probability to be picked. In our simulation, we just maintain a project list of all existing project and let developer randomly pick one when he decided to join a project.

- *Scale-free network*

  Scale-free network model is also one of the models in our simulation iterations. This model can reproduce all the topological and statistical properties we observed in empirical data, especially the power law in both developer distribution and project distribution. There are two major improvements made in scale-free network: growth and preferential attachment. Growth is inherent property in our simulation, so we just need to add preferential attachment in this model and related simulation.

In our model and simulation, preference will be in effect when a developer want to choose a action (developer preference) and when a developer select a project to join (project preference). For project preference, the probability a project will be selected is proportional to its degree. The probability of project i being selected is defined as

$$P(i) = \frac{\alpha k_i}{\sum_{j \in AllProjects} k_j} \tag{1}$$

For developer preference, the more projects the developer paticipated, the higher probability he will take an action(create, join or abandon) instead of idle. So in our simulation, The degree the developer already have is also one of the factors to determine the developer action probabilities. The actual function to calculate these probabilities will also based on our statistical result from the empirical data, which we will discuss in next section.

- *Scale-free network with constant fitness*

  Fitness is an appended attribute proposed by Dr Barab$\acute{a}$si for Scale-free network model to depict the phenomenon that young node may outrun old ones some time. We also implement scale-free with constant fitness in our simulation iterations. We defined the fitness as a parameter $\eta$for every project which does not change in time. Thus at every time a new project will be created, a new project j with a fitness η_j is added to the system, where η_j is chosen from a distribution $\rho(\eta_j)$. And the probability of participating a project i is proportional to the degree and the fitness of project i,

$$\prod_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{2}$$

  Theoretically the fitness distribution $\rho(\eta_j)$ may be any distributions (like normal or exponential distribution). we used the simulation to verify the decaying function of fitness that we get from the statistical analysis of the empirical data.

- *Scale-free network with dynamic fitness*

  In last chapter, we already discovered several phenomenon that scale-free network with constant fitness is not able to explain. We thus proposed a dynamic fitness factor in the scale-free network, which means the fitness for every project will decay by time. We will use simulation to verify our proposal in two aspects:

1. Whether the scale-free network model with dynamic fitness can still keep the properties we have for real networks, like power law distribution, high clustering coefficient and small diameter?
2. Whether the scale-free network model with dynamic fitness can make up the discrepancy between real network and scale-free network with constant fitness?

Also wee used the simulation to verify the decaying function of the dynamic fitness that we get from the statistical analysis of the empirical data.

3.3 Evolution related analysis

During our simulation iterations. We used those well-studied properties, like average degree, degree distribution, diameter and clustering coefficient, to verify the correctness of the models at every step. Also we also used other properties to validate the models at every step. These properties used for validation are the properties related to the evolution of the empirical data.

We will start our simulation iteration from ER model, this is not only for the theoretical reasons given in last section but also for the practical reason for our simulation iteration.

1. All the other models we will iterated can be generated from ER model by simple modifications, so we can repeat our simulation iterations smoothly if we started from ER model.
2. ER model is the most studied model. Almost all the measures we will use have been measured and mathematically proved. Started from this model, we can verify the basic model infrastructure by comparing the simulation results with theoretical results.

After running the simulation based on the ER model we described in last section, we have the simulation output. Then we will use the well-studied properties to verify the ER model. The first property we investigated is the clustering coefficient. The clustering coefficient (CC) of the simulated ER model is shown in Figure 2. We can see the clustering coefficient of ER model is below 0.24, which is much smaller than what we see in the empirical data of SourceForge developer collaboration network (where $CC \sim 0.75$). And it diminishes quickly when the overall system size increase. The relation between clustering coefficient CC and system size N is approximately

$$CC \sim N^{-0.14} \tag{3}$$

This observation is identical to the mathematical result of clustering coefficient of ER model, which proved $CC \sim N^{-\beta}$, β is constant smaller then 1.

Before we investigate other well-studied properties, we investigate the cluster distribution first. The cluster distribution of the simulated ER model is shown in Figure 3. The upper figure is the cluster distribution with normal coordinates. There are one big cluster existing, which is proved to appear in ER model

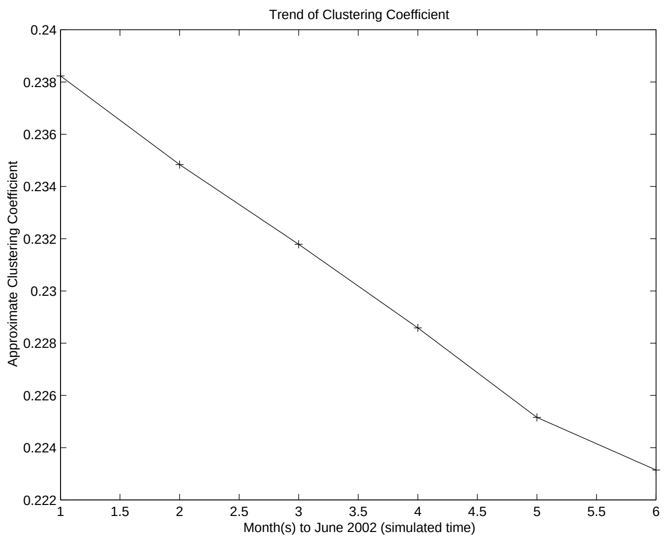

Figure 2: Clustering coeffi cient in ER

at some time in evolution. The process to generate this cluster is called percolation, which we discussed in chapter II. The lower fi gure is the cluster distribution with log log coordinates. The linear polynomial fi t is quite strange due to the heavy tail. Without consideration the tail effect, the cluster distribution of ER model has similar slope to the empirical data. This means ER model has similar characteristics with empirical data in clustering distribution. This is the only property the ER model can match with the empirical data. And every succeeding models can also match. So we will not include this property in discussions of succeeding models. However, the related simulation results can still be found in Appendix A.

Diameter is another attribute calling ER model into question. The diameter result of the simulated ER model is shown in Figure 4. In ER model, the average degree is decreasing by time while the diameter of the network is increasing. This result can fi t the mathematically proved function for diameter in ER model,

$$D \sim \frac{log(N)}{log(<k>)} \tag{4}$$

From last chapter, we know that the average degree is increasing while diameter of the network is decreasing in SourceForge developer collaboration network. The arbitrary value of the average degree of SourceForge (which is over 7) is bigger than the average degree of ER model (which is below 6) while the value of diameter of SourceForge (which is over 6) is also bigger than the diameter of the ER model (which is below 3).

Finally, we will investigate the degree distribution of ER model. The result of the simulated ER model is shown in Figure 5. The upper two fi gures are developer distribution and project distribution in normal

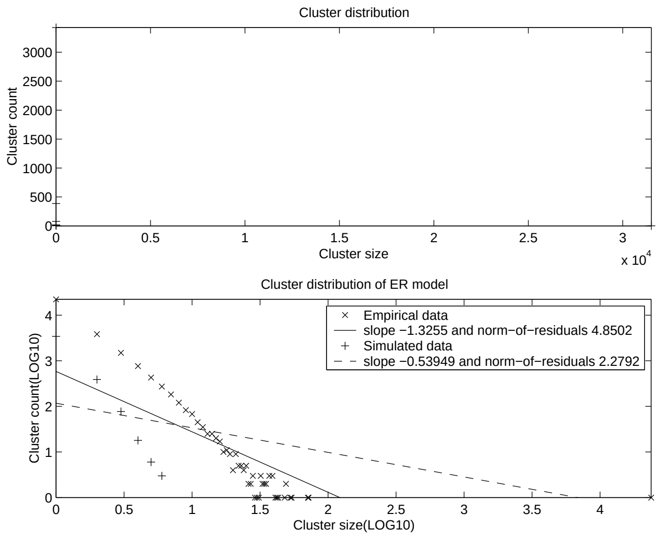

Figure 3: Cluster distribution in ER

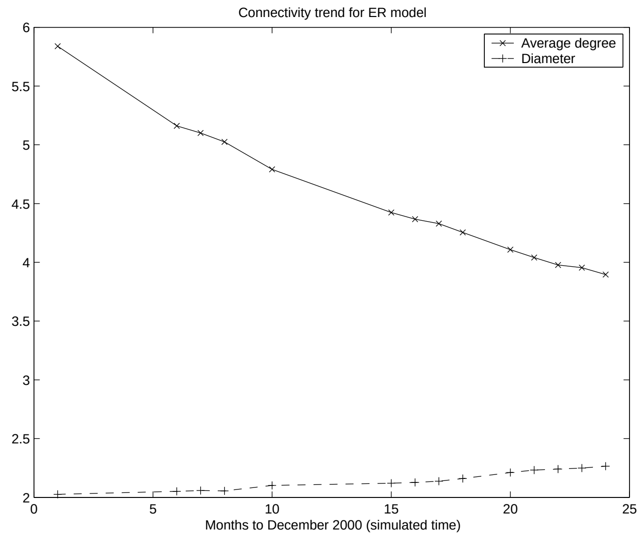

Figure 4: Average degree and Diameter in ER

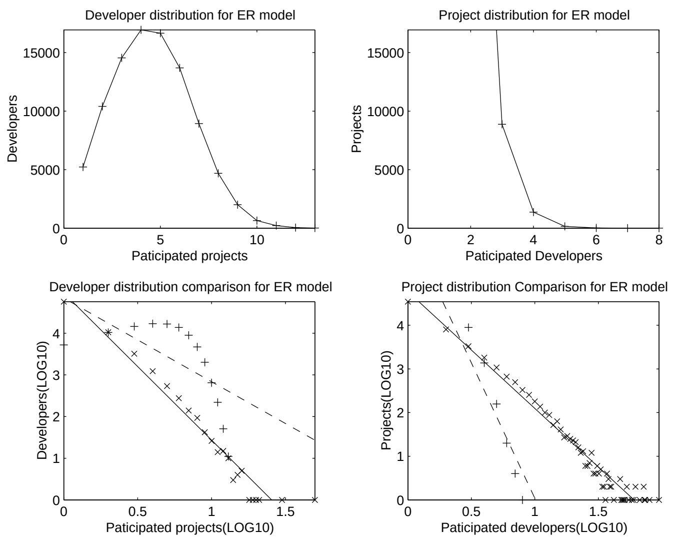

Figure 5: Developer and project distributions in ER model

coordinates. There are no power law in the distribution. Actually we can see a clear normal distribution in the developer distribution, which is proved mathematically for ER model. And we also compared the distributions with the distributions of empirical data in log log coordinates, which are shown in lower fi gures. For empirical data, the developer and project distributions can all fi t a straight line well, which is property of power law distribution. In detail, the developer distribution can fi nd a linear polynomial fi t with slope of -3.5311 and the standard deviation of 1.5639; the project distribution can fi nd a linear polynomial fi t with slope -2.6781 and the standard deviation of 1.6281. The ER model has signifi cant different distributions from our SourceForce network. Also we can see that for project distribution, the maximal developers in a single project is just 8, which is much less than the empirical data which is almost 100. This is because that the projects have the same probabilities when being picked, then no project will have preference to obtain signifi cantly more developers. Anyway, we can confi rm by our simulation that ER model is not the model for SourceForge developer collaboration network.

BA model is the next step in our simulation iterations. BA model is the fi rst complex network model that catches all the topological characteristics of real network, especially the power law distribution.

Before we verify the BA model by simulation results, we will explain the parameters we used in the simulation iterations. To be fair, all the simulation results listed here are all based on same simulation parameters. These parameters include the daily new developer rate $N(ND)$ and the six action probabilities we discussed before in the model description. These parameters are originally set according to the statistics of the empirical data. But we found the simulation cannot reproduce the empirical data with these “original”

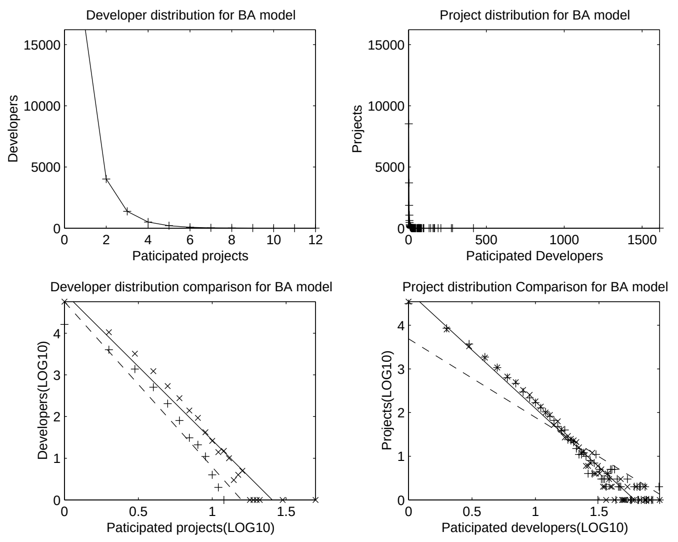

Figure 6: Developer and project distributions in BA model

parameters in iterations. So that we need to adjust these parameters and these adjustments are quite tricky due to the dependence of these parameters. The parameters we used in the iterations are listed as follows: 1) $N(ND)$ is random generated number averaged at 170 per day. 2) $P(CN)$ and $P(JN)$ are 0.1558 and 0.8442, which are same and fi xed for all models. 3) $P(CO)$, $p(JO)$, $P(AO)$ and $P(IO)$ are 0.001885, 0.0058, 0.00375 and 0.988565. These parameters are the actual parameters used in ER model, where no developer preference is applied. But actual parameters in BA or further models are varying, where developer preference is applied. In these models with varying parameters, the four values we given are just the average of the varying parameters.

To verify the BA model, we investigate the degree distribution fi rst. Since BA model is the fi rst to reproduce the power law distribution. The developer and project distribution of our simulated BA model are shown in Figure 6. In the fi gure, the upper two fi gures are the developer distribution and project distribution in normal coordinates and the lower two fi gures are their distributions in log-log coordinates. In lower fi gures, the “x” points are the distribution of empirical data and the solid line is its linear polynomial fi t; the “+” points are the distribution of simulated data and the dash line is its linear polynomial fi t. We observed that for the developer distribution, simulated data fi ts the empirical data quite well in both real point and the polynomial fi t; for the project distribution, the polynomial fi t line is not fi t well. The discrepancy between the polynomial fi t lines of simulated data and empirical data is only because of the tails. Without considerating the heavy tail, we can also fi nd it is a good match between empirical data and simulated data by inspecting the real points.

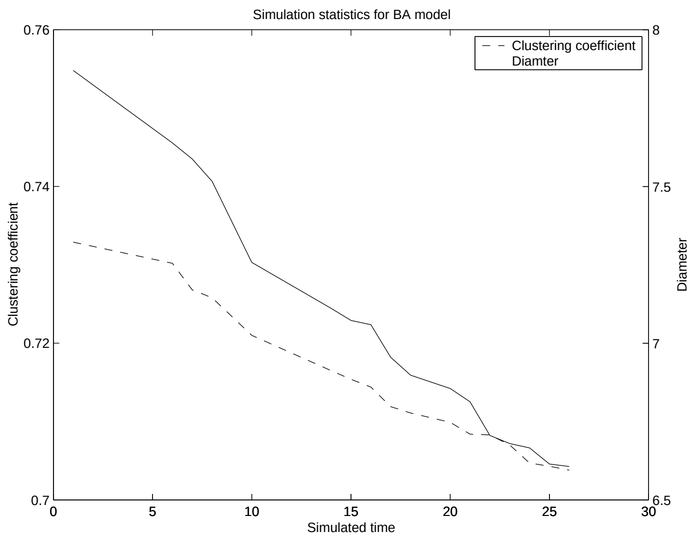

Figure 7: Diameter and Clustering coefficient in BA model

Then we will investigate the diameter and the clustering coefficient of simulated data based on BA model. The results are shown in Figure 7. The diameter is around 7 and decreasing. The clustering coefficient is around 0.7 and decreasing. These values all match the empirical data well.

BA model is proved to be defective since its failure to explain the phenomenon of "Young project may sometimes outrun old projects". The next model we investigate is BA model with constant fitness. The degree distribution of this model is already proved to be

$$P(k) \sim \frac{k^{-C-1}}{ln(k)} \tag{5}$$

where C is a constant parameter [**?**], which is a power law with logarithmic correction. We also proved that this model has a power law distribution by simulation, the result is shown in Figure 8. And we proved that the value and developing trend of diameter and clustering coefficient of this model match those of the empirical data[3].

Also at system level, BA model with constant fitness can match all the statistics of the empirical data, which means simulated data from the BA model with constant fitness fits empirical data in every well-studied properties, in other words that the BA model is verified to be a "fit" model. However we still find the deficiency of the BA model with constant fitness by validation. The problem is that it failed to reproduce the development of individuals (like developer or project) in the network. Following the same procedures we investigating the developments of individual projects in Chapter III, we used the new project set in July

[3]The figures are included in Appendix A

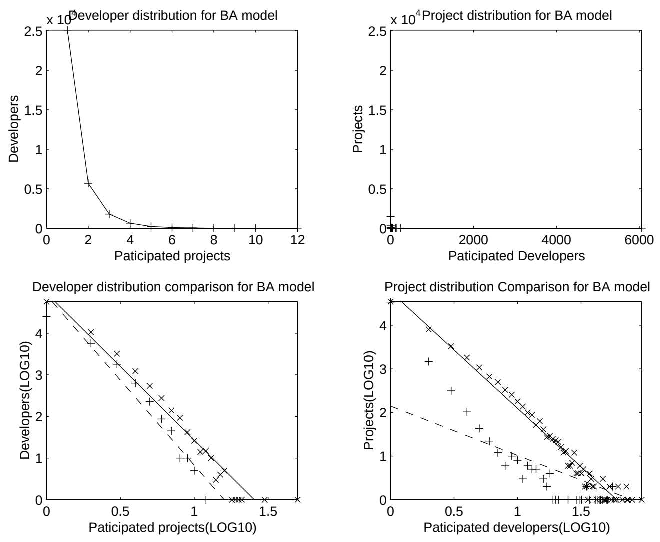

Figure 8: Degree distribution in BA model with constant fitness

2001(In our simulation, we reproduce the evolution of SourceForge developer collaboration network day by day from January 2001 and this July 2001 is simulation time, which means six simulated months after simulation started) and traced all the developments of these new projects. The average monthly increase is just 0.1748. The increase rate is decreasing from 0.9613 to 0. The maximum of the monthly increases is just 5. Most important, the serrate pattern is dominant in the developments of individual project, which is rarely found in the empirical data. These discrepancy is the problem of BA model with constant fitness.

To make up this discrepancy, we proposed a BA model with dynamic fitness. In our model, we define the fitness decays linearly by month. Fitness is a non-negative integer, which means the fitness can be 0 but no negative values. Every project will be allocate a random fitness when it is created by a developer, the system-wide fitness distribution is a exponential distribution as

$$P(f) = \lambda e^{-\lambda f} \tag{6}$$

where λ is equal to 2. By simulation, we proved our model can also match all the statistics (degree distribution, diameter and clustering coefficient) of the empirical data[4].

After analysis the simulation output of the BA model with dynamic fitness, we found that our model can reproduce the similar network as BA model with constant fitness in sense of the attributes we investigated in this thesis. Also we observed that the stationary state existed in the developing patterns of most projects instead of the serrated patterns in BA model with constant fitness. Also the average monthly increase rate

[4]The related figures are included in Appendix A

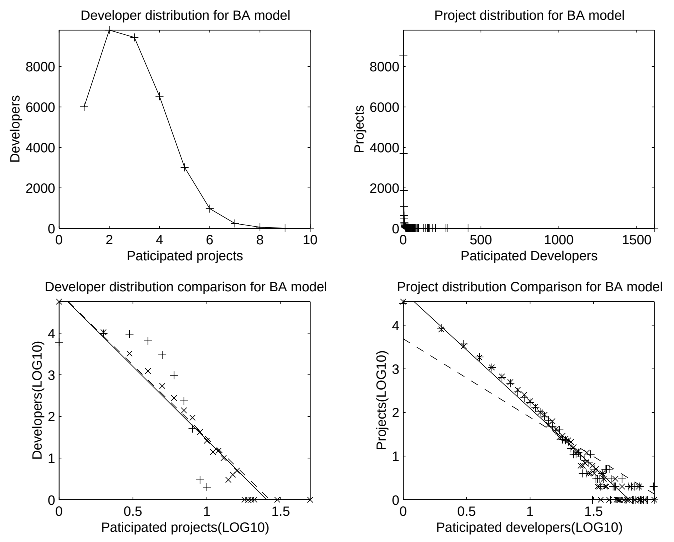

Figure 9: Developer and project distributions in partial BA model

of our model is slightly higher than that of the BA model with constant fi tness. The diminishing tread and maximal monthly increase rate of our model are similar to that of the BA model with constant fi tness.

3.4 Simulation discussions

In our simulation iterations, we have several interesting observations which need for further discussions.

- *Independence of developer distribution and project distribution*

  In the simulation iterations, we found the distribution of developer and the distribution of project are loosely related. Actually, in our experience, these two distributions are independent to each other. Figure 9 shows the degree distribution result of one of the simulation we have taken. This simulation is identical to the simulation of BA model except that we didn't apply developer preference in this simulation. Then in the fi gure, we can still found signifi cant power law in project distribution and the project distribution is almost identical to the project distribution we have in Figure 6, which has both developer reference and project reference. But the developer distribution in this simulation is normal distribution, which is due to the randomness of developer action selection in this simulation (without the developer preference). This developer distribution is different from the developer distribution in BA model shown in Figure 6, which is signifi cant power law. We have also found similar results in other simulations.

  This is because that our simulation is based on the bipartite graph. This structure separates the developer group from the project group. Developer distribution is determined by the developer preference and project is just abstractive object in action selection. So the project distribution or preference will not interfere with the developer distribution. For the same reason, the developer distribution or preference will not interfere with project distribution also. This independence of project and developer distributions lets us investigate the developer distribution and project distribution separately, thus makes the network simpler to inspect.

- *Fitness consideration*

  Fitness is a good patch for scale-free network model. The fitness can depict some exceptional phenomena of the network evolution theoretically and practically. Theoretically, with the consideration of fitness, the patched scale-free network model is proved to keep all the significant properties of the original scale-free network model (power law distribution, high clustering coefficient, "small world" phenomenon) while it can also explain the "Google" phenomenon, that young node (Google) can sometimes outrun old one (Yahoo!). Practically, we can understand that in most of the complex networks, the nodes are not equally weighted, and the weight is not only based on age. Some node may be more attractive even that they are new. This phenomenon can be easily discovered in many complex system like metabolism and internet evolution.

  Scale-free model with constant fitness proposed by A.L. Barabási describe the empirical data more accurately using statistical attributes of the whole network, but not completely according to the individual development patterns(e.g. life cycle of developer and project) of the existence of relatively long period of stationary state. This is because that in a scale-free network with constant fitness, the node with higher degree will always have higher preference, which is based on fitness and link counts, and thus make its degree constantly increasing. According to this, we proposed a dynamic fitness factor to be added to the scale-free network model. Dynamic fitness, implemented as an individual developer or project life cycles with declining fitness over time, kept all the existing statistical attributes of BA model and modeled the individual developer and project development patterns in the empirical data.

- *Life cycle consideration*

  In our simulation iterations. Besides the macroscopic statistical properties, we also inspect the properties of individual entities by statistical analysis. Therefor we found that the development patterns of both the developer and the project have life cycles. We will discuss them separately.

  For developer, there are several stages in a developer's life through out the SourceForge community. The first stage is "debut", the developer can only choose creating a project or joining a project. Then the developer will go into "growing" stage, in which the developer will has possibilities to take all

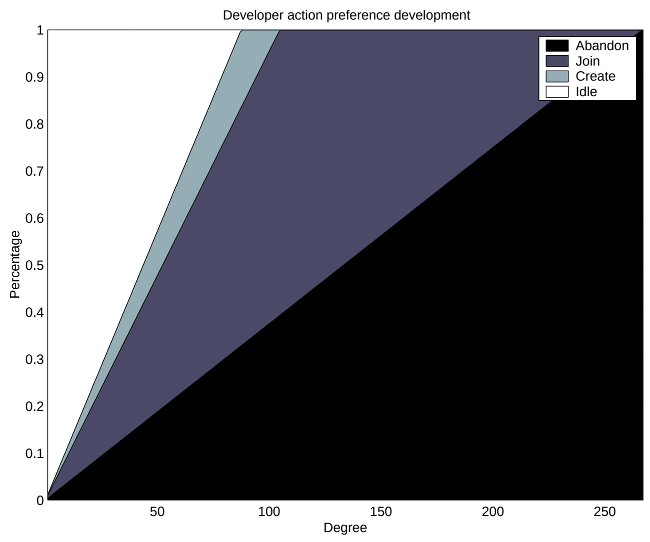

Figure 10: Developer action probabilities trend

four possible actions, but the idle action is dominant. With the number of projects the developer paticipated increasing, the developer will have higher and higher probabilities of taking active actions instead of idling. After that the developer will go into “mature” stage, where the probabilities of three active actions are dominant. With the continuous increasing the probabilities of active actions in “mature” stage, the developer will become more and more apathetical in creating new project. This is the start of “stable” stage, in which joining action and abandoning action have the dominant probability. In “stable” stage, the probability of abandon action will still increasing. Finally, when a developer reaches a saturate state, the only action he will take is abandoning. This is the beginning of the “regression” stage. We observed this kind of developer life cycle in empirical data and thus designed the developer preference according to this life cycle behavior. The developments of the relative percentages of the probabilities of these four actions in our simulation are shown in Figure 10.

Since our empirical data collection is not long enough, there are not enough information about the status of developer after “regression” stage. Thus in our simulation, we suppose that the “regression” stage is not the end of a developer’s “life”. The “regression” is just like a rollback of time, so that the developer is able to back to other stages to keep the developer “alive”.

For project, there is also life cycle alike behavior in the project development. We have already know the statistics of the project development about the empirical data in last chapter. In simulation, we collected the same statistics (monthly data of new project in simulated July 2001) from the scale-free network. The average monthly increase is also diminishing, but the arbitrary value of the monthly increase is approximate half of the empirical data and the maximum increases of every month are all about half of the empirical data. The maximal monthly increase of empirical data is 13, meanwhile the maximal monthly increase of BA model is just 5. But these discrepancies are all made up by adding constant fi tness in the model.

But there are still discrepancy in the individual project development between BA model with constant fi tness and empirical data. We found a common project development pattern in empirical data that it will increase fast in the beginning few months then the increase rate will diminish and the project will reach a stationary status for a period time, after that the size of the project will dramatically decrease. This stationary period of time can be seem as the mature state of the project, which is important in project development. But we cannot fi nd this stationary state in any of the complex network from ER model, BA model to BA model with constant fi tness.All of the project development is made up with size changing all the time. It is because the preference will not decrease, thus the size changing will always exists. Dynamic fi tness is able to fi t the empirical data better by implicitly applying a upper bound of project size.

- *Network evolution hypothesis*

  Not only the individuals like developer and project have life cycle, the whole SourceForge developer collaboration network may also have some life cycle alike evolution pattern. But we cannot may sure of this due to the limitation of empirical data.

  In our last chapter, we investigate the development trends of several statistics like average degree, clustering coeffi cient, diameter and major cluster. The fi rst three statistics are fuzzy and sensitive to the temporary change in the network. But we can fi nd clearly that the relative percentage of size of the major cluster in the network is approaching a fi xed percentage. We need more investigation to get the exact relation between this development trend and the network.

4 Conclusion

4.1 Summary of our works

Open Source Software (OSS) development is a classic example and prototype of collaborative social networks. Based on the empirical data we collected from SourceForge over the last two years, we are able to investigate the structure and the dynamical mechanisms that determine the topology and govern the evolution of such systems. SourceForge is the largest online collaboratory for open source software development projects, hosting over 50 thousand projects and over 80 thousand developers. We used data on projects and developers that were collected monthly start in January 2001. In this thesis, we analyze the empirical data we collected from SourceForge to obtain statistics and topological information of the OSS developer collaboration network. We extract the parameters of the evolution by inspecting the network over time. We generate a model that depicts the evolution of this collaboration network. Finally we use simulation to verify and validate the model we proposed by comparing the simulation output with the properties we obtained by statistical study of the empirical data.

In statistical study of the empirical data, we investigated not only the topological properties like degree distribution, diameter and clustering coefficient, but also the evolution related properties like the developing patterns of degree distribution, diameter and clustering coefficient. Though these investigations, we calculated all of these properties of the empirical data and compared with the properties of all existing models. We observed that even the newest complex network – BA model with constant fitness cannot depict all the existing properties of the SourceForge collaboration network, especially the developing patterns of the individuals (developer or project). Based on these observations, we proposed our model for the SourceForge collaboration network, which is based on dynamic fitness and bipartite graph.

In simulation study of the model of SourceForge collaboration network, we proposed a framework for modeling and simulation related study and apply this framework in our study of SourceForge collaboration network. We used the statistics we extracted from empirical data as the simulation parameters and iterated from ER model, BA model, BA model with constant fitness to BA model with dynamic fitness. Through simulation, we verified all the properties in these models that proved by mathematical methods (degree distribution) and proved the other properties that is difficult for mathematics to prove (diameter, clustering coefficient and developing patterns). By comparing the simulation output with the empirical data, we get the most "fit" model and related simulation parameter, which can reproduce all the investigated properties in this thesis.

4.2 Limitation of our work

Our work was based on the empirical data we obtained from SourceForge. However this empirical data is incomplete, because that 1) The SourceForge is still in progress, we can't have a complete set empirical data covering all the evolution of the collaboration network. 2) We don't have the complete set of empirical data of the past years (Empirical data for several months is not available in our empirical data). This "incompleteness" property may inference the research result we have.

Although SourceForge is one of the biggest and most famous hosting site in the OSS community. This is only a single case in the OSS community. It is improper to study the common topology and evolution of OSS community by investigating a single case – SourceForge.

Simulation toolkits always have some default setting for some simulation parameters that user can not change. Thus using single simulation toolkits may also bring errors into the simulation output. Unfortunately, we just used Swarm to run the simulation due to limited time. This may cause some imprecision in the simulations.

In the simulation iterations, we found that the parameters we get from the statistics of the empirical data is not exact fi t in our simulation. Thus we need adjust these parameters according to the simulation results. Unfortunately, these parameters(7 parameters) are dependent, it is really diffi cult to fi nd good parameters by naive “guess and try”. So we cannot guarantee the fi nal simulation we have is the best we can have.

4.3 Future work

According to the limitations of our work, there may be several future works as follows:

- We need more complete empirical data for experience.
- We may analyze other famous OSS hosting site like Savannah using the same methods.
- We will migrate our simulations to Repast to verify the inference of simulation toolkits to the simulation output.
- We will use more advanced method like operation research in tuning up the parameters to fi nd the best solution.

References

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

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

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

[4] G. Madey, V. Freeh and R. Tynan, *The Open Source Software development phenomenon: an analysis based on social network theory*, 8^{th} Americas Conf. of Information Systems, 2002.

[5] L. A. Adamic and B.A. Huberman, *Scaling behavior of the world wide web*, Science, 287(2115), 2000.

[6] Z. Neda, E. Ravasz, A. Schubert and A.L. Barab$\acute{a}$si, *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] D. Hiebeler, *The swarm simulation system and individual-based modeling,* Advanced technology for natural resource management, 1994.

[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 resules*, 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] Liuyuan Lu, W. Aiello and F. Chung, *Random evolution in massive graphs*, IEEE symposium of foundations of computer science, 2001.

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

[20] R. Kumar, F. Maghoul, P. Raghavan and *et. al.*, *Graph structure in the web: experiments and models*, Computer networks, 33(309), 2000.

[21] M. Barthelemy, L.A.N. Amaral and *et. al.*, *Classes of small-world networks*, Proceedings of the national academy of sciences, 97(21), 2000.