

Typeset with N_DThesis version 2.14 (2000/09/08)
on November 17, 2003

for
Yongqin Gao
entitled

TOPOLOGY AND EVOLUTION OF THE OPEN SOURCE SOFTWARE
COMMUNITY

This class conforms to the University of Notre Dame style guidelines established Fall 2000. However it is still possible to generate a non-conformant document if the published instructions are not followed! Be sure to refer to the published Graduate School guidelines as well.

This summary page can be disabled by specifying the `nosummary` option to the class invocation. (i.e., `\documentclass[nosummary]{ndthesis}`)

**THIS PAGE IS NOT PART OF THE THESIS, BUT
SHOULD BE TURNED IN TO THE PROOFREADER!**

N_DThesis documentation can be found at these locations:

<http://www.nd.edu/~afsunix/faq/tetexdoc/latex/ndthesis/>
<http://www.gsu.nd.edu/Committees/ITC/ndthesis.pdf>
http://www.gsu.nd.edu/Committees/ITC/sample_ndthesis.tar.gz

General L^AT_EX documentation and info:

On-line docs:

ND installation <http://www.nd.edu/~afsunix/faq/tetexdoc/>
T_EX User's Group <http://www.tug.org/>

Books:

A Guide... for Beg. & Adv. Users by Kopka/Daly
L^AT_EX User's Guide ... by Lamport
The L^AT_EX Companion by Goossens/Mittelbach/Samarin

Packages: (check on-line docs)

rotating sideways tables and figures
longtable multi-page tables
graphicx using Postscript and other figures

TOPOLOGY AND EVOLUTION OF THE OPEN SOURCE SOFTWARE COMMUNITY

A Thesis

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

by

Yongqin Gao, B.S.

Vincent W. Freeh, Director

Kevin Bowyer, Committee Chair

Graduate Program in Computer Science and Engineering

Notre Dame, Indiana

November 2003

TOPOLOGY AND EVOLUTION OF THE OPEN SOURCE SOFTWARE COMMUNITY

Abstract

by

Yongqin Gao

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. After two years of collecting developer and project information from SourceForge, we have sufficient data to determine the dynamic and structural mechanisms that govern the evolution and topology of this complex system. First, we analyzed the empirical data we received from SourceForge in order to ascertain the statistics of the topological information of the OSS developer social network. By inspecting the network's evolution over time, we were able to gain knowledge about the appearance of the complex network and characteristics of the system. Secondly, we generated a model to depict the evolution of this network. Finally, using Java/Swarm, we simulated the evolution of the OSS developer network using our model to compare against the real data. In simulation, we used several different models to replicate the empirical data: random network theory, scale-free network, scale-free network with constant fitness and scale-free networks with dynamic fitness. From these simulation iterations, we were able to verify and validate our model for this network and to increase our understanding of the OSS developer collaboration network.

Dedicated to my dear parents,
Chuanshan Gao and Xiuzheng Wang.
From whom I have received enormous
support, encouragement, and love.

献给我敬爱的父母
高传善 王秀珍

感谢他们给我的支持、鼓励和爱

CONTENTS

TABLES	v
FIGURES	vi
ACKNOWLEDGEMENTS	ix
SYMBOLS	x
CHAPTER 1: INTRODUCTION	1
1.1 Open source software phenomenon	1
1.2 Complex networks can model this problem	4
1.3 What is the challenge	6
1.4 Thesis layout	7
CHAPTER 2: STATE OF THE ART	8
2.1 Similar real networks in research	8
2.1.1 WWW and Internet	8
2.1.2 Collaboration networks	9
2.1.3 Ecological networks	10
2.1.4 Other networks	10
2.1.5 Simulation of social networks	11
2.1.6 Open source software community	12
2.2 Complex network models	13
2.2.1 Random network theory	13
2.2.2 Small world	16
2.2.3 Scale free network	19
2.2.4 Scale-free network with fitness	20
2.3 Agent-based computer simulation toolkits	22
2.3.1 Starlogo	23
2.3.2 Swarm	24
2.3.3 RePast	26
2.4 Summary	27

CHAPTER 3: STATISTICS OF NETWORK TOPOLOGY AND EVOLUTION	28
3.1 Topology statistics	28
3.2 Evolution statistics	39
3.2.1 Average degree is increasing	39
3.2.2 Cluster distribution also follows the power law	41
3.2.3 Diameter and clustering coefficient decays with time	43
3.2.4 Multiple preferences	52
3.2.5 Projects have fitness and life cycle	53
3.3 Summary	55
 CHAPTER 4: SYSTEM MODELING AND SIMULATION	 58
4.1 Experiment environment	58
4.2 Framework for simulation study	59
4.3 Modeling	60
4.3.1 Model description	61
4.4 Simulation models	62
4.4.1 Model one: random network (adapted ER model)	62
4.4.2 Model two: scale-free network (BA model)	71
4.4.3 Model three: scale-free network with constant fitness	77
4.4.4 Model four: scale-free network with dynamic fitness	78
4.5 Discussion	81
4.6 Summary	83
 CHAPTER 5: CONCLUSION	 85
5.1 Summary	85
5.2 Limitations	87
5.3 Future work	88
 APPENDIX A: EXTRA SIMULATION RESULTS	 89
 APPENDIX B: DOCKING EXPERIENCE BETWEEN SWARM AND REPAST	 96
 APPENDIX C: PROOF OF APPROXIMATE METHOD OF GRAPH DI- AMETER	 99
 BIBLIOGRAPHY	 101

TABLES

1.1	SOFTWARE LICENSING TAXONOMY (FROM THE “HALLOWEEN DOCUMENTS”)	2
3.1	SUMMARY OF STATISTICS	57
4.1	PARAMETERS FOR EMPIRICAL DATA	65
4.2	SUMMARY OF SIMULATION RESULT	84
A.1	ORIGINAL PARAMETERS FOR EMPIRICAL DATA, PART I	89
A.2	ORIGINAL PARAMETERS FOR EMPIRICAL DATA, PART II	95

FIGURES

1.1	Size increase survey about Apache (survey result by NetCraft.com) . . .	3
2.1	Random networks: p is the probability of edge existence, please refer to Definition 2.2.2	14
2.2	Small world: from order (1-lattice) to random (randomly rewiring based on p)	17
3.1	Transformation of the bipartite graph (from Newman, Strogatz and Watts 2001)	29
3.2	Degree distribution of SourceForge developer network on January 2003	37
3.3	Degree distribution of SourceForge project network on January 2003 .	38
3.4	Average degree of SourceForge (developer network): unit of X coordinate is month, 0 represents December 2000	40
3.5	Developer cluster distribution (developer network): the upper figure is in linear coordinates and the lower one is in log-log coordinates. . .	42
3.6	Relative size of major cluster (developer network): unit of X coordinate is month and 1 represents January 2001	44
3.7	Approximate diameter of the developer network decays with time (function of linear regression is $y = -0.0510x + 7.8717$ and the R_2 is 0.9619):unit of X coordinate is month and 0 represents December 2000	45
3.8	Approximate diameter of project network: unit of X coordinate is month and 0 represents January 2001	46
3.9	Approximate diameter vs. measured diameter (major cluster in the developer network): unit of X coordinate is month and 0 represents December 2000	47
3.10	Approximate clustering coefficient (developer network) decays with time: unit of X coordinate is month and 0 represents December 2000	49
3.11	Approximate clustering coefficient in project network: unit of X coordinate is month and 0 represents December 2000	50

3.12	Representative projects' evolution with time, unit of X coordinate is month and 0 represents June 2001	54
3.13	Average monthly growth (project size measured by developer count): unit of X coordinate is month and 0 represents June 2001	56
4.1	Conceptual framework for agent-based modeling and simulation	60
4.2	Clustering coefficient in ER: unit for X coordinate is month, and 1 represent July 2002	66
4.3	Average degree and diameter in ER simulation and the empirical data: unit of X is month, 0 represents December 2000 and simulated time 0	68
4.4	Cluster distribution in ER simulation and the empirical data	69
4.5	Developer and project distributions in ER simulation and the empirical data: 'x' represents empirical data and 'o' represents simulation data of ER model	70
4.6	Developer action probabilities trend	73
4.7	Diameter and clustering coefficient in BA simulation and the empirical data: unit of X coordinate is month and 0 represent December 2000	75
4.8	Developer and project distributions in BA simulation and the empirical data: 'o' represents simulation data of BA model and 'x' represent empirical data	76
4.9	Degree distributions in BA simulation with constant fitness and the empirical data: 'o' represents simulation data of BA model with constant fitness and 'x' represent 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	79
4.10	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	80
4.11	Developer and project distributions in partial BA model: "+" represents simulated data and "x" represents empirical data. There are no preference for developer, but there are preference for project	82
A.1	Average degree of project network: unit of X coordinate is month and 0 represents January 2002	90

A.2	Cluster distribution of BA model with constant fitness: ‘o’ represents simulated data and ‘x’ represents empirical data. The R^2 of the linear regression for simulated data is 0.5317	91
A.3	Cluster distribution of BA model with dynamic fitness: ‘o’ represents simulated data and ‘x’ represents empirical data. The R^2 of the linear regression for the simulated data is 0.4490	92
A.4	Diameter and clustering coefficient for BA model with constant fitness	93
A.5	Diameter and clustering coefficient for BA model with dynamic fitness	94
B.1	Developer Distributions	97
B.2	Community Size Development	98

ACKNOWLEDGEMENTS

I would like to express my great appreciation and gratitude to my advisors, Dr. Greg Madey and Dr. Vincent Freeh, for their patient guidance and constant encouragement through this research. All their suggestions, technically and non-technically, will be valuable assets to me in the time coming. Without their supervision, the thesis would have been impossible.

I am grateful to Dr. Kevin Bowyer for serving on my defense committee and for many insightful suggestions.

I am also grateful to my colleagues, Jin Xu, Jeffrey Goett and Yingping Huang, for helpful discussions.

I also acknowledge the assistance of Patrick McGovern, Director of SourceForge.net, and the contributions of University of Notre Dame students Chris Hoffman, Carlos Siu and Nadir Kiyancilar with their assistance with data collection on this project.

Finally, this research was partially supported by the US National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829.

SYMBOLS

N	number of nodes in the network
k	degree of a given node
$P(k)$	degree distribution of a network
$\langle k \rangle$	average degree of a network
DD	degree distribution, same as $P(k)$
D	diameter
CC	clustering coefficient

CHAPTER 1

INTRODUCTION

1.1 Open source software phenomenon

Open source¹ software development, despite usually consisting of volunteers dispersed worldwide, now competes with commercial software firms. This competition is due in part to closed-source proprietary software being associated with risks to users. These risks, often referred to as a “software crisis,” can be summarized as systems that take too long to develop, are too costly and do not work very well when delivered. Over the following years after software crisis first addressed, many unsuccessful solutions [16] were proposed in an effort to solve the software crisis but the approach promoted by Open Source Software (OSS) may finally present a good solution.

In 1984, Richard Stallman founded the Free Software Foundation², the first official organization dedicated to OSS. OSS should not be confused, however, with “freeware,” “shareware,” “use-restricted” software or “royalty-free” software. Table 1.1³ summarizes the differences between these various categories of software.

The purpose of Open Source Software was not to ensure distributing software to the end user without cost, but to ensure that the end user could use the software

¹“Open source” is a certification mark owned by the Open Source Initiative (<http://www.opensource.org>).

²<http://www.fsf.org/fsf/fsf.html>

³This is adapted from “The Halloween Documents”, which refers to a set of confidential internal Microsoft memos, leaked to the OSS community at the end of October 1998, which discuss the concerns of Microsoft at the threat posed by OSS.

Table 1.1. SOFTWARE LICENSING TAXONOMY (FROM THE “HALLOWEEN DOCUMENTS”)

Software Type	Zero Price	Redistributable	Unlimited Users and Usage	Source Code Available	Source Code Modifiable
Commercial (e.g., typical Microsoft products)					
Trial Software (e.g., time-bounded evaluation products)	X	X			
User-Restricted (e.g., Netscape Navigator, freely available and redistributable only to non-profit-making entities)	X	X			
Shareware (e.g., WinZip, which have a license that eventually mandates purchase)	X	X			
Freeware (e.g., Leap Frog, released in binary form only and in public domain)	X	X	X		
Royalty-free binaries (e.g., Microsoft’s Internet Explorer and NetMeeting, distributed in binary form only)	X	X	X		
Royalty-free libraries (e.g., class libraries)	X	X	X	X	
Open Source (e.g., Linux, Apache)	X	X	X	X	X

these developers collaborated with each other to develop software in such a fast, efficient and low-cost way. Understanding the mechanism of the OSS community may help us conceive a new software development model or at least address the software crisis.

SourceForge⁶ is used as the data source of our research since it is one of the largest and most famous OSS hosting web sites, offering features like bug tracking, project management, forum service, mailing list distribution, CVS and much more. As of April 2003, SourceForge hosted more than eighty thousand developers and fifty thousand projects. Thus the study of this community can let us understand more about the Open Source Software phenomenon. Furthermore, by obtaining an inside view and, hopefully, discovering the mechanisms producing the topology and evolution of the OSS collaboration network, we may predict the future of the OSS through modeling and simulation.

1.2 Complex networks can model this problem

Many well-studied networks have apparent generating principles, which means the topology of the network is deterministic by the given generating principle. These kind of networks are called simple networks. But there are also large-scale networks to which we can't attribute apparent generating principles, like friendship networks, power grids and natural neural networks. These networks have a mixture of randomness and structure and are called complex networks. Graph theory is the common method to study these networks. The graph used to study complex networks is called a random graph. Complex networks are decentralized and self-organized. The control and order in this kind of network is emergent rather than predetermined.

⁶<http://sourceforge.net>

The collaborative relationships between the developers forming this network are the focus of our study. These relationships can be represented by complex networks. So we used complex networks to model the SourceForge collaboration network.

Complex networks research began with the foundation of random network theory by Erdős and Rényi in 1959 [21]. Differing from the traditional networks computer scientists have studied, in which the generation of the networks are deterministic, randomness is a dominant factor in this kind of network, since there is no centralized control. The construction and evolution of a random network is determined by each individual node deciding its own connection based on the local knowledge it has. The topology of this kind of decentralized and self-organized network is unclear. Several important studies on complex networks have been conducted and provide methods of analyzing and understanding the underlying mechanisms of their topology. Random network theory(proposed by Erdős and Rényi) [21], Small world networks(proposed by Watts and Strongatz) [56] and Scale-free networks(proposed by Albert and Barabási) [5] are three important complex network models. The next chapter discusses these models in detail.

During those studies of complex networks, there were several important properties, which were often studied to characterize the topology of the complex networks. These properties are:

1. Degree distribution. The degree of a node, k , equals the total number of other nodes to which it is connected, while $P(k)$ is the distribution of the degree k throughout the network. Degree distribution in real networks was believed to be a normal distribution (when $N \rightarrow \infty$), but recently, Albert and Barabási and others found it fit a power law distribution in many real networks [7] (this will be discussed in the next chapter).
2. Diameter. The diameter of a network is the maximum distance (number of

hops or edges) between any pair of vertices. The diameter can also be defined as the average length of the shortest paths between any pair of nodes in the network. In our research, the diameter of the network characterizes efficiency because the time to propagate information is proportional to network diameter. Since the average value is more suitable for our purposes, we used the second definition in our research. Strictly speaking, the diameter of a disconnected graph (i.e. one containing isolated clusters) is infinite, but it also can be defined as the maximum diameter of its sub-clusters. Random graphs and real complex networks all tend to have small diameters. This is the phenomenon scientists referred to as the “small world phenomenon” .

3. Clustering coefficient. The neighborhood of a node consists of the set of nodes to which it is connected. The clustering coefficient of a node is the ratio of the number of links to the total possible number of links among the nodes in its neighborhood. The clustering coefficient of a graph is the average of all the clustering coefficients of the nodes. Recent research has found that real complex networks typically have a high clustering coefficient, which means that they exhibit a large degree of clustering [4].

These three properties will be significant in later discussion.

1.3 What is the challenge

The SourceForge community is one of the largest OSS communities. Our recent research is focused on the exploration of the internal mechanism of the OSS community [36, 37, 38]. The collaboration network of SourceForge is different from the traditional collaboration networks frequently studied (e.g., movie actor networks, or scientist collaboration networks). One of the big differences is the collaboration relationships they address. Traditional, well-studied collaboration networks merely

discuss the existence of the relationship, while it is the activities within these relationship networks that we are studying. In the collaboration network of SourceForge there were sometimes deletions of edges. In traditional well-studied collaboration networks, once an edge exists, it will remain persistent. For example, in movie actor networks, once an actor acts in a movie, he or she will always have a link to that movie. The link is persistent. The situation is different in the OSS collaboration network. The links between vertices can change. We notice an edge detachment process since a developer can quit the project at any time. Through our research, we also identified SourceForge as a collaboration network.

After obtaining a good theoretical model for the community, the next step was to use simulations to verify and validate it. By having the correct model and parameters, we were able to reproduce the evolution of the network. From this interior view of the network, we could understand, even predict and adjust, the evolution and future developments of the network.

Finally, the goal of this research was to find the underlying mechanisms of the topology and evolution of the OSS collaboration networks.

1.4 Thesis layout

The remainder of this thesis is structured as follows. Chapter 2 defines the state-of-the-art of related research and existing toolkits, focusing especially on their limitations. Chapter 3 discusses the statistics of the real data we collected from SourceForge over the last two years for the purpose of developing our model of the SourceForge community. Chapter 4 delves into the simulation we built for the SourceForge community and the verification and validation of the simulation. Finally, we give a conclusion of our work in Chapter 5.

CHAPTER 2

STATE OF THE ART

2.1 Similar real networks in research

Various real systems, ranging from communication networks to ecological webs, have proven to be complex networks [6]. A good understanding of the topology and evolution of these complex networks can aid in analyzing and understanding other real systems, such as the Open Source Software community. These real systems exist in different domains, but the networks are similar. The results can be shared among these related research projects. Actually, many of the references in this thesis also come from parallel research areas. Thus, in this section we will briefly review all other areas that have been studied by researchers aiming to uncover the general features of complex networks.

2.1.1 WWW and Internet

The WWW (World Wide Web) is currently the biggest global computer network. The web pages or the web sites constitute the nodes of the network and the hyperlinks pointing from one page(site) to another are the edges. Since its inception in December 1969, the size of this network grew to one billion nodes at the end of 1996 [55] [29] and is anticipated to expand 1000% in the next few years [13]. Because of its tremendous size and dynamic nature, its topology is still unavailable. The interest in the WWW as a network boomed after it was discovered that the degree distribution of the nodes follows a power law over several orders of magni-

tude [7]. That study is based on 325,729 sample nodes crawled from the WWW. A later survey of the WWW topology by Border *et al.* [40] used two 1999 Altavista crawls containing 200 million documents to discern similar attributes of the WWW topology. Adamic and Huberman [2] used a somewhat different representation of the WWW, with each node representing a separate domain and two nodes being connected if any of the pages belonged to one domain linked to any page belonging to the other. They all observed relatively high clustering coefficients, small diameters and a power law in the node degree distribution.

The WWW is a virtual network based on pages and sites. The infrastructure of the WWW, which is called the Internet, is also a network, which consists of physical links between computers and other telecommunication devices. There are two levels of topology for the infrastructure of Internet – router level and domain level. Faloutsos *et al.* [22] have studied both of them. High clustering coefficient, small diameter and a power law in the node degree distribution are also observed.

2.1.2 Collaboration networks

Recently, collaboration networks have gained more and more attention. Many of these networks can be precisely described by bipartite graphs [46]. The two typical well-studied examples are the movie actor collaboration network and scientific collaboration networks.

The movie actor collaboration network is based on the Internet Movie Database, which contains all movies and their casts since the 1980s. In that network, the nodes are the actors, and an edge exists between two nodes only if the two actors have acted in at least one movie together. This network expands with the database. Watts and Strogatz [58] studied this network from small world point of view. Albert and Barabási [4] have similar research based on a scale free network model.

The science collaboration network is similar to the movie actor network. The nodes are the scientists and two nodes are connected if the two scientists have written an article together. To uncover the topology of this complex network, Newman [45, 42, 43, 44] studied four databases spanning physics, biomedical research, high-energy physics and computer science over a five year window (1995-1999). Barabási [12] also investigated the collaboration network of mathematicians and neuroscientists publishing between 1991 and 1998. These two collaboration networks all have high clustering coefficients.

2.1.3 Ecological networks

Ecological systems can also be modeled as complex networks. One of the classic systems is the food web, which is used by ecologists to quantify the interaction between various species [48]. In this type of network, species are considered nodes, and every edge represents a predator-prey relationship between the two nodes connected. In a recent study, Williams *et al.* [60] investigated the topology of the seven largest and most documented food webs: Skipwith Pond, Little Rock Lake, Bridge Brook Lake, Chesapeake Bay, Ythan Estuary, Coachella Valley and St. Martin Island. Although these food webs are quite different in species, all exhibit a small world phenomenon. This result was supported by the independent investigations of Montoya and Solé [41] and Camacho *et al.* [19].

2.1.4 Other networks

There are also many other complex network related research studies. We will just focus on two of them. The first is the web of human sexual contacts. This network can be used to model the spread of many sexually transmitted diseases, including AIDS. Liljeros *et al.* [34] studied the web constructed from the sexual relations of 2810 individuals based on an extensive survey conducted in Sweden in 1996. This

complex network research can help us understand the spread of the disease. Recent research also found that there was almost no strategic method to stop the spread of the diseases over a scale-free network [47].

The second network involves a large, directed graph constructed from long-distance telephone call patterns. In this scenario, the nodes are phone numbers, and every completed phone call directed from the caller to the receiver is an edge. Abello, Pardalos and Resende [1] studied the call graph of long-distance telephone calls made during a single day. In this calling graph, every caller or callee is represented by a node. An edge connecting two nodes depicts a call made between them. After generating the network, they found that it also has a degree distribution of a power law, which is defined as a distribution following $y = x^\alpha$, where α is constant.

2.1.5 Simulation of social networks

Axelrod has done a good deal of work in simulation of social networks. Part of his research [11] included the replications of eight popular simulation models using Swarm (a simulation toolkit which will be discussed later in this chapter) and comparing the results to the original. These models include Conway's game of life [49], Schelling's tipping model [54], Axelrod's evolution of prisoner's dilemma strategies [9], March's organizational code [39] and Riolo's prisoner's dilemma tag model [53]. These models are selected for their simplicity, diversity and reasonable run times. As result, Axelrod concluded – "In most cases, the results were so close that we probably attained distributional equivalence for all the models." In his paper, he also discussed the docking of simulation (Docking is an alignment process and experiment for verifying simulations. detailed definition can be found in [11]). We also did the docking for our simulation, which is discussed in the Appendix B.

Prietula, Carley and Gasser [35] discussed the importance of computational mod-

eling and simulation in organizational design, analysis and reengineering, especially in large scale transnational organizations. They also discussed the existing methods in modeling and simulation of social networks like organizations.

2.1.6 Open source software community

The Open Source Software community can be described as a collaboration network. In this network, the nodes are developers and projects. There is a link between a developer and a project if the developer joins the project. This network is similar to the two collaboration networks mentioned earlier (movie actor and scientist collaboration network) because of common features like the bipartite property, small world phenomenon and high clustering coefficient. But the OSS network also differs from these traditional networks. In those two traditional collaboration networks, the edge (link, relationship) is persistent, which means it will never be removed from the network after it was added. This is not the case in the Open Source Software community, where the edge can be removed if the developer (node) quits the project (node). Thus the collaboration network in OSS is based on active relationships instead of once-present relationships found in many collaboration networks. The detachment property is one way that makes the complex network for Open Source Software community different.

Previous research on the complex networks was often based on snapshots [58, 45, 6], which focused on the topology of the network at a given point in time. Since we have collected the data over two years, 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 to the evolution of the overall network system. The papers related to this research have been published in proceedings of NAACSOS

(North American Association for Computational Social and Organizational Science) 2003 [23, 24] and SwarmFest 2003 [25].

2.2 Complex network models

Complex networks are not a new topic in scientific research. There are many existing models used to depict complex networks. In this section, we will review these models to aid in understanding and conceptualizing our work.

2.2.1 Random network theory

Random network theory, proposed by Erdős and Rényi is the first model to depict complex networks [21]. We will call this the “ER model” in the remainder of this thesis. These kinds of random graphs do not model our real complex networks correctly. However, they do constitute an important limiting case and are historically the root of further complex network model research. We will go through some relevant terminology and definitions, and a few significant results of this model.

Erdős and Rényi define the random graph as N labeled nodes connected by m edges, which are chosen randomly from the $N(N-1)/2$ possible edges [21], as shown in Figure 2.1. There are two similar definitions for random graphs by Bollobás [15].

Definition 2.2.1 $G(N, m)$ is a labeled graph¹ with vertex set $V(G) = \{1, 2, \dots, N\}$, having m randomly chosen edges (where m usually depends on N). $G(N, m)$ is frequently abbreviated as G_m .

Definition 2.2.2 $G(N, p)$ is a labeled graph with vertex set $V(G) = \{1, 2, \dots, N\}$, in which every one of the possible $\binom{N}{2}$ edges exists with probability $0 < p < 1$, independent of any other edges. $G(N, p)$ is frequently abbreviated as G_p .

These two definitions are similar in most cases, and they are practically interchangeable when $m \approx pN$. G_p uses probability p as a parameter and G_m uses m , the

¹labeled graph means a graph with each node labeled differently but arbitrarily.

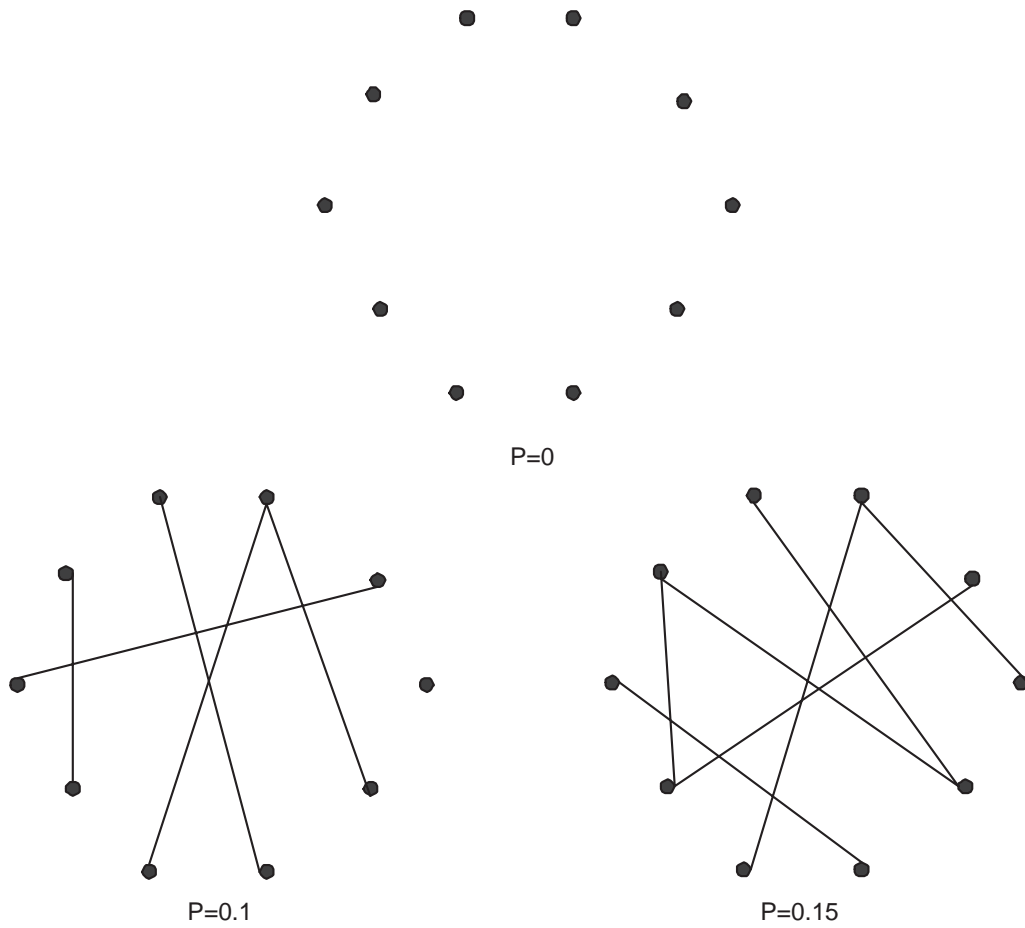


Figure 2.1. Random networks: p is the probability of edge existence, please refer to Definition 2.2.2

number of edges, as a parameter. Since p has a single value of the network, single value for the whole network is much easier to deal with than considering all the edges. We will just use G_p in the following discussions.

Random graph theory defines a theoretical method and parameters needed to generate a graph and guarantees that the graphs generated under similar conditions will have similar properties. The research on random graphs is focused on the emergence of some given properties Q when $N \rightarrow \infty$, where N is the size of the network. For example, connectedness of a random graph is one of the properties.

One of the most striking results of random graph theory is that most monotone properties² appear suddenly. This is explained by Duncan Watts [56] as the following. “That is, there exists a threshold function $M^*(n)$ that determines whether or not a graph is either very unlikely or very likely to have property Q . The important thing to understand is that if we imagine random graphs as dynamic networks, growing in time, then the appearance of practically any property of interest will occur on a time-scale that is very short compared with the time-scale of the whole process.” In a random graph with connection probability p , the degree k_i of a node i follows a binomial distribution with parameters $N - 1$ and p :

$$P(k_i = k) = C_k^{N-1} p^k (1-p)^{N-1-k} \quad (2.1)$$

Réka Albert and Barabási [6] proved the degree distribution of random network is as:

$$P(k) \approx e^{-pN} \frac{(pN)^k}{k!} = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \quad (2.2)$$

where $\langle k \rangle$ is the average degree of all vertices in the network. Thus with a good approximation the degree distribution of a random graph is a binomial distribution.

²Monotone means just that if a particular random graph G possesses Q , then any graph H that includes G as a subgraph will also have Q . (Defined by Erdős and Rényi.)

Of all possible properties Q , connectedness has received more attention than the others. A disconnected graph has infinite diameter D ; only a connected graph will have finite diameter D . A famous theorem by Erdős and Rényi [21] guarantees that almost any random graph with more than $N/2\ln(N)$ edges (equivalent to $\langle k \rangle \geq \ln(N)$) will be connected. And the general conclusion for the diameter D of a connected random graph is that for most values of p , almost all graphs with the same N and p have precisely the same diameter [3], which is given by:

$$D = \frac{\ln(N)}{\ln(pN)} = \frac{\ln(N)}{\ln(\langle k \rangle)}. \quad (2.3)$$

The diameter of the network is logarithmic to the system size. This means diameter will increase with respect to the size of the network.

Cluster coefficients are another important property. If we consider a node in a random graph along with its nearest neighbors, the probability that two of these neighbors are connected is equal to the probability that two randomly selected nodes are connected [6]. Consequently the clustering coefficient of a random graph is

$$C_{rand} = p = \frac{\langle k \rangle}{N} \quad (2.4)$$

However, real networks typically do not follow the prediction of random graphs in their clustering coefficients.

2.2.2 Small world

Watts and Strogatz [57] found the cluster coefficients of real networks similar to large ordered lattices. Based on this observation, they proposed their complex network model, called the “WS model” in the rest of this thesis. The algorithm used to build the model as explained by Barabási [6] is the following:

1. Start from order: It starts with a perfect 1-lattice, in which each vertex has precisely k neighbors ($k/2$ on either side), where $N \gg k$ and $\ln(N) \gg 1$.

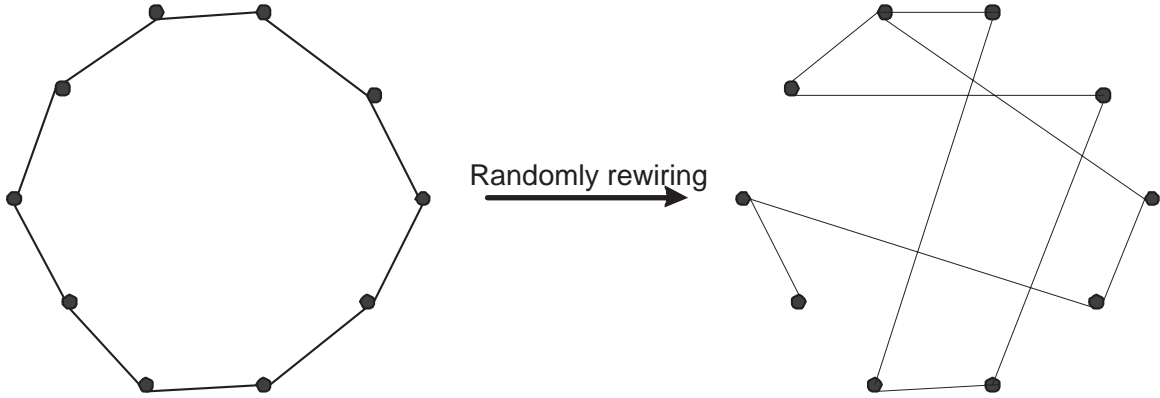


Figure 2.2. Small world: from order (1-lattice) to random (randomly rewiring based on p)

2. Randomize: Randomly rewire each edge of the lattice with probability p . This process introduces $pNk/2$ long-range edges crossing the lattice. By varying p , we get precisely a 1-lattice when $p = 0$; perfect approximation to a random graph when $p = 1$.

Figure 2.2 shows an example of generation of a small world network with random edge rewiring. A much-studied variant of the WS model was proposed by Newman and Watts [46], in which edges were added between randomly chosen pairs of nodes, but no edges were removed from the regular lattices.

The diameter of the WS model is dependent on the system size. There is a change in the scaling of the diameter with respect to probability p . For small p , D scales linearly with the system size, while for large p the scaling is logarithmic [6]. As discussed by Watts [56], the origin of the rapid drop in D is indicated by the appearance of shortcuts between nodes, which will have significant impact on the diameter by connecting the distant parts of the lattice. He proved that even a relatively low fraction of shortcuts is sufficient to drastically decrease the diameter, yet locally the network remains highly ordered. The diameter is logarithmic to the

system size N . This is called the small world phenomenon, which has become so familiar recently [56].

WS models also have a relatively high clustering coefficients (CC). Following the building sequence of the WS model of a regular lattice ($p = 0$), the clustering coefficient does not depend on the size or other parameters of the lattice but only on the topology of the lattice itself. After rewiring the edges of the lattice based on probability p , it shows that the clustering coefficient of the rewired graph is still similar to CC at $p = 0$, when p is not too large. The related equation of clustering coefficient for the WS model is

$$CC(p) = \frac{3K(K-1)}{2K(2K-1) + 8pK^2 + 4p^2K^2} \quad (2.5)$$

where K is the degree of each node in the original lattice and p is the probability of rewiring edges [6]. This clustering coefficient is independent of the system size and it is relatively high, which matches the results obtained from the real networks.

Next, consider the degree distribution, which contains a discrepancy between the WS model and the real-world networks. As Albert and Barabási [6] discussed, in the WS model for $p = 0$, each node has the same degree K . Thus the degree distribution is a delta function centered at K . After randomization, the degree distribution maintains the average degree equal to K . They have also calculated the distribution as

$$P(k) = \sum_{n=0}^{f(k,K)} C_{K/2}^n (1-p)^n p^{K/2-n} \frac{(pK/2)^{k-K/2-n}}{(k-K/2-n)!} e^{-pK/2} \quad (2.6)$$

for $k \geq K/2$, where $f(k, K) = \min(k - K/2, K/2)$.

The shape of the degree distribution is similar to that of a random graph. It has a pronounced peak at $\langle k \rangle = K$ and decays exponentially for large k . The WS model does not fit the real networks in degree distribution.

2.2.3 Scale free network

From inspection of several real networks, Barabási found that many large networks are scale free, that is, their degree distribution follows a power law for large k . The ER model and WS model do not reproduce this feature.

The power law degree distributions observed in networks were investigated by Barabási and Albert [5]. They proposed a new network model, called the BA model in the remainder of this thesis. They proposed this new model based on the idea that the scale-free nature of the real networks is rooted in two generic mechanisms.

- Growth- The network models discussed so far assumed that we started with a fixed number N of vertices that are randomly connected or rewired without modifying N . But, most real networks describe systems that grow by the sequential addition of new nodes.
- Preferential attachment- The network models that have been mentioned were based on an assumption that the probability of two nodes being connected (or their connection being rewired) was independent of the nodes' degree. However, in most real networks, the probability of connecting a node is related to the node's degree, which is called preferential attachment or the “rich get richer” rule.

Thus, the algorithm of the BA model as Barabási proposed [5] is the following.

1. Starting with a small number m_0 of nodes, at every time step, a new node is added with $m \leq m_0$ edges linking the new node to m different nodes already present in the system.
2. When choosing the nodes to which the new node would connect, Barabási and Albert assumed that a new node will be connected with the probability P to node i depends on the degree k_i of that node i , such that

$$P(k_i) = \frac{k_i}{\sum_j k_j} \quad (2.7)$$

Over t time steps, this procedure results in a network with $N = t + m_0$ nodes and mt edges. Numerical simulations indicated that this network evolved into a scale-invariant state with the probability that a node had k edges, which followed a power law with exponent $\gamma_{BA} = 3$. This is called the “power law distribution,” which exists in many real networks but not in any networks generated by the ER or WR model.

Barabási and Albert also proved that the diameter of the networks generated by their model was smaller than the diameter of a network generated by random network theory with the same size and average degree.

As to the clustering coefficient, there is no analytical prediction in the BA model. Barabási and Albert [6] compared a BA model network with average degree $\langle K \rangle = 4$ to the random network with clustering coefficient $CC_{rand} = \langle k \rangle / N$. They found that the clustering coefficient of a scale free network was about five times higher than that of a random graph, and this factor slowly increased with the number of nodes. However, the clustering coefficient of the BA model decreased with the network size, following approximately the power law $C \sim N^{-0.75}$. But the clustering coefficient of the BA model is still different from what was found in the WS model, which is independent of network size N .

2.2.4 Scale-free network with fitness

A scale-free network is a good complex network model to fit real networks. This model-generated network fits a real network in most of the global statistical features – degree distribution, diameter and clustering coefficient. However, it lacks some real network’s attributes, especially in regards to the developing patterns of a single vertex in the network.

Researchers have discovered that some special networks have single-scale exponential distributions or have degree distributions where power-law scaling is followed by an exponential cutoff for large k . Amaral *et al.* [8] suggested incorporating aging and cost or capacity constraints in order to solve these deviations. Their model also followed growth and preferential attachment, but when a node reached a certain age or had more than a critical number of edges, new edges could not connect to it. Numerical simulations indicated that while for small k the degree distribution still followed a power law, for large k an exponential cutoff developed, inconsistent with the power law for real networks.

Also, numerous examples indicate that in real networks a node's degree and growth rate do not depend on age alone. Bianconi and Barabási [14] used another new feature in the model – fitness. They proposed a model in which each node is assigned a fitness parameter η_i which does not change over time. Thus at every time step a new node j with a fitness η_j is added to the system, where η_j is chosen from a distribution $\rho(\eta)$. Each new node is connected to some of the nodes already in the network, and the probability of connecting to a node i was proportional to the degree and the fitness of node i ,

$$\Pi_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \quad (2.8)$$

They also verified that the degree distribution of the network generated by their new model is

$$P(k) \sim \frac{k^{-C-1}}{\ln(k)} \quad (2.9)$$

which is a power law with a logarithmic correction.

Through our study of Open Source Software community data, we found these models did not match the empirical data. Thus we proposed our new model for the Open Source Software community, which will be discussed in Chapter 3.

2.3 Agent-based computer simulation toolkits

Computer simulation, which integrates scientific visualization, mathematical modeling, knowledge-based design aids, intelligent user interface and many other techniques for creating platforms for trying out new ideas and doing research, was first developed in 1953 (by Enrico Fermi and his colleagues) and has gained increasing attention ever since. For example, in a July 1990 article in the *Washington Post* argued for a national “super highway” for computer information. Senator Albert Gore of Tennessee described the essence of simulation without using the word:

It is hard to understand an ocean because it is too big. It is hard to understand a molecule because it is too small. It is hard to understand nuclear physics because it is too fast. It is hard to understand the greenhouse effect because it is too slow. Supercomputers break these barriers to understanding. They, in effect, shrink oceans, zoom in on molecules, slow down physics, and fast-forward climates. Clearly, a scientist who can see natural phenomena at the right size and the right speed learns more than one who is faced with a blur. [33]

Modeling is always one of the most important parts of a simulation. Mathematical methods (ODE, PDE, statistical approaches) are traditional modeling methods, but they are not sufficient enough in increasingly complex system simulations. These methods can describe macroscopic properties of a system, but fail to explain the origin of those properties. These methods cannot be easily extrapolated into situations where the assumptions behind the equations no longer hold. Also these methods are inadequate in handling discontinuous systems and heterogeneity in populations [27].

Agent-based modeling can complement and enhance these traditional approaches. An agent is a self-contained entity with internal behavior and attributes. The typical components of an agent include internal data representations and methods of modifying their internal data and environment.

We used agent-based computer simulation to assist in analyzing and understanding the underlying mechanisms that produced the topology and evolution of

the Open Source Software community. There are normally four major objects in an agent-based simulation – model (the overall control of the simulation model), agent (the individual in the simulation), space (the place the individual is in) and probe (the information collector for the simulation). In the rest of this section, we will review some popular agent-based simulation toolkits.

2.3.1 Starlogo

Starlogo is a programmable modeling environment used for exploring the behaviors of decentralized systems, such as bird flocks, traffic jams, and ant colonies [31]. Starlogo was first developed as a parallel programming language for the massively parallel Connection Machine [52]. Starlogo now has a Java-based version after having a Macintosh-only version for several years. Netlogo is an upgraded version of Starlogo in the sense that Netlogo is a cross-platform and parallel-ready [26].

In Starlogo, there are three main types of objects:

- Turtles. The main inhabitants of the Starlogo world are graphic creatures known as “turtles”. Each turtle has a position, a heading, a color and a “pen” for drawing although more specialized traits and properties can be added. In Starlogo, the actions and interactions of thousands of turtles can be controlled in parallel. This is the “agent” in the Starlogo.
- Patches. Patches are pieces of the world in which the turtles live. Patches are not merely passive objects upon which the turtles act since, like turtles, patches can execute Starlogo commands and act on turtles and other patches. Patches are arranged in a grid, with each patch corresponding to a square in the Graphics area. The grid of patches works like a cellular automata. This is the “space” in Starlogo.

- Observer. The observer “looks down” on the turtles and patches from a bird’s eye perspective. The observer can create new turtles, and it can monitor the activity of the existing turtles and patches. This is the “model” and “probe” in the Starlogo.

A simulation in Starlogo is composed of turtles (agents), patches and an observer. The observer initializes the environment, generates the patches and creates the turtles. Then turtles and patches act and interact according to their own rules. The observer also has the responsibilities of monitoring and recording the status of the simulation.

Starlogo was designed especially for use by students. It has no supportive libraries, and the overall design is not flexible enough for complex simulation. We did not use it as one of our simulation toolkits.

2.3.2 Swarm

Swarm is a simulation toolkit which facilitates development and experimentation with simulations involving a large number of agents which behave and interact within a dynamic environment [28]. The Swarm system provides a framework that allows people to write code describing the details of their specific problems and then gain easy access to user-interface, simulation management and analysis tools.

Using an object-oriented method, all entities in Swarm are called objects. An object has three main characteristics: *Name*, *Data* and *Rules*. An object’s name consists of a unique ID tag, made of the type and module name, which is used to send messages to that object. The data is the internal information in the object. The rules are a set of functions that handle messages that are sent to the object. There are several special objects within a Swarm simulation.

- Agents. Agents are the objects written by the user. They represent entities

that exist within the model. There are several components that they must have: internal state-variables, step function (which is invoked on every time step) and action functions (which handle messages sent to the agent by other objects).

- Space. Rather than building assumptions into Swarm about the type of environment agents would move around in, the notion of space was encapsulated within objects. The space keeps track of the locations of any agents that are located in the space. Space is normally a two- or three-dimensional lattice space with spatial variables at every grid. Functional spatial variables are also allowed, where value lookup is performed via a function rather than by maintaining an array of values.
- Probe and graph. Probes and graphs are objects in Swarm. There are two standard messages that all agents in Swarm are required to recognize. A probe message is used to retrieve some internal state variables from an agent. The set message is used to place an internal state variable in an object. Probes and graphs use these two messages to obtain and set the designated internal state variable.
- Object List Manager (OLM). Swarm needs to keep a list of all the objects in the simulation – regular agents, space object, analysis objects, and so on. In time step simulations, all of the objects need to have a “step” message sent to them at every time step. Swarm encapsulates these control functions in an object called the *Object List Manager*. It maintains lists of all objects and valid object-types in the system, handles registration of new objects and allocates system wrappers for the objects by sending messages to populations and destroying objects and other system tasks.

Simulation in Swarm is implemented by creating an OLM. The OLM will generate space, new agents and probes. Then it sends “step” messages to every registered object on every time step. The objects will interact with each other in the simulation. This is a traditional time-related simulation. Swarm also supports event-triggered simulations. Details about Swarm can be found in [27]. Meanwhile, Swarm also provides many supportive libraries, like random number generators and distribution support. Its numerous support libraries and flexible framework led us to choose it as our current simulation toolkits.

2.3.3 RePast

RePast is a software framework for creating agent-based simulations using the Java language. It provides a library of classes for creating, running, displaying and collecting data from an agent based simulation. In addition, RePast can take snapshots of running simulations, and create quicktime movies of these simulations. RePast borrows much from the Swarm simulation toolkit and can be properly termed “Swarm-like.”

RePast has an almost identical framework to Swarm, including agents, space, models and probes. RePast, however, is a purely Java based programming whereas Swarm was originally written in Objective C and extended to Java using JNI. Compared to Swarm, RePast has the following advantages:

1. Integrated Developing Environment (IDE). SimBuilder is an IDE provided for developers to write, manage and debug RePast projects.
2. Understandable library interface. Swarm in Java is extended from Object C thus the methods and variables all have long and strange name to keep them similar to the original objective. RePast does not have this disadvantage.

3. Debugging support. Since the core part of the toolkit is still derived from the convoluted Object C, debugging of the simulation in Swarm is difficult, because the project cannot be trace if the execution goes into the core. Since RePast is pure Java, debugging is much more straightforward for developers.

RePast has more advantages in several aspects than Swarm. However, it does have a similar library interface, so we will try to migrate our simulation to RePast in future work. Other detailed information about RePast can be found at their web site [30].

2.4 Summary

In this chapter, we reviewed some of the existing research related to this thesis, including complex network research and social network simulations. We also distinguished our research from the related researches. Then we introduced the existing complex network models, which we will use in the following chapters. Finally, we explained several agent-based simulation toolkits.

CHAPTER 3

STATISTICS OF NETWORK TOPOLOGY AND EVOLUTION

3.1 Topology statistics

Complex network related research became popular because of random network theory. Degree distribution, diameter and clustering coefficient are also used as popular attributes to describe a network system. The development of these attributes is related to the evolution of complex networks. In this chapter, we will investigate the attributes of the empirical data that we collected from SourceForge in order to understand that collaboration network. Also we use these statistics for further modeling and simulation.

Before discussing these attributes, we need to explain the developer collaboration network that we studied. From SourceForge, we collected data on two major entities – developers and projects. In this data, only one relationship existed– the participation between developer and project. There were no direct links between developers or between projects. So, we looked at this network as a bipartite network, where projects and developers were the two kinds of nodes, and edges could be only connected to different kinds of nodes. The developer network we studied is a transformation of this bipartite network. In the developer network, there is only one type of node representing the developer in the collaboration network and the edges in the network represent the relationship of collaboration. For every pair of nodes i and j , there is an edge connecting i and j only if i and j are collaborating in at least

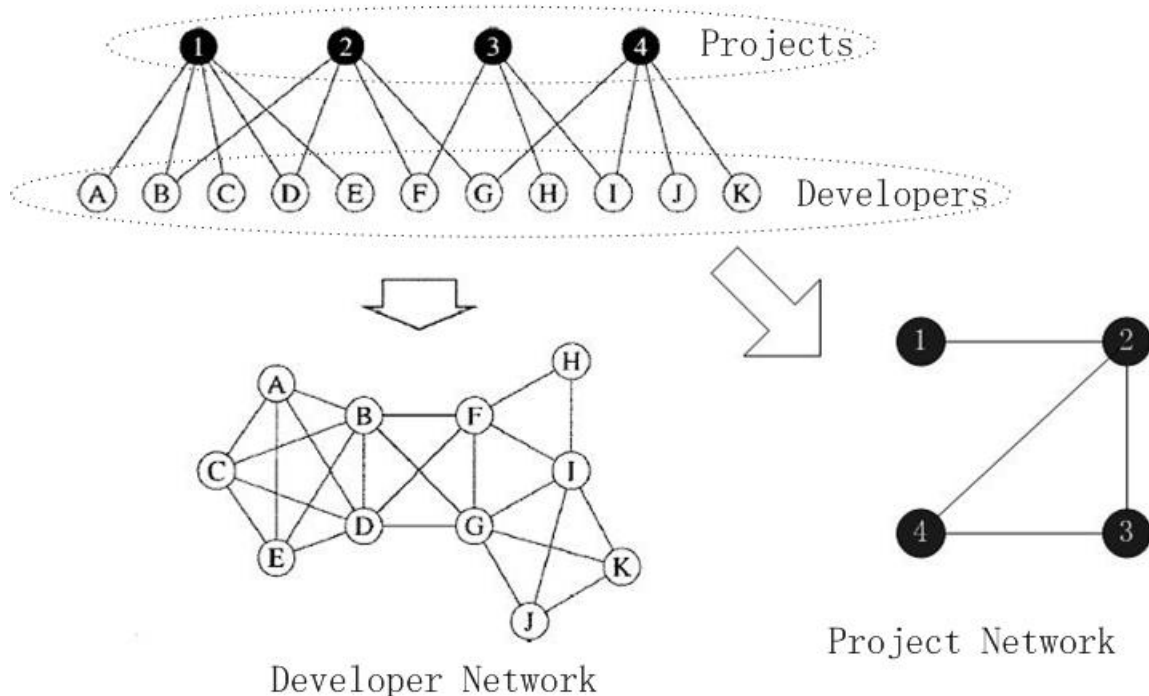


Figure 3.1. Transformation of the bipartite graph (from Newman, Strogatz and Watts 2001)

one project. The transformation of the bipartite network to a developer network is shown in Figure 3.1 [46]. Through the same method, we generated the project network, where a node represented a project in the collaboration network and the edges in the network represented the relationship of sharing the same developer(s).

Diameter is one of the important attributes in complex network research especially since the small world phenomenon¹ was popularized. The diameter of a network can be obtained by a modified breath first search of the whole network. The complexity of the algorithm is bounded by $O(MN)$, where N is the number of vertices in the graph and M is the number of edges in the graph (the pseudo code will be given later). But since most complex networks are often disconnected and

¹“Six degrees of separation” is a famous claim by Ouisa, a popular character in John Guare’s play (1990)

the actual sizes of these networks are huge and dynamic, e.g., (the Internet), it is not feasible for us to obtain the diameter by such a naive approach. Fortunately, we are still able to obtain the approximate diameter by statistical methods.

The theory of generalized random graphs is an approach proposed to summarize the BA model, WS model and ER model into a uniform definition. Newman and Watts proposed this theory by the idea of random graphs with arbitrary degree distributions [46]. They summarized all the complex networks into random graphs using the input from the parameters of degree distribution. For example, $BA = RandomGraph(PowerLaw)$ and $ER = RandomGraph(Normal)$. Detailed definitions and algorithms can be found in their paper. Based on this discovery, it is possible to obtain common network attributes for different networks from the same generalized model. Also in a generalized random graph, it is discovered that many properties described above, e.g., diameter, are obtainable in large networks. The crucial trick was working with the generating function rather than directly with the degree distribution p_k . The “generating function” [59] $G_0(x)$, is defined as

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k \quad (3.1)$$

This function encapsulates all the information in p_k , since every p_k can be generated by this function as

$$p_k = \frac{1}{k!} \left. \frac{dG_0(x)}{dx^k} \right|_{x=0} \quad (3.2)$$

The average degree z_1 of vertices in the network is given in terms of a derivative of G_0 :

$$z_1 = \langle k \rangle = \sum_k k p_k = G_0'(1) \quad (3.3)$$

Newman and Watts also discovered that the average number z_m of vertices a distance m steps away from a given vertex is given recursively by [46]

$$z_m = \frac{G_0''(1)}{G_0'(1)} z_{m-1} \quad (3.4)$$

and hence that

$$z_m = \left[\frac{z_2}{z_1} \right]^{m-1} z_1 \quad (3.5)$$

where z_1 is the average degree of a vertex as in Equation 3.3 and z_2 is the average number of vertices that can be reached in two steps from a given vertex. Thus, by knowing these two numbers for a network, we were able to predict the average number of vertices any distance away from a given vertex.

To calculate the diameter of a network, from any given vertex, we noticed that z_D , the number of vertices we can reach in less than D steps from the given vertex, is equal to the total number of vertices in the whole network. Then D is roughly equal to the average distance between all pairs of vertices, yielding the diameter of the network. Thus, by substituting $m \rightarrow D$ and $z_D \rightarrow N$ in Equation 3.5 and rearranging, we got

$$D = \frac{\log(N/z_1)}{\log(z_2/z_1)} + 1 \quad (3.6)$$

A more detailed derivation of equations (3.1 - 3.6) is presented in Appendix C or in [46]. We were able to calculate the approximate diameter of a network by z_1 and z_2 , which are easily obtainable attributes of the network.

In order to verify the correctness of this method, we also used the naive breadth-first search method to obtain the “real” diameter value of the network. The algorithm is given as follows:

- Pseudo-code

FOR every vertex v in the graph

```

mark every vertex in the graph as  $\infty$  and mark vertex  $v$  to be 0
set  $d$  to be 0
DO
  FOR every vertex  $i$  marked as  $d$ 
    following the attached edges of  $i$ , find all the vertices which are marked
    as  $\infty$  at the other side of the edges and store in  $lset$ ,
    FOR every  $l$  in the  $lset$ 
      mark  $l$  as  $d + 1$ 
    END FOR
  END FOR
   $d = d + 1$ 
WHILE no vertex is marked as 0 except  $v$ 
END FOR

```

In this algorithm, the “do-while” loop will calculate all the shortest paths from a given vertex to any other vertex in the graph. This is Dijkstra’s single source shortest path algorithm, which is proved to have time complexity as $O(M + N \log N)$ using best implementation, where M is the number of edges and N is the number of vertices. Thus the overall complexity of the algorithm is $O(NM + N^2 \log N)$. We will discuss the comparison between the results of the approximate method and the breath-first search in the next section.

We used this method to calculate the diameters of the monthly developer networks generated by the empirical data showing the network diameters were between 6 and 8, while the numbers of developers in the developer networks ranged from 30,000 to 70,000. Thus the diameter of the network is quite small compared to the overall network size (the number of developers in the network). As described

in the last chapter, for the ER model, the diameter was large and monotonically increased with the network size, whereas for the WS model and the BA model, they were much smaller. The diameter of the SourceForge developer network was much smaller than the network size.

Due to the bipartite property of the collaboration network, we also investigated the diameters of the monthly networks of projects, which is called “project network.” Investigation of project networks with similar statistics can reveal the relationship of the developer network and project network. In the project network, the average degrees are between 3 and 4, which is half of the degree of the developer network. In the developer network, the diameter ranged between 6 and 7 and the number of projects is from 30,000 to 60,000. Small diameter also exists in the project network and the actual value is similar to that of the developer network, especially when the network size is large.

The naive algorithm to get the clustering coefficient of a network requires three nested scans of the whole network. The complexity of the algorithm is high for networks of such a huge size. However, clustering coefficients of some real networks like the network we studied in SourceForge, can be calculated more easily from bipartite graphs [46]. Albert and Barabási have generalized the generating function method for bipartite graphs. The following are their processes and results for the clustering coefficient of bipartite graphs. Two generating functions are defined, one for each of the two degree distributions. If we denote the probability that a developer joins j projects by p_j and the probability that a project has k developers by q_k , then the two functions are

$$f_0(x) = \sum_j p_j x^j, \quad g_0(x) = \sum_k q_k x^k \quad (3.7)$$

From these, we can define a function

$$G_0(x) = f_0(g'_0(x)/g'_0(1)) \quad (3.8)$$

which is the generating function for the number of neighbors of a developer in the unipartite projection of the collaboration network in Figure 3.1. This function plays exactly the same role as the generating function in the diameter calculation. For example, the average number of co-developers of a vertex in the network is $z_1 = G'_0(1)$. By this method, the clustering coefficients of these kind of bipartite structures result in a non-vanishing value,

$$C = \frac{1}{1 + \frac{(\mu_2 - \mu_1)(\nu_2 - \nu_1)^2}{\mu_1 \nu_1 (2n u_1 - 3n u_2 + n u_3)}} \quad (3.9)$$

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

The result of Equation 3.9 has been tested in several collaboration networks [46]. In some cases the theoretical results (calculated through equation 3.9) correlated with the experimental results. In some other cases, the theoretical results were only half of the experimental results, and still in others the results deviated by a factor of 2 from the clustering coefficient of the real network. We also verified the approximate clustering coefficient method by measuring values that we had calculated directly from the network using the naive algorithm. The algorithm we used to calculate the clustering coefficient is given as follows:

- Pseudo-code

FOR each vertex v in the graph

```

set conn as 0
collect all the neighbors of v and set the pconn as  $\frac{v(v-1)}{2}$ 
FOR every possible pair of neighbors for v
    IF there is edges between these pair, increase conn by 1
END FOR
 $CC[v] = \frac{conn}{pconn}$ 
END FOR
gCC = average(CC)

```

For every given vertex, we checked the connections of all its neighbors. So the complexity is $O(z^2)$ for every vertex, where z is the average degree of the network. More precisely, the complexity should be $O(C_2^z)$. The algorithm will iterate all the vertices in the graph, so the overall complexity of the algorithm is $O(N \times C_2^z)$. We used this method to obtain the measured value of the clustering coefficients. The result is similar to the approximate value, where the average difference is just 0.05 when the measured values are between 0.7 and 0.75.

We use this method of calculating the clustering coefficient of the empirical data we collected from SourceForge. The results were between 0.7 and 0.75 for all developer networks that were developed in an 18-month period. From the material of last chapter, we know that the ER model always has a small clustering coefficient and it will diminish to zero as the system size grows. The clustering coefficient of the WS model is relatively larger, and irrelevant to the system size. The BA model also has a large clustering coefficient. The clustering coefficient we get from the SourceForge developer network is significantly larger than that of the ER model. Henceforth we conclude that the developer network we were investigating was not a type of ER model but could be a WS model, BA model or some other more advanced

model based on the highly clustered property we observed.

We also used a similar method to calculate the clustering coefficient of the project network. The results were between 0.5 and 0.6, which is smaller than the developer network.

Finally, we investigated the degree distribution of the SourceForge developer network. Degree distribution is proven to have a normal distribution in the ER model when $N \rightarrow \infty$. This was believed to be a good model for the real complex network before the power law was reported for many real network systems by Barabási et. al. In the SourceForge developer network, we found that the degree distribution also followed a power law, as shown in Figure 3.2. This graph is based on empirical data collected from SourceForge in January 2003. The X coordinate is the number of projects in which each developer participated, and the Y coordinate is the number of developers in the related categories. In the left figure, we can see a distribution that appears to follow a power law. To verify this, we applied a log-log transformation on the data, so that we could see if the distribution fit a straight line with R^2 as 0.9536 in the right figure. Also we found the similar power law degree distribution for the project network, which is shown in Figure 3.3. These power law degree distributions exist in all the monthly empirical data we collected from SourceForge. Again, from the information in the last chapter, degree distribution for the WS model is also a Poisson distribution, which in the limit becomes Gaussian, with the mean at K , the degree of original lattice in WS model. Therefore, the WS model also does not fit the SourceForge collaboration network. It can only be a BA model or some other more advanced model.

From the investigation of these attributes, we eliminated the ER model and the WS model. The BA model fit the empirical data better than the ER or WR model

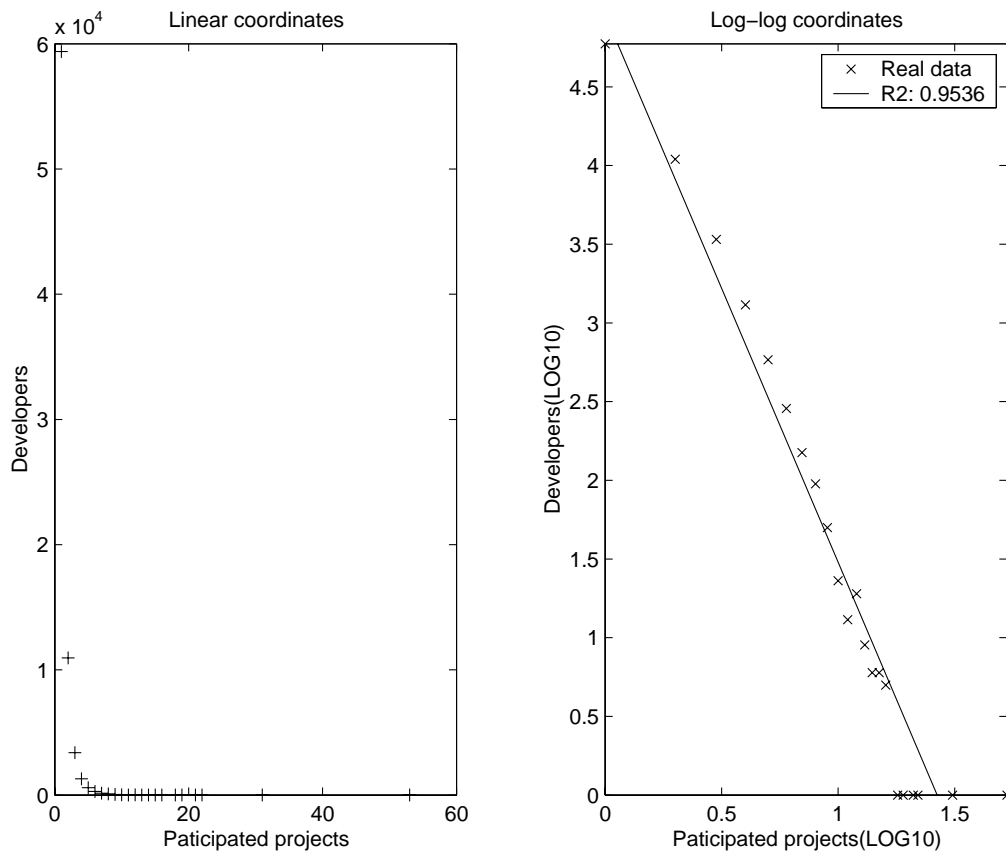


Figure 3.2. Degree distribution of SourceForge developer network on January 2003

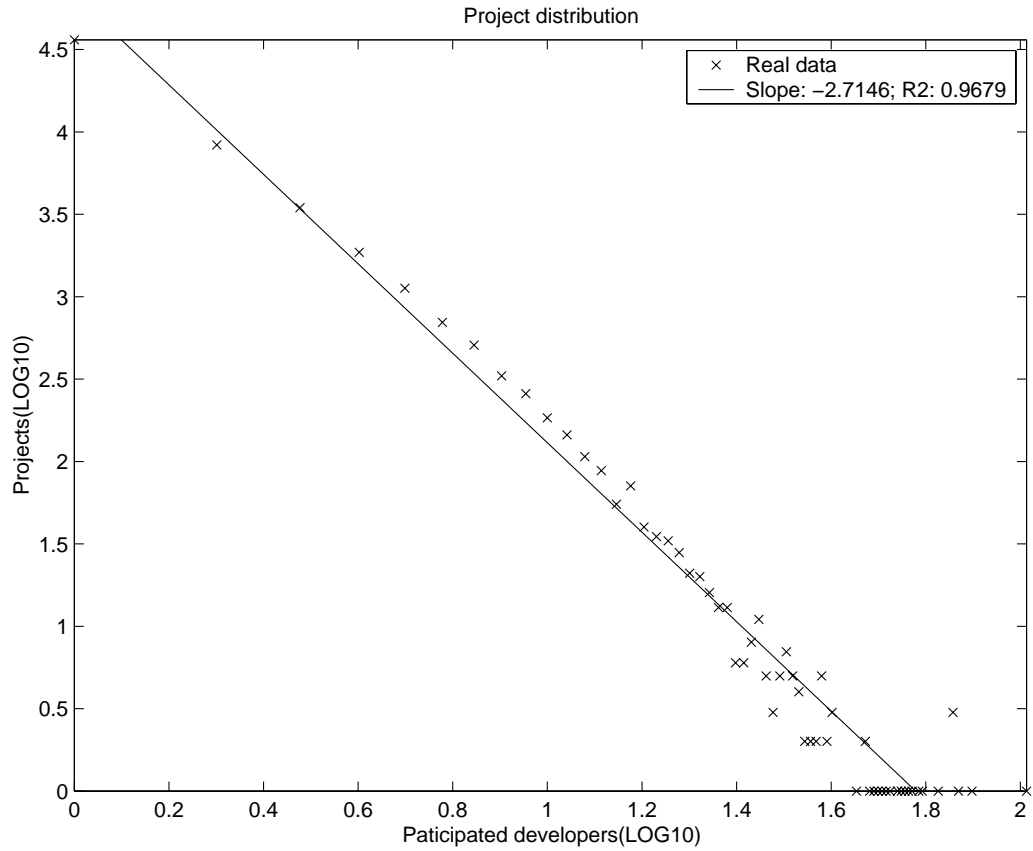


Figure 3.3. Degree distribution of SourceForge project network on January 2003

and it can also explain all the topological properties of the SourceForge developer network.

3.2 Evolution statistics

All the attributes in the previous section (diameter, clustering coefficient and degree distribution) are about the topology of the developer network. They depict the characteristics of a static network at a given point of time. The BA model is a good fit based on these attributes. But these are not the only important attributes in a network, especially an evolving network. Since we have collected data from SourceForge for over two years, we were able to investigate the development patterns of these attributes of this developer network. During this study, we found that the BA model is not the exact model for the SourceForge developer network.

3.2.1 Average degree is increasing

Average degree $\langle k \rangle$, which gives the average number of links per developer is a good quantitative measurement for the connectivity of a graph. The developing pattern of $\langle k \rangle$ in the developer network is shown in Figure 3.4, indicating an approximately linear increase of $\langle k \rangle$ with time. The X coordinate in the figure is the number of months that passed from the beginning of the SourceForge data collection. The increase of $\langle k \rangle$ may come from two changing parts in the network. First, the number of nodes in the network increased with time due to the arrival of new developers. Second, the total number of edges also increased through the collaborations made by new developers with old ones and by new collaborations made by old developers. These two changes act together to increase the $\langle k \rangle$.

In the figure, we can see the globally increasing trend of $\langle k \rangle$. This is an important feature to characterize a network in evolution. Using linear regression, the function that fits the data is $y = 0.0753x + 7.1958$ with R^2 is 0.8987. The actual

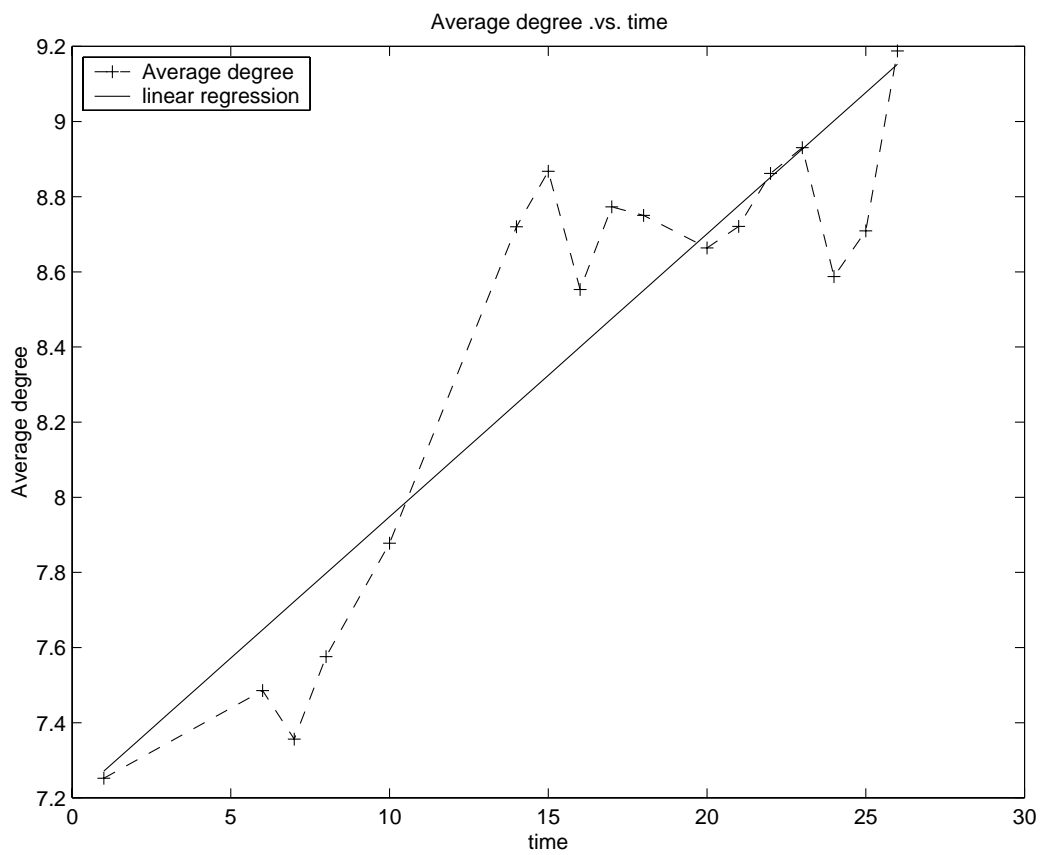


Figure 3.4. Average degree of SourceForge (developer network): unit of X coordinate is month, 0 represents December 2000

data points are distributed above or below the linear regression line. This deviation could be caused by significant network topology change. We will investigate this phenomena later in this chapter.

We also investigated the $\langle k \rangle$ for the project network. The values of $\langle k \rangle$ for project networks are around 3, which are smaller than those of the collaboration network. But they have the similar slopes of linear regression as in Figure A.1.

3.2.2 Cluster distribution also follows the power law

A cluster is defined as the maximal subset of connected nodes (a path exists between any pair of nodes). It is important to realize that the developer network is composed of many clusters. There are two reasons for this. First, there are developers that do not collaborate at all, i.e., each of them is the only developer of a certain project. We defined these as isolated clusters, and the percentage of isolated clusters in the whole network is not negligible (about 30%). Second, the links between developers are connected by projects. As long as a project is still active, developers may still collaborate with more newcomers to the project. But the life of a project is finite (about 65% of the new projects in July 2001 died in less than 9 months), so the collaborating relationship between developers changed with time.

Since the cluster is an important property of a developer network, investigation of the cluster is helpful in our understanding of the network. In the SourceForge developer network, a power law also exists in the cluster distribution, as shown in Figure 3.5. In the second figure, in which we applied *log* transformation on both coordinates, we found that the cluster distribution fits a straight line quite well without considering the biggest cluster (which will be called the major cluster in latter discussion). The R^2 of linear regression with major cluster is 0.7426 and the

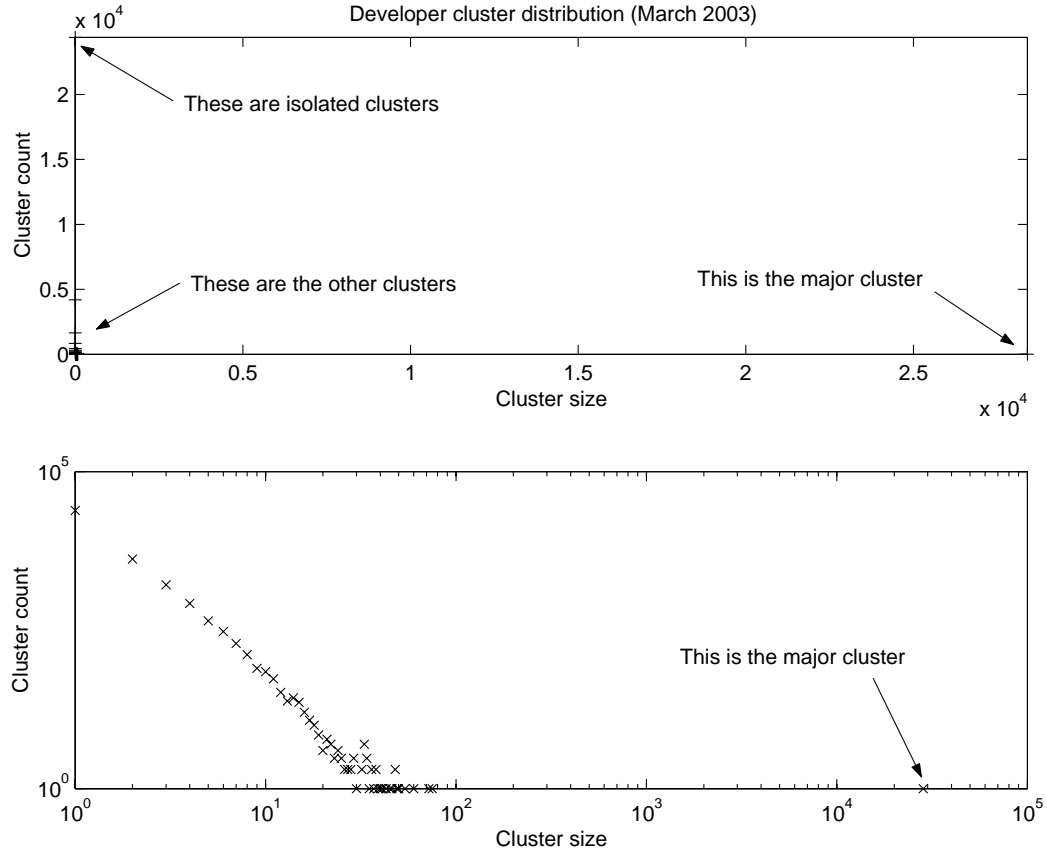


Figure 3.5. Developer cluster distribution (developer network): the upper figure is in linear coordinates and the lower one is in log-log coordinates.

R^2 of linear regression without major cluster is 0.9799. In the first figure, where the coordinates are in linear scale, we made another interesting discovery: almost all the clusters are quite near to each other except two extremes. One is the major cluster and the other is the isolated cluster. Isolated clusters in the graph were generated because of the isolated developers we mentioned in the beginning of this subsection.

Studying the emergence of major clusters may help us understand the evolution of the developer network. To do so, we measured the relative size of the major cluster, r , which is the ratio of the number of nodes in the major cluster relative to the total number of nodes in the developer network. Results are shown in Figure 3.6.

The phenomenon of the continuous increase in r with the evolution of time is called percolation [17]. The basic idea of percolation is the existence of a sharp transition at which the overall connectivity of the network appears. This transition occurs abruptly when the density of network edges reach a critical value, which is called the percolation threshold. The major cluster is also called the “giant component” in random network theory [15]. However, the process leading to the major cluster in SourceForge developer network is fundamentally different from these extensively studied phenomena. In the developer network, the network was composed of clusters from the very beginning, since the developers with similar backgrounds collaborated from the very beginning instead of being totally isolated as assumed in the previous research. However, in the figure we found that an increasing trend existed in the evolution of the relative size of the major cluster and isolated clusters. The relative size of the major cluster increased quickly in the beginning and slowly converged to approximately 35%, indicating that the developer network is going to reach some stationary topology. Meanwhile, the relative size of the isolated clusters, which is mainly caused by the new developers, is constant at around 30%, which also indicated that the developer network is going to reach a stationary topology. This is more interesting, given that the number of developers and projects has more than doubled in the same period.

3.2.3 Diameter and clustering coefficient decays with time

The diameter of the developer network is a good measure of network communication ability. A shorter diameter results in fewer average steps needed for one developer to spread a message to another developer and less time needed for an idea to spread through the network. The developer network has a small diameter, which was calculated in a previous section. Also we investigated the evolution of

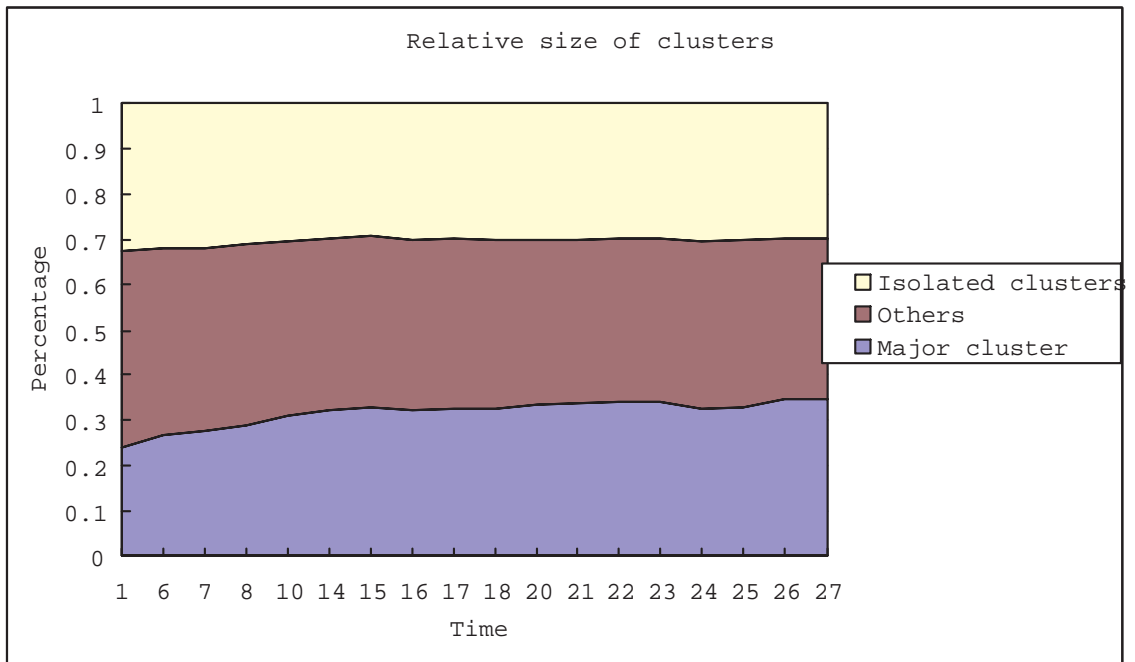


Figure 3.6. Relative size of major cluster (developer network): unit of X coordinate is month and 1 represents January 2001

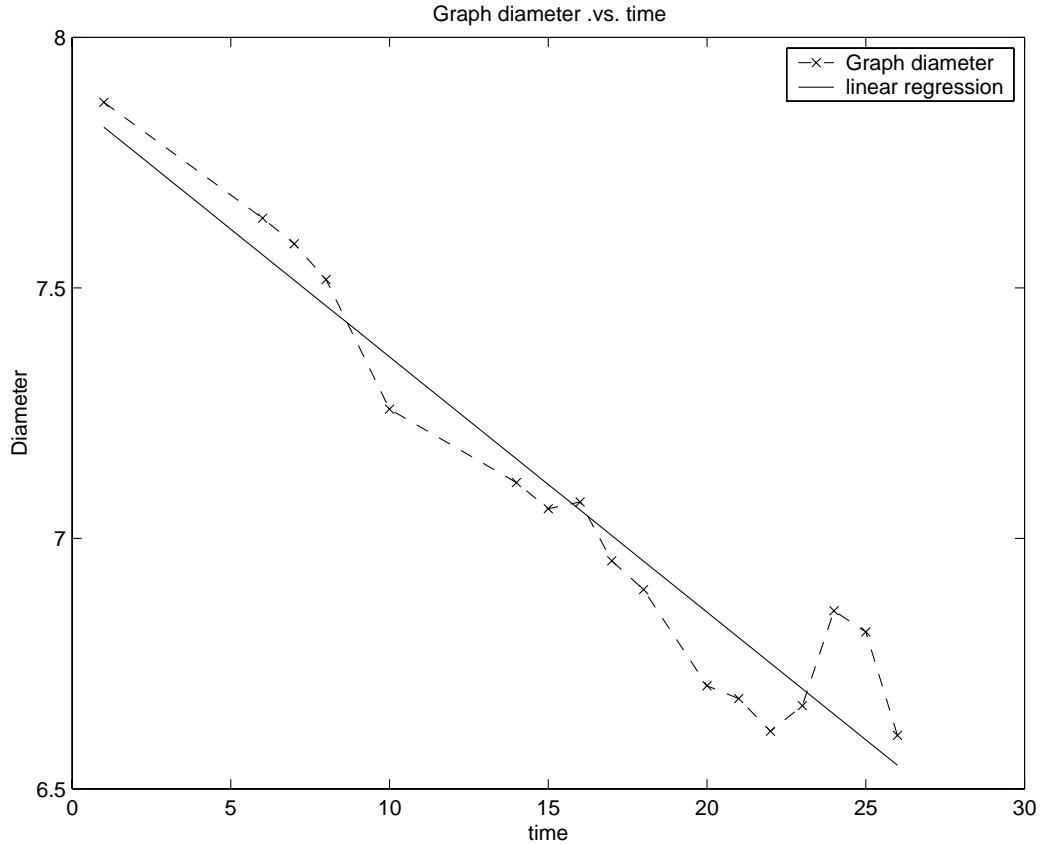


Figure 3.7. Approximate diameter of the developer network decays with time (function of linear regression is $y = -0.0510x + 7.8717$ and the R_2 is 0.9619):unit of X coordinate is month and 0 represents December 2000

the diameter of the network, as shown in Figure 3.7.

The figure indicates that D decreases with time, which is different from the previous research [15] on random networks that reports that diameter increases with network size. This decreasing trend could have three reasons. First, preferential attachment will cause a significant increase in the degree of hubs in the network (the vertices with more links than average in the network), thus decreasing the diameter. Second, the increasing arbitrary size of a major cluster means the connected component may weight more in the overall system, thus increasing the overall connectivity

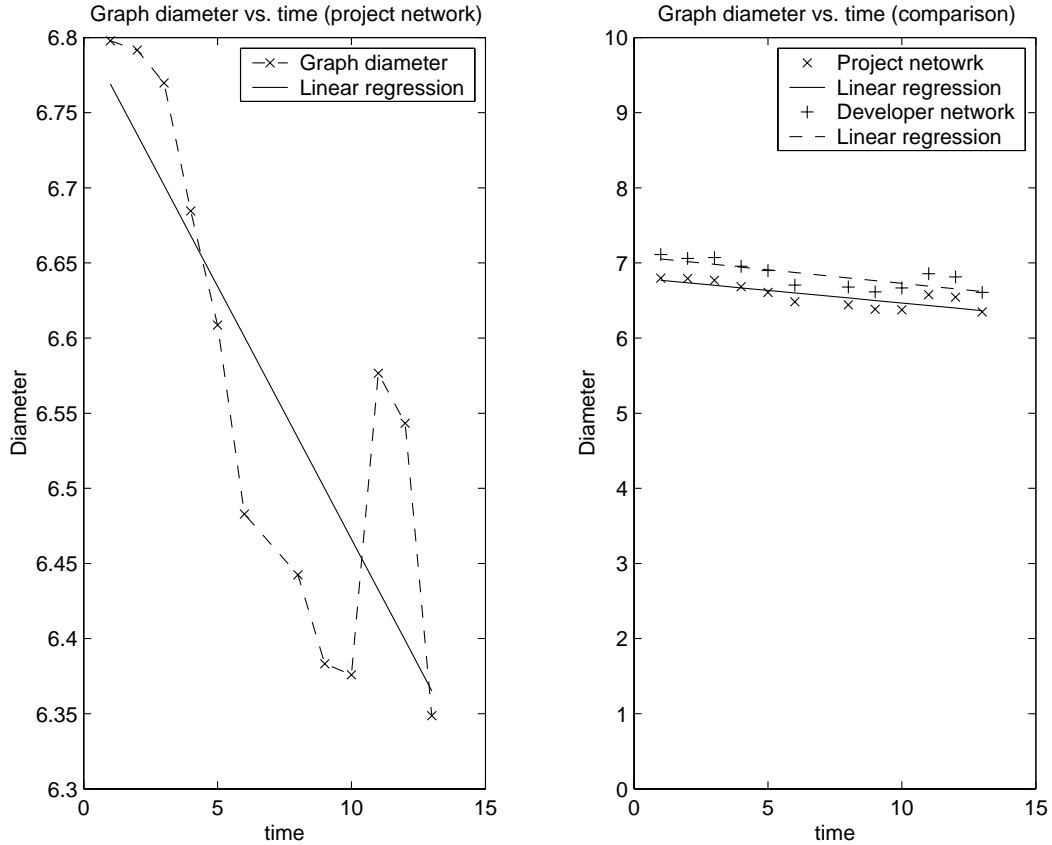


Figure 3.8. Approximate diameter of project network: unit of X coordinate is month and 0 represents January 2001

of the network. Finally, the diameter decreases as internal links (for example, old developers joining projects that previously existed) increase the interconnectivity of the network, thus decreasing the diameter.

Furthermore, we compared the developing patterns of diameters in both the collaboration network and the project network. The result is shown in Figure 3.8. The developing patterns of diameters in these two networks are almost identical and the difference between the actual values diminishes over time.

To verify the approximate method of calculating the diameter, we used the algorithm discussed in the last section in order to obtain the actual diameters and

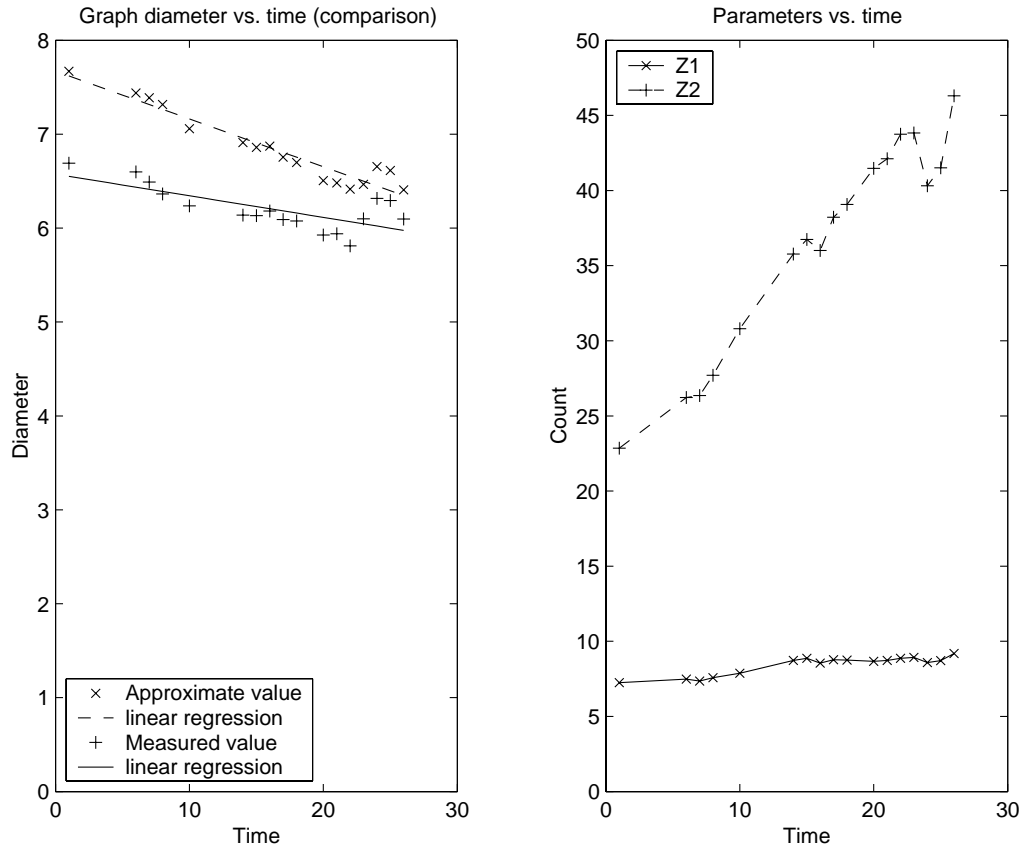


Figure 3.9. Approximate diameter vs. measured diameter (major cluster in the developer network): unit of X coordinate is month and 0 represents December 2000

compare them with the approximate values. Since the algorithm requires a connected network (paths exist for any given pair of vertices) and we only needed to verify the connectedness of the approximate method, the diameters computed by the approximate method and simple method for the major cluster are shown in Figure 3.9.

The left figure is the comparison of the approximate diameters and the measured diameters. The right figure displays the related parameters, z_1 and z_2 . The approximate values are larger than the measured values for the major cluster (largest connected component), but the difference is decreasing over time. Also we found the difference between z_1 and z_2 is increasing at the same time. This means that the difference between the approximate value and the measured value is decreasing while the difference of z_1 and z_2 is increasing. This observation is the same as that by Barabási [6] stating that the approximate value is correct when $z_2 \gg z_1$.

Clustering is another important characteristic of the topology of real networks. So the clustering coefficient, a quantitative measure of clustering, CC , is also an attribute we investigated. The approximate clustering coefficient for the developer network as a function of time is shown in Figure 3.10.

In the figure, we can also observe the decaying trend of the clustering coefficient. The clustering coefficient for a developer network tells us how much a node's co-developers are willing to collaborate with each other, and it represents the probability that two of its developers are collaborating on a project together. Thus with the evolution of the developer network, more edges (collaboration relations) are formed. This will lead to a decrease in the probability that two co-developers will join a new project together due to the fact that they are participating on a number of projects approaching their limits. Furthermore, this leads to the decrease in the clustering coefficient.

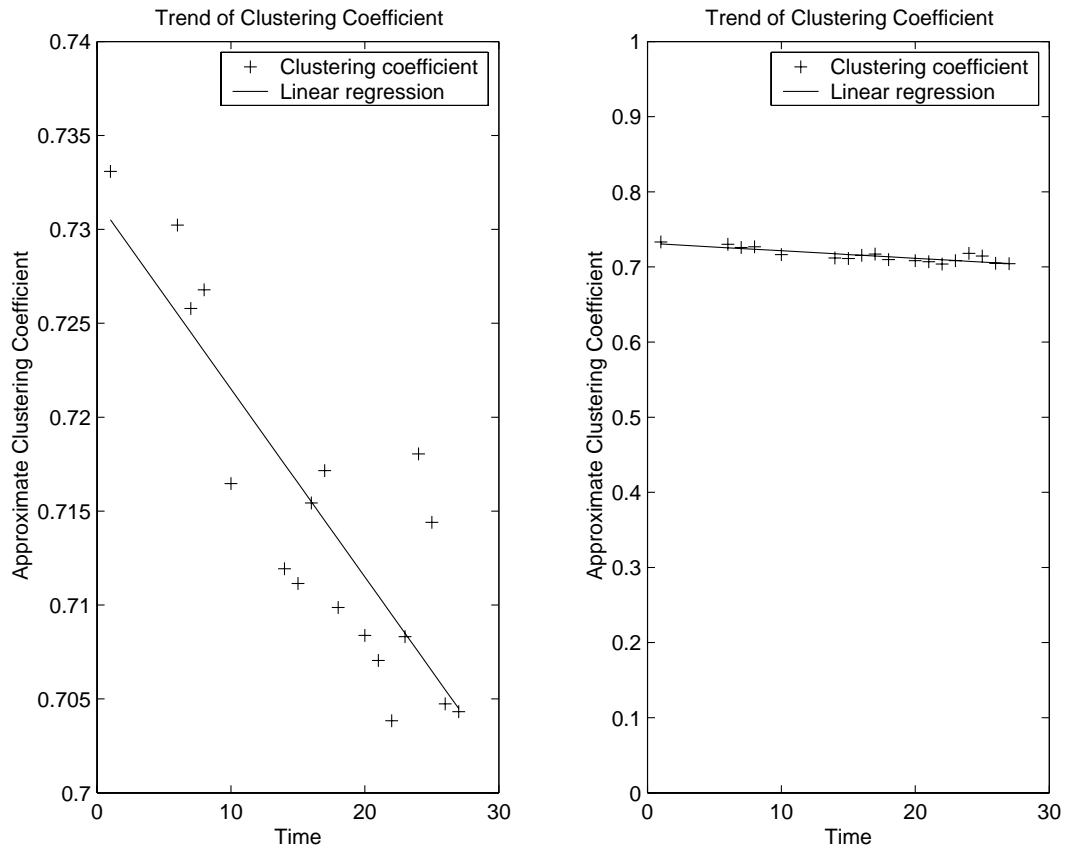


Figure 3.10. Approximate clustering coefficient (developer network) decays with time: unit of X coordinate is month and 0 represents December 2000

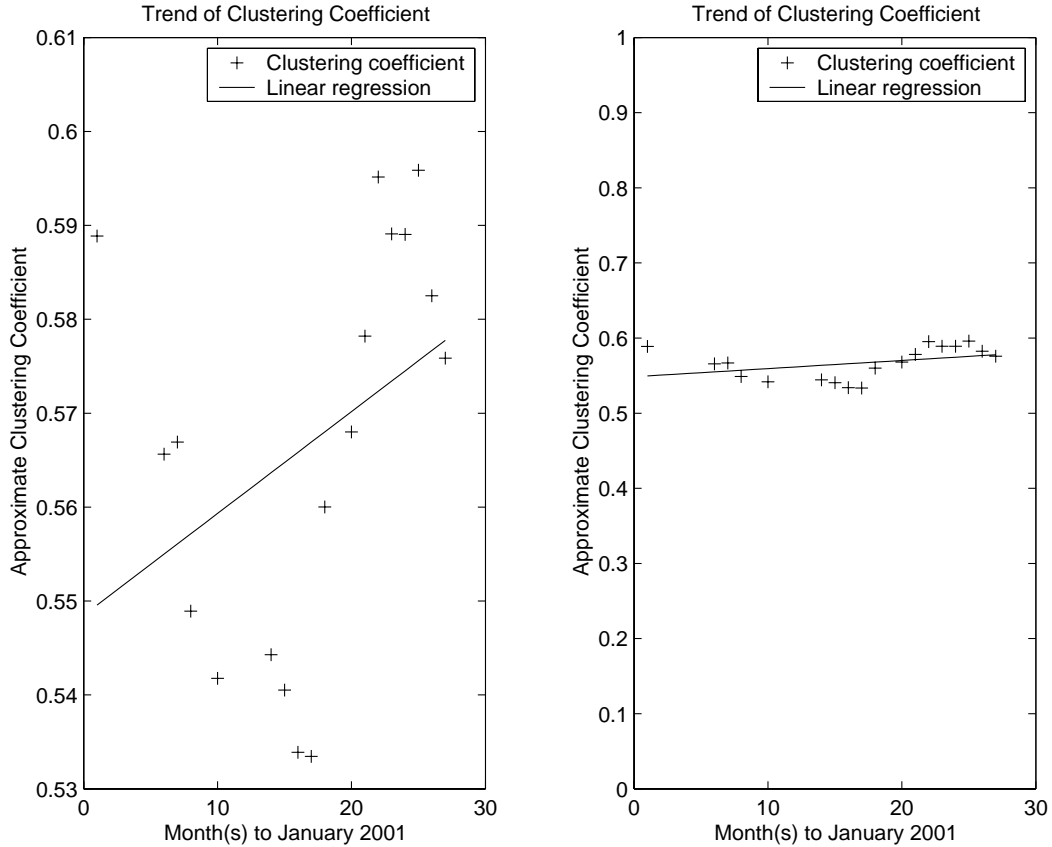


Figure 3.11. Approximate clustering coefficient in project network: unit of X coordinate is month and 0 represents December 2000

We also included the linear regression (the function is $y = -0.0010x + 0.7315$ and R^2 is 0.8531) of the clustering coefficient developing trend in the figure. The deviations to the linear regression at certain data points are more varied and some of these data points (each data point represents a certain monthly developer network) also have relatively large deviations in Figure 3.4 and Figure 3.7. We will investigate this phenomenon later in this section.

The approximate clustering coefficient in the project network, shown in Figure 3.11, is different from the clustering coefficient in developer networks. The data points are more scattered. Comparing these two figures (3.10 and 3.11), we found

that CC of the project network is much more noisy compared with CC of the developer network. This may be because of the small average degree of the project network. Thus the developer network has more accurate approximate value than the project network in statistical analysis, at least in regards to clustering coefficients.

If we look at Figures 3.4 and 3.7 together, we find there are some data points with significantly larger deviation than others (e.g. the 7th, 15th and 24th month, which represents July 2001, March 2002 and December 2002 respectively). We investigated December 2002 data to understand this kind of phenomena. After studying the statistics of the related months, we found that the total number of developers increased from 69,306 to 73,488 from October 2002 to December 2002, while the normal increase of each previous month was always under 400. The total number of projects increased from 46,932 to 51,676 from October 2002 to December 2002, while normal increases during previous months were always below 800. This indicated that there is a big growth in size for SourceForge at the end of 2002. Also we checked the cluster distribution of the related months and found that the numbers of developers who participated in more than 10 projects were similar from October to December (from 61 to 57), but the number of developers who participated in just one project increased from 53,842 in October to 57,305 in December. Normal increases in the previous months were always under 200 for each month and the number of projects that had only one developer usually showed monthly increases fewer than 600. All these statistics pointed to one fact: there was a significant number of new developers, only participating in one project, who joined SourceForge during November and December 2002. These new developers led to the decrease of the average degree and the increase of diameter, which all match the observation from the statistics shown in Figures 3.4 and 3.7.

3.2.4 Multiple preferences

Before the appearance of the scale-free network model, randomness was always the dominant property in network models (single probability is the only parameter in a model). The discovery of the power law distribution in many real networks made us look for a new model for complex networks. This brought about the advent of the scale-free network model. In scale-free networks, we have “preferential attachments,” which means that nodes are more likely to link to the nodes which already have a larger number of links [5]. Preferential attachments are a part of all network models that aim to explain the emergence of the heterogeneous network structure and power law connectivity distribution. The availability of dynamic data on the SourceForge collaboration network evolution allowed us to investigate the presence of preferential attachment and its implication for the network topology and evolution. There are two kinds of preferences in the collaboration network.

1. Project preference. This is the preference associated with a developer opting to join an existing project. It can be further divided into two types. For a new developer, it is more likely that the first project will be the more popular project (the one with more developers) if he decides to join a project instead of creating a new project. This makes the projects with more developers have a higher probability of gaining new developers than the others. For an “old” developer, it is also likely that the project selection is based on preference (the popularity of the existing projects) if the developer decides to join an existing project. The quantitative proof of the existence of preference in the project attachment is given by Barabási [12]. We also collected the statistics for the SourceForge collaboration network to verify the existence of preferential attachment in project selection. Since the table is too extensive, we will not include it here, but it will be listed in Appendix A for further investigation

(Table A.1 and Table A.2).

2. Developer preference. This preference involves a developer deciding which action he will take. This preference is trivial for a new developer, because he has not participated in a project, which means that he has no preference yet. We believe that an old developer will decide to take an action on the basis of the projects in which he has already participated. This means the more projects the developer has, the higher the probability that he will choose to take action instead of idling. This is the developer's preference.

3.2.5 Projects have fitness and life cycle

Based on the SourceForge empirical data over two years, we were not only able to look at the topology of the network at a given time, but also the evolution of the network, and the developing patterns of a single entity in the network. The developing pattern of the project is one which we will investigate. The project's developing patterns are characteristics of the project during its entire lifetime (from "birth" to "death"). With this information, we were able to understand the attributes that influence the success or failure of a single project.

We investigated all the new projects beginning on July 2001, and traced their developments month by month until February 2003. Our data collection started from January 2001, but June 2001 and July 2001 were the first two adjacent months. There are 1660 brand-new projects for July 2001. We picked the most representative 10 projects for the purpose of display, which is shown in Figure 3.12. The statistics we discuss below are based on the whole collection of new projects (1660 projects).

In the figure, the X coordinate is the number of months from June 2001 and the Y coordinate represents the project size (which is the number of the developers in the project). The average monthly increase of these 1660 projects throughout the

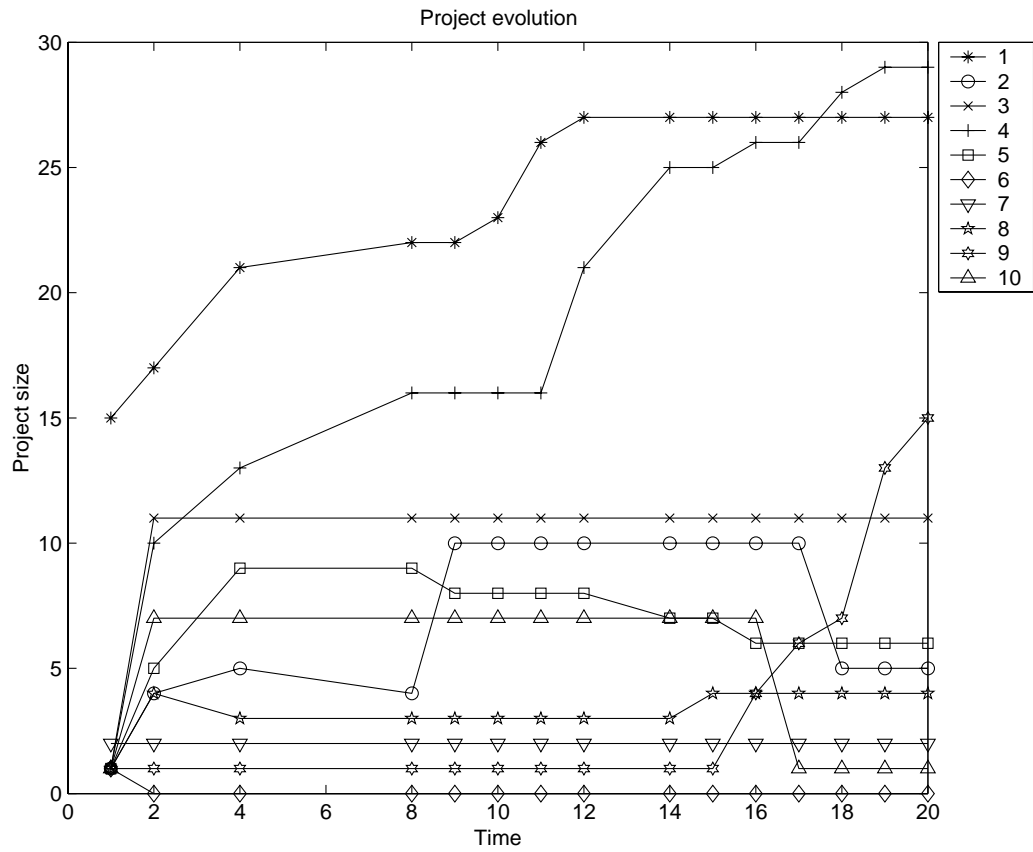


Figure 3.12. Representative projects' evolution with time, unit of X coordinate is month and 0 represents June 2001

sample period is 0.3639 (percentage of new developers amongst all developers in a project). The maximum monthly increase per project is 13, which is much higher than the average. This is an indication that a few projects increase more quickly than the others, which is good evidence for the existence of fitness. Fitness is the parameter that describes the abilities of different projects to attract new developers. The introduction of this parameter, found in the SourceForge data, explains the phenomena that young projects may outrun old ones in some circumstances, which is impossible in a simple BA model. But further research is still needed to find the detailed relation of fitness to the preferential attachment.

In the statistics (not only the selected projects in the figure), we found a common phenomenon for every project: they will increase rapidly in the beginning and then the increase will slowly decrease, which means the project has reached some stationary state. Some of the projects will begin to decrease after the stationary state and the others will still remain stationary until the end of our sample period. This may be an indication of a life cycle in the development of projects. We investigated the average of the monthly increase for all the 1660 samples. The change of the average is given in Figure 3.13. We can see the constantly decaying trend of average monthly growth in the figure.

In the previous discussion, we obtained many statistics from the empirical data. Also we compared the ER, WS and BA models with the empirical data based on these statistics.

3.3 Summary

In this chapter, we studied the statistics of the topology and evolution of these statistics for the empirical data. For the topological statistics, we calculated the degree distribution, cluster distribution and average degree of the network. Also we

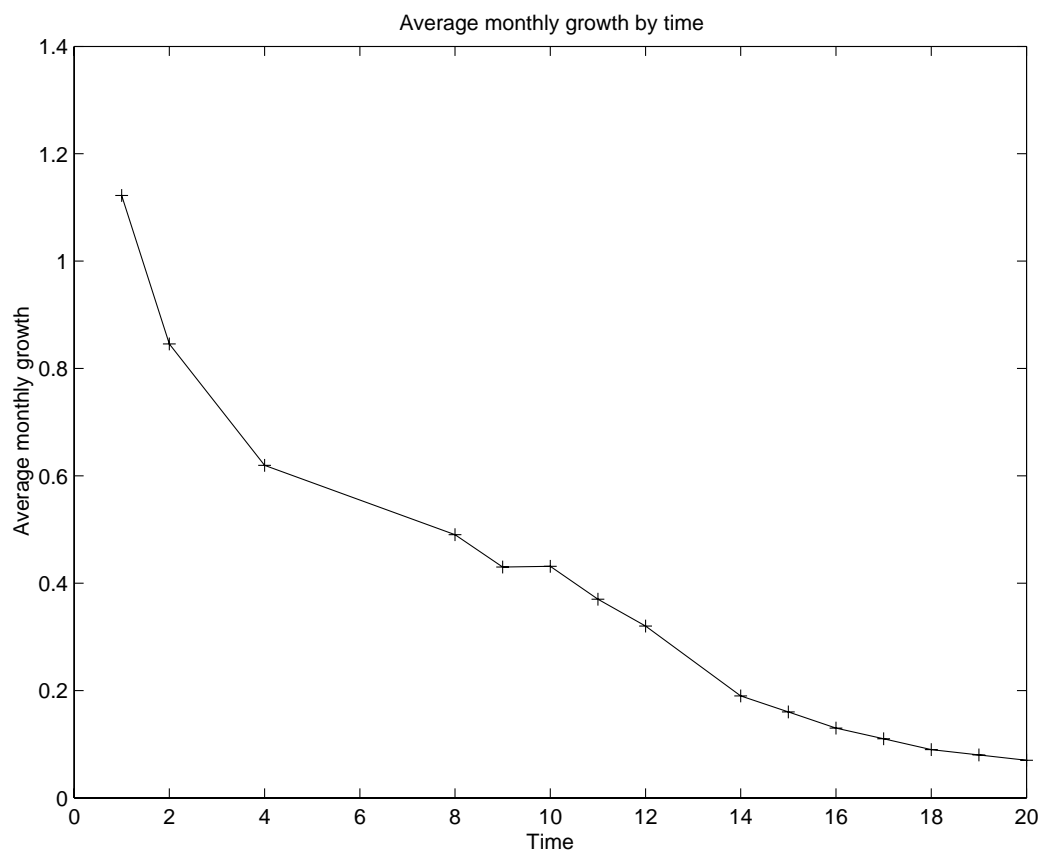


Figure 3.13. Average monthly growth (project size measured by developer count): unit of X coordinate is month and 0 represents June 2001

applied approximate methods to calculate the diameter and clustering coefficient of the network and verified the correctness of these approximate methods. These two approximate methods are

$$D = \frac{\log(N/z_1)}{\log(z_2/z_1)} + 1$$

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

For the evolutionary statistics, we investigated evolutions of all the topological statistics. Among these, we observed a decaying trend in diameters and clustering coefficients and we also observed the relationship between the evolution of the relative size of the major cluster and the evolution of the overall network. All the investigated attributes are summarized in Table 3.1. We will further investigate the SourceForge collaboration network in Chapter 4 by modeling and simulation.

Table 3.1. SUMMARY OF STATISTICS

Statistics	Developer Network	Project Network
Average degree	Yes	Yes
Diameter	Yes	Yes
Measured diameter	Yes	N/A
Clustering coefficient	Yes	Yes
Measured clustering coefficient	Yes	N/A
Degree distribution	Yes	Yes
Cluster distribution	Yes	N/A
Major cluster	Yes	N/A
Project behavior	Yes	N/A
Average monthly project growth	Yes	N/A

CHAPTER 4

SYSTEM MODELING AND SIMULATION

In chapter 3, we inspected and analyzed the statistics of the network topology, but as yet we have not captured all the features of a complex social network like SourceForge just by statistical analysis. So to complement the statistical analysis, we used modeling and simulation to help our understanding of the SourceForge collaboration network. The advantage of modeling and simulation is that they reproduce the network dynamics across time so that we can compare the results with the empirical data. First, we will define a model for simulation based on the statistics we calculated. Then we will use simulation to verify and validate the model.

We used an iterative framework for this simulation study and applied this framework in our research. We generated models for the adapted ER model, the BA model and the BA model with constant fitness. Then, we verified and validated these models with the empirical data. Finally, we proposed the BA model with dynamic fitness.

4.1 Experiment environment

We ran our simulations on a computer cluster, which included three simulation servers, three database servers and one gateway machine. These machines were 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 are 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 programming language we used is Java.

4.2 Framework for simulation study

The conceptual framework used in our research for agent-based modeling and simulation is shown in Figure 4.1. There are three entities in the framework: empirical data collection (limited real data), model and simulation. The model is the definition of procedures and related parameters, by which we could reproduce the evolution of empirical data in simulation. The simulation is an implementation of the model. We also had six relationships in the framework, each representing one kind of process: characterization, description, generation, adjustment, verification and validation. Characterization is a process of abstracting the characteristics of empirical data and generating the procedures and parameters in the model. Description manifests the underlying mechanisms of the evolution of the empirical data. Generation builds a simulation based on the given model. Adjustment modifies the model according to the feedback from the verification process. Verification is used to test 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 the simulation output with the empirical data by attributes not used to define the model. Validation may add more rules or attributes into the model for prospective improvement.

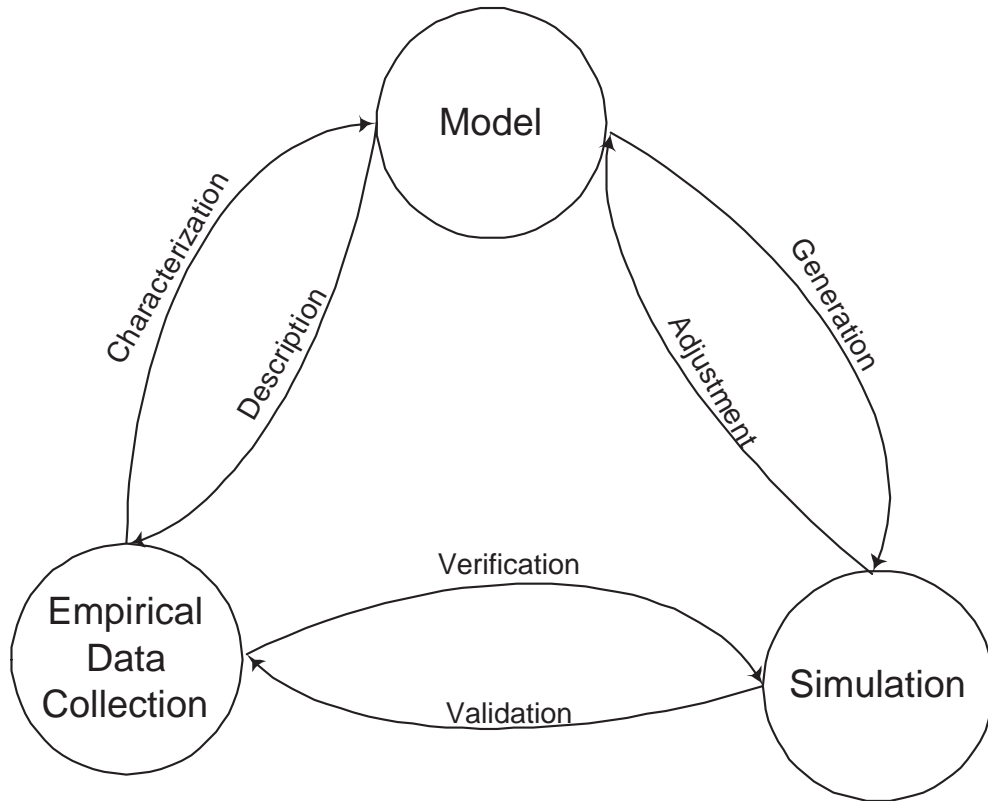


Figure 4.1. Conceptual framework for agent-based modeling and simulation

The main goal of our study is to get a “fit” model to describe the evolution of a collaborative network by simulation iterations.

4.3 Modeling

As discussed in Chapter 2, our collaboration network of SourceForge is different from many other types of collaboration network in regards to detachment (detachment means a developer quits a project). We made our model based on the bipartite graph (the collaboration network of SourceForge) so we would be able to describe the detachment more straightforwardly.

4.3.1 Model description

The model for simulation is like a procedure definition with related parameters. In our simulation, there were two kinds of entities – developer and project. Developers, which are the active roles in 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 is triggered to act 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 actual number of new developers at every time step was a random number generated by a uniform distribution averaged at $N(ND)$. Then every existing developer (existing and new developers) in the simulation was triggered to act one by one. The procedure definitions for new developer and existing developer were different, which we will discuss separately.

- *New developer.* Every new developer is triggered to act right after it is 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 exemption 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, joining, abandoning and idling. 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. Idling 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 idling. The probabilities that a developer would choose a certain action are different in different models for simulations based on different considered attributes.

This is just the skeleton of the procedure definitions in our models. Different simulation models may have different procedure 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 be different for different simulation iterations. We will explain these special procedure definitions and related parameters when we discuss that specific model.

4.4 Simulation models

According to this framework, 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 the “fit” model for the empirical data. Now we will discuss the details of our simulation models one by one.

4.4.1 Model one: random network (adapted ER model)

We started from the ER model since it is the first well-known complex network model and was the base of the others. This is the first model that employed nonlinear

dynamics and displays some extraordinary properties caused by phase transition. One of them is the existence of well-defined boundaries that separate order from disorder [32].

We started our simulations from an ER model. This was not only for the theoretical reasons but also because of the following practical reasons.

1. All the other models that we iterated could be generated from an ER model by simple modifications and additions, making this the most convenient way to accomplish all the simulations.
2. The ER model is the most well-studied model. Almost all the measures we use have been measured and mathematically proven. Starting with this model, we can easily verify the basic model infrastructure by comparing the simulation with theoretical results.

The network in this model is growing, which is different from the traditional ER model. So we call it an adapted ER model. But 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.

In detail, for developers, the randomness is the uncertainty in the 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 the previously stated 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 and every project has the same probability of being chosen. In our simulation, we maintained a project list of all existing projects and let developers randomly chose which to

join.

According to the model description in last section, we had six simulation parameters which were 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 used 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 was 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 were as follows: $P(CO) = 0.018028$, $P(JO) = 0.005974$, $P(AO) = 0.0076881$ and $P(IO) = 0.96831$. They were calculated from the empirical data which is shown in Table 4.1.

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 4.2. The clustering coefficient of the 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 as the overall system size increased. The relation between clustering coefficient CC and system size N (between 21,200 and 35,320) is approximately

$$CC \sim N^{-0.14} \quad (4.1)$$

This observation is identical to the mathematical result for the clustering coefficient

Table 4.1. PARAMETERS FOR EMPIRICAL DATA

	Create/New	Join/New	Create/New	Join/Old	Abandon/Old	Idle/Old
Feb-02						
Mar-02	0.7372673	0.2627327	0.015049	0.007169	0.0093427	0.9684398
Apr-02	0.73761865	0.2623815	0.014952	0.004479	0.008067	0.9725016
May-02	0.71132597	0.2886740	0.018696	0.00594	0.0064923	0.9688714
Jun-02	0.70125504	0.29874496	0.022372	0.007518	0.0156165	0.9544943
Jul-02	0.73587082	0.26412918	0.035157	0.009078	0.0095542	0.9462106
Aug-02	0.73615917	0.26384083	0.035157	0.009078	0.0095882	0.9461766
Sep-02	0.59382819	0.40617181	0.00431	0.002772	0.0014315	0.991487
Oct-02	0.65029326	0.34970674	0.010138	0.004686	0.0033642	0.9818124
Nov-02	0.67282609	0.31717391	0.01665	0.007155	0.0059622	0.9702328
Dec-02	0.51726519	0.48273481	0.004997	0.004081	0.0172249	0.9736977
Jan-03	0.64162404	0.35837596	0.01143	0.00577	0.0034453	0.9793555
Feb-03	0.7084419	0.2815581	0.024452	0.005154	0.0038227	0.9665713
Average	0.67331893	0.32668107	0.018028	0.005974	0.0076881	0.9683101
STDEV	0.06903709	0.06903709	0.010595	0.002037	0.005062	0.0143676

of the ER model [6], which proved $CC \sim N^{-\beta}$, where β is a constant smaller than 1.

From the previous chapter, we know that the clustering coefficient of SourceForge is relatively high at around 0.7 and decreasing. So the CC result of ER simulation is different from the empirical data.

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 4.3. The average degree of the simulated data is decreasing while the diameter of the simulated data is increasing ('x' in Figure 4.3). In theoretical ER models, the average degree should decrease while the size of the network increases and the mathematical expression for the diameter of an ER model [6] is like

$$D \sim \frac{\log(N)}{\log(\langle k \rangle)}, \quad (4.2)$$

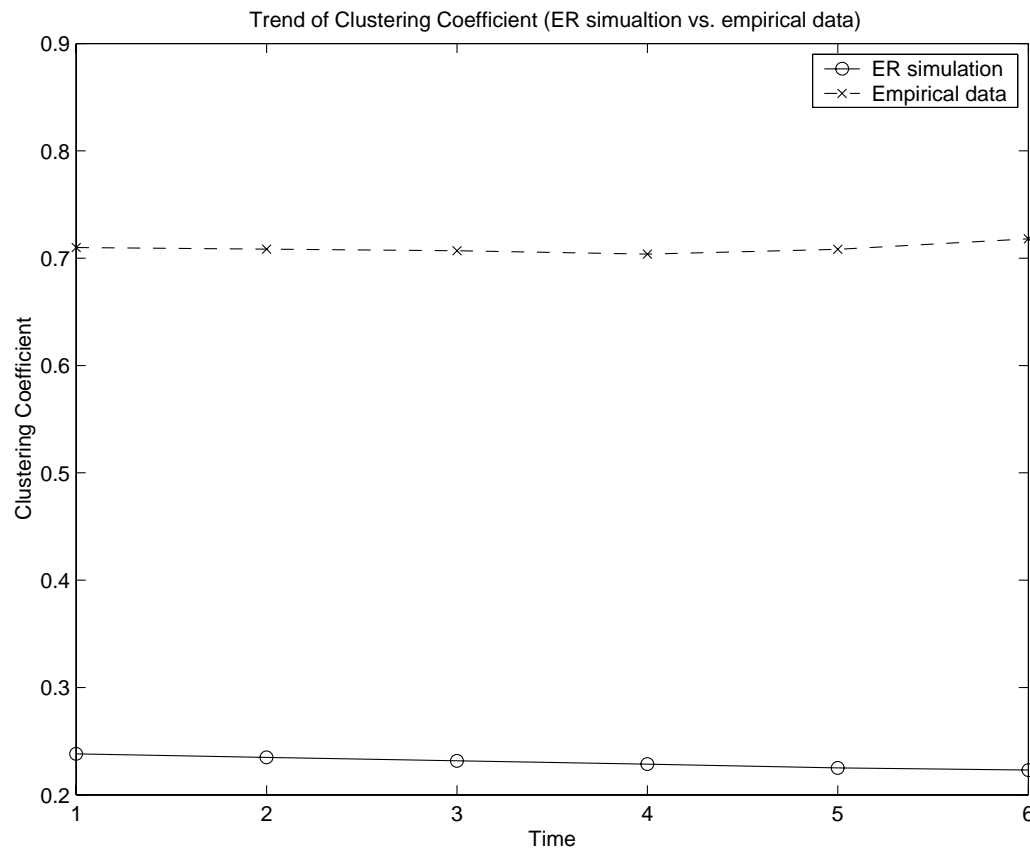


Figure 4.2. Clustering coefficient in ER: unit for X coordinate is month, and 1 represent July 2002

so the average degree and diameter of simulated data qualitatively agree with the mathematical results.

From the previous chapter, we know 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 4.3). The arbitrary value of the average degree of SourceForge (which is bigger than 7) is bigger than the average degree of an ER model (which is less than 6) while the value of diameter of SourceForge (which is bigger 6) is also bigger than the diameter of the ER model (which is less than 3). From the experimental data on the diameter, it is clear that the ER model does not fit the SourceForge collaboration network well.

Then we studied the cluster distribution. The cluster distribution of the simulated ER model is shown in Figure 4.4. The upper figure represents the cluster distribution with linear coordinates. There is one big cluster present, which is proven to appear in ER models at some time during evolution. The process of generating this cluster is called percolation, which we discussed in Chapter 2. The lower figure is the cluster distribution after log-log transformation. 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 cluster. 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 similar slopes of linear regressions to the empirical data in regards to cluster distribution and data points. We will not discuss this in the coming models, however, the related simulation results are included in Appendix A.

Finally, we investigated the degree distribution of ER model. The simulation

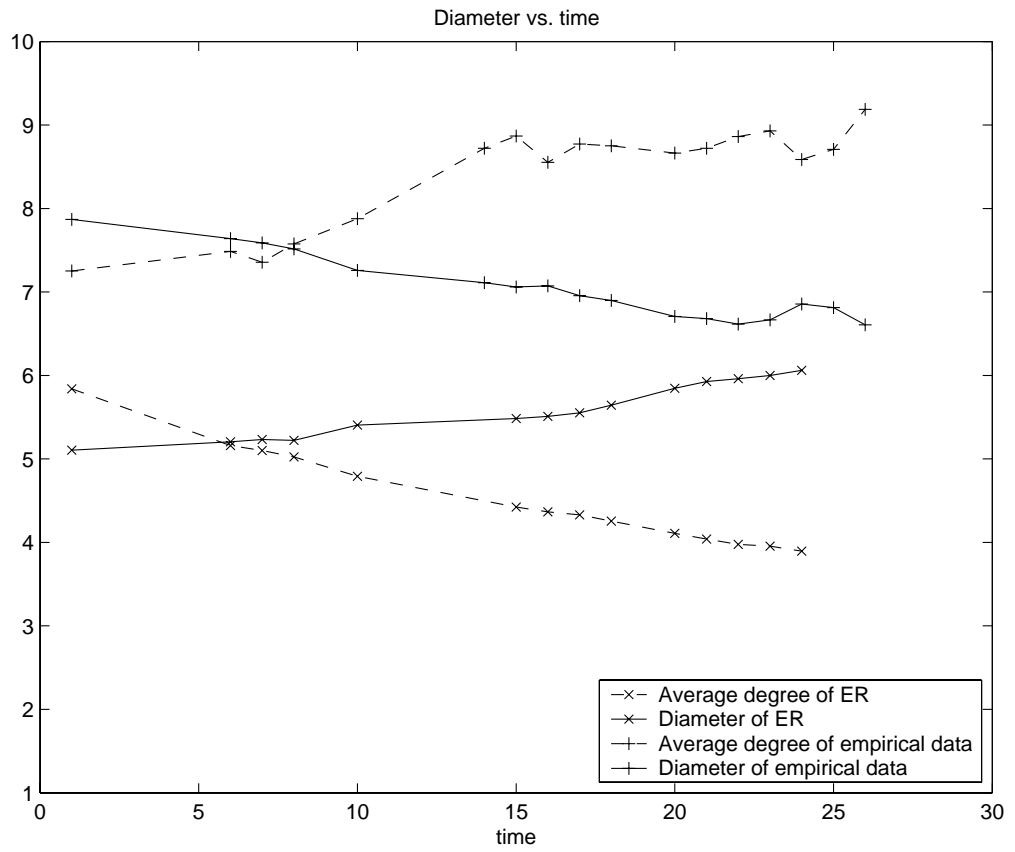


Figure 4.3. Average degree and diameter in ER simulation and the empirical data: unit of X is month, 0 represents December 2000 and simulated time 0

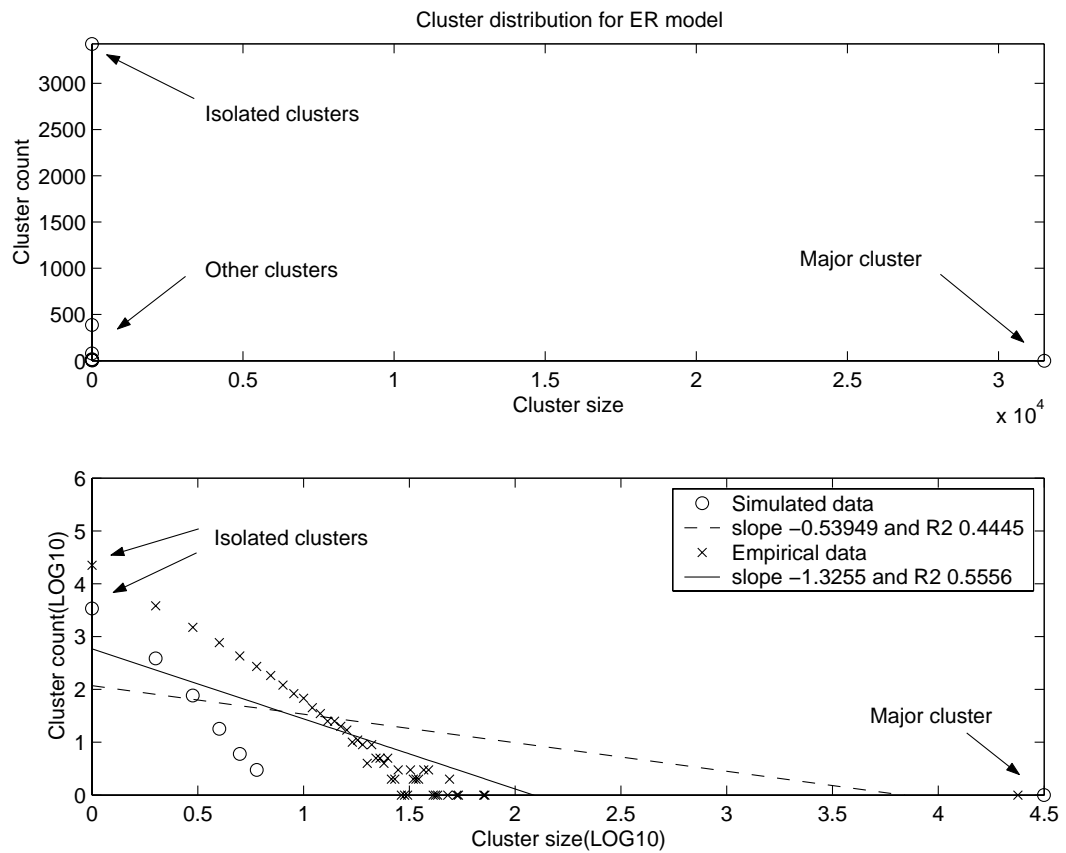


Figure 4.4. Cluster distribution in ER simulation and the empirical data

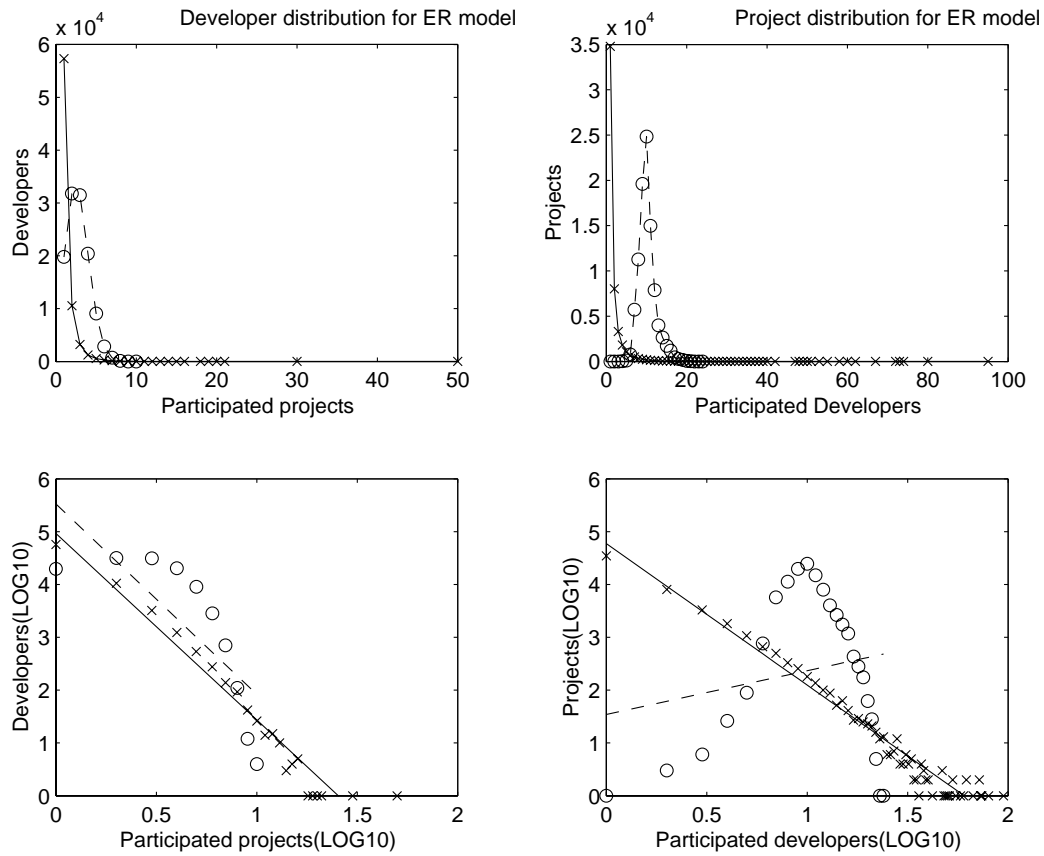


Figure 4.5. Developer and project distributions in ER simulation and the empirical data: 'x' represents empirical data and 'o' represents simulation data of ER model

results of the ER model are shown in Figure 4.5. 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 proven Poisson distribution. Also, we compared them with the distributions of empirical data in log-log coordinates, which are shown in the lower figures. For empirical data, the developer and project distributions fit a straight line well, characterizing the power law distribution. We found a linear regression with a slope of -3.5311 and a R^2 of 0.9533 for the developer degree 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. We found the developer degree distribution has a R^2 as 0.619 and the project degree distribution has a R^2 of 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. 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.

4.4.2 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. Since growth is already included in our simulation of adjusted ER model, so we just needed 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, and the rules that a developer used to choose what project to join, which is called project preference. 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

$$P(i) = \frac{k_i}{\sum_{j \in AllProjects} k_j} \quad (4.3)$$

For developer preference, we suppose that the more projects in which the developer participated, the higher probability he would take an action (create, join or abandon) as opposed to being idle. So in our simulation, the degree the developer already had was also one of the factors that determined the developer's action probabilities. As we discussed in Chapter 2, our SourceForge collaboration network is different from the well-studied collaboration networks (e.g., actor collaboration network and scientist collaboration network), in the sense that edge removal (detachment) is possible in our network. This detachment also influences the developer's preference in the network. By studying the empirical data, we found that the probability of link removal increased when the degree of a developer increased. At some point, the probability of link removal will dominate the action selection of a developer, which means that the developer has reached some limitation of total project participation and can not join any more projects without dropping some of the links.

In implementation, a probability in $[0,1]$ was generated first, and the action selection decision was based on which range (every action has its own probability range) the probability was in. The four action ranges are changing according to the developer degree, which is shown in Figure 4.6.

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 4.3. The initial values of these parameters are chosen by the

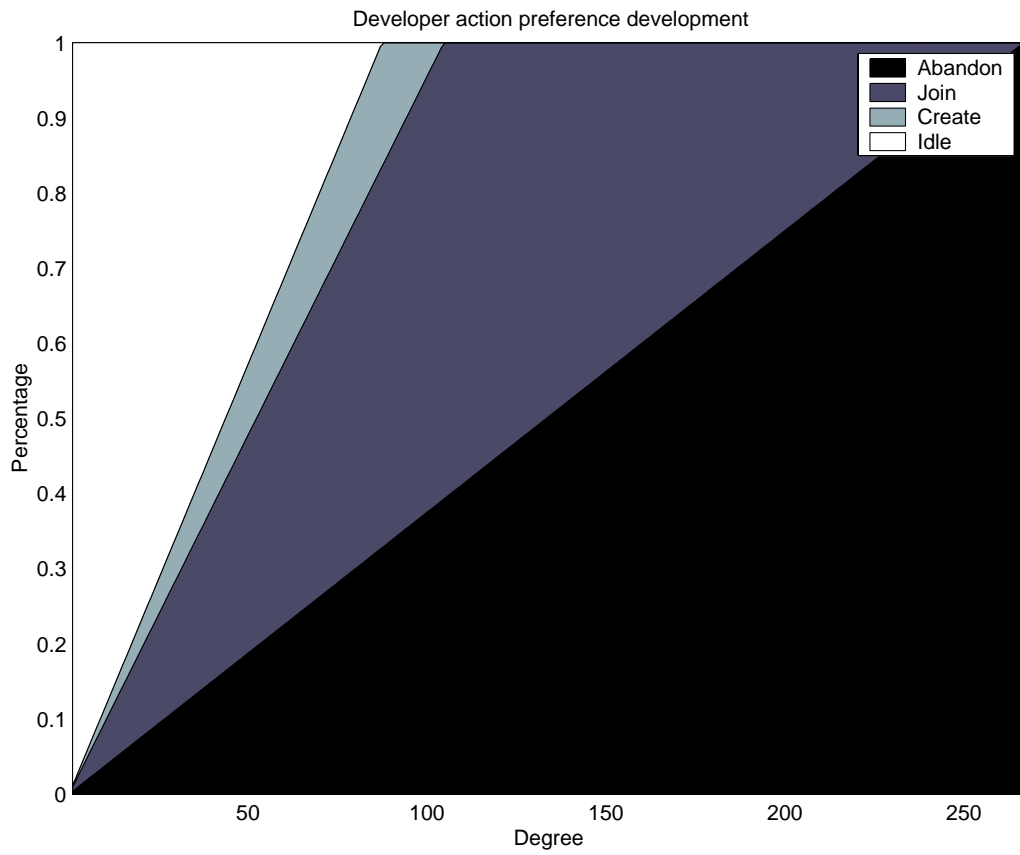


Figure 4.6. Developer action probabilities trend

statistical average of the empirical data, which are the same parameters used in the adapted ER model. However, these parameters are not able to reproduce the empirical data. So we need to calibrate the parameters. The final parameters we used in this iteration are listed below: 1) $N(ND)$ is a random generated number (uniform distributed) which averages 170 per day. 2) $P(CN)$ and $P(JN)$ are 0.1558 and 0.8442, which are same for all models. 3) $P(CO)$, $P(JO)$, $P(AO)$ and $P(IO)$ are 0.001885, 0.0058, 0.00375 and 0.988565, respectively. These parameters are different from the parameters used in ER model, where no preferences existed. But the parameters in the BA model are the same as the parameters in the models evaluated later in this chapter.

First, we investigated the diameter and the clustering coefficient of simulation data based on the BA model. The results are shown in Figure 4.7. The diameter is around 7 and decreasing. The clustering coefficient is around 0.7 and decreasing. They both adequately match the empirical data.

Then we investigated the degree distributions of this simulation iteration – BA model. The developer and project distributions of our simulation for the BA model are shown in Figure 4.8. In the figure, the upper two figures are the developer distribution and project distribution in normal coordinates and 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 polynomial fit; the “o” points are the distribution of simulated data and the dashed line is its linear polynomial fit. We observe that for the developer distribution, the simulation data fits the empirical data quite well. 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, however, do not fit well. The function of linear regression for simulated data is $y = -1.2454x + 2.5823$ and R^2 is 0.442. The

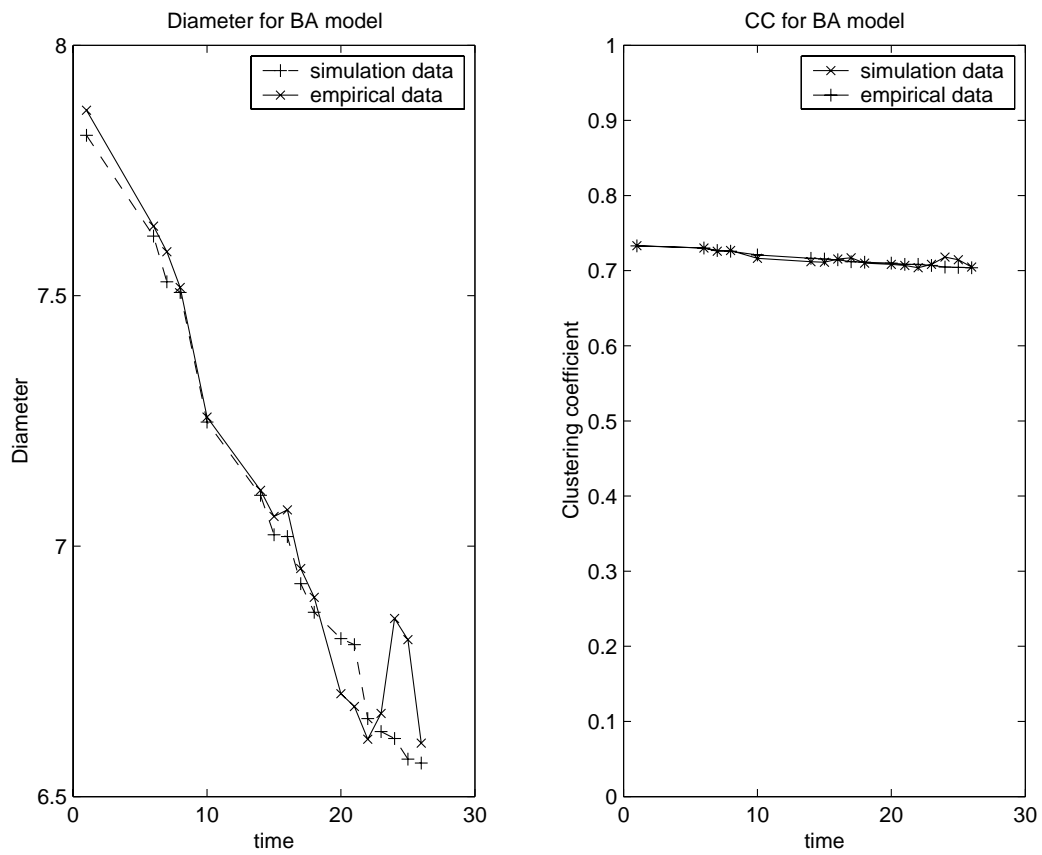


Figure 4.7. Diameter and clustering coefficient in BA simulation and the empirical data: unit of X coordinate is month and 0 represent December 2000

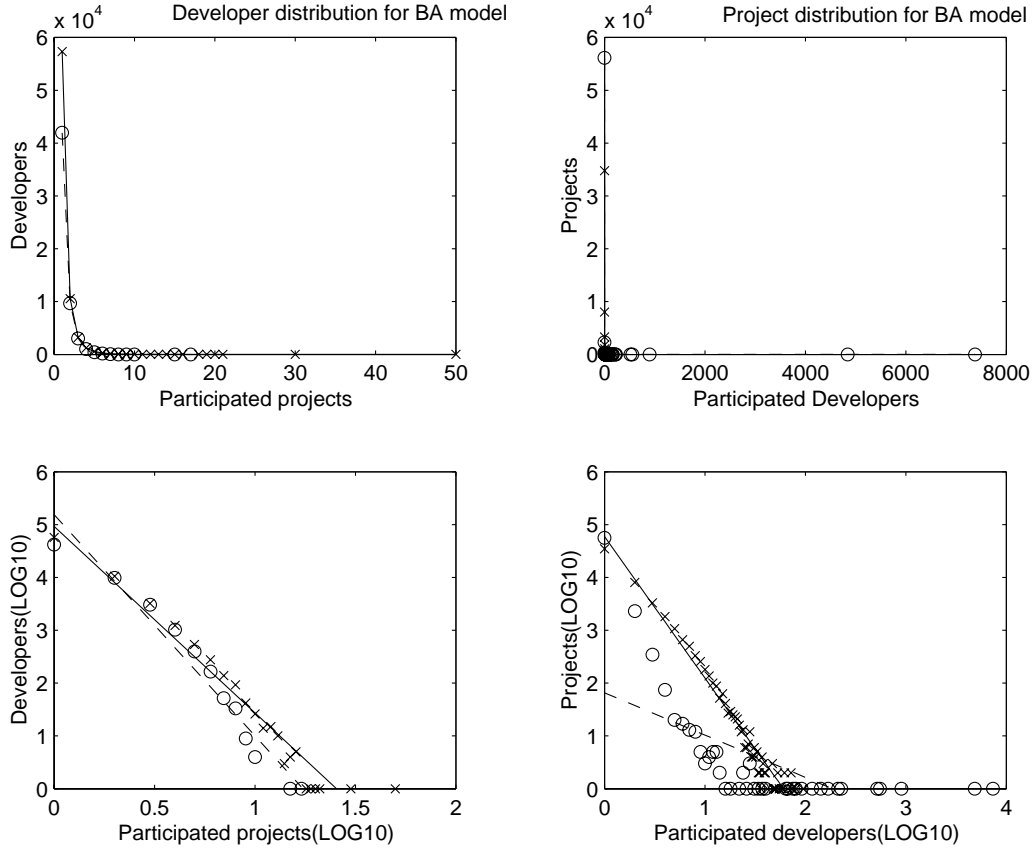


Figure 4.8. Developer and project distributions in BA simulation and the empirical data: ‘o’ represents simulation data of BA model and ‘x’ represent empirical data

discrepancy between the linear regressions of the simulation data and empirical data occurs because of the tails, especially for 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 100 developers.

Also there exists the phenomenon that young nodes were 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 the 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.

4.4.3 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 Barabási 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 ,

$$P_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \quad (4.4)$$

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 for the fitness.

$$P(\eta) = \lambda e^{-\lambda\eta} \quad (4.5)$$

where λ is equal to 2.

The degree distribution of this model is mathematically proven to be

$$P(k) \sim \frac{k^{-C-1}}{\ln(k)} \quad (4.6)$$

which is a power law with logarithmic correction [6] (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 4.9. 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 100 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 data¹.

So the BA model with constant fitness not only matches most of the statistics of the empirical data but also is capable of explaining the “young upcomer” phenomenon. It still has some deficiency in reproducing the same result as the empirical data especially in project distribution. Further iteration was needed.

4.4.4 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 is a non-negative integer. The rate of decay is uniform for all projects. Every project possessed a random fitness when it was created by a developer.

¹The related figures are included in Appendix A

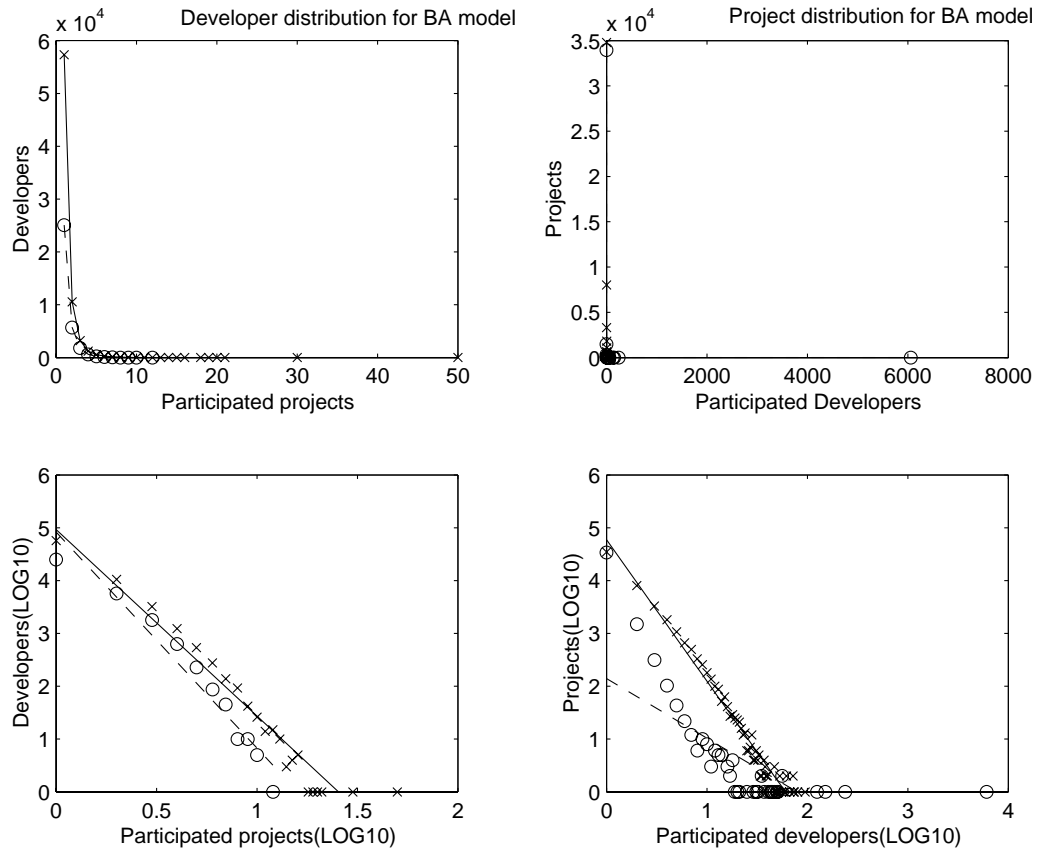


Figure 4.9. Degree distributions in BA simulation with constant fitness and the empirical data: 'o' represents simulation data of BA model with constant fitness and 'x' represent empirical data. For developer distribution, function of linear regression is $y = -4.1860x + 5.1876$ and R^2 is 0.9559; for project distribution, function of linear regression is $y = -0.8008x + 1.8169$ and R^2 is 0.5261

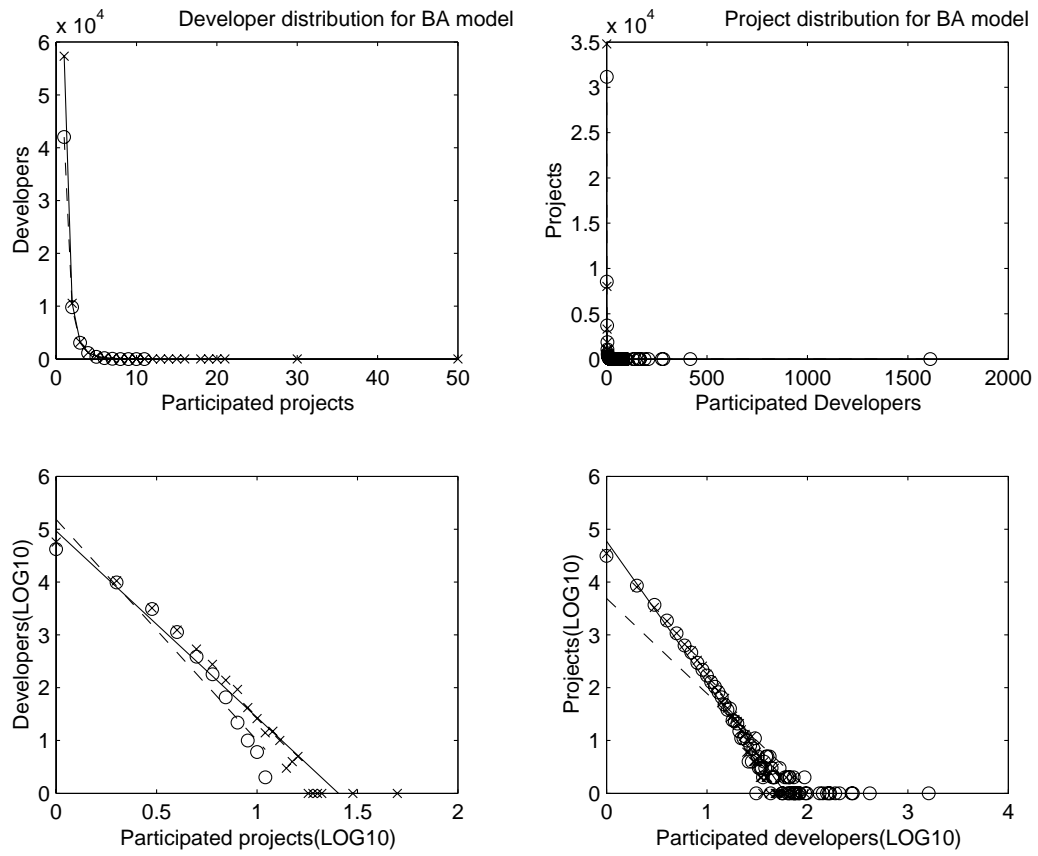


Figure 4.10. Degree distributions of BA simulation with dynamic fitness and the empirical data: 'o' represents the data points from the simulated data and 'x' represents the data points from the empirical data

The results of the degree distributions of the BA model with dynamic fitness are shown in Figure 4.10. 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 data².

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 than the BA model with constant fitness, in relation to the attributes that we investigated (degree distribution, diameter and clustering coefficient).

4.5 Discussion

In our simulation iterations, we have an interesting observation, which need further discussion.

- *Independence between the developer distribution and the project distribution*

In the simulation iterations, the distribution of developer and the distribution of project are loosely related. Actually, in our simulation experiments, they were independent of each other. Figure 4.11 shows the degree distribution from one of our simulations. This simulation is identical to the simulation of the BA model with dynamic fitness except that developer preference was not considered. In the figure, significant power law in project distribution can be found

²The related figures are included in Appendix A

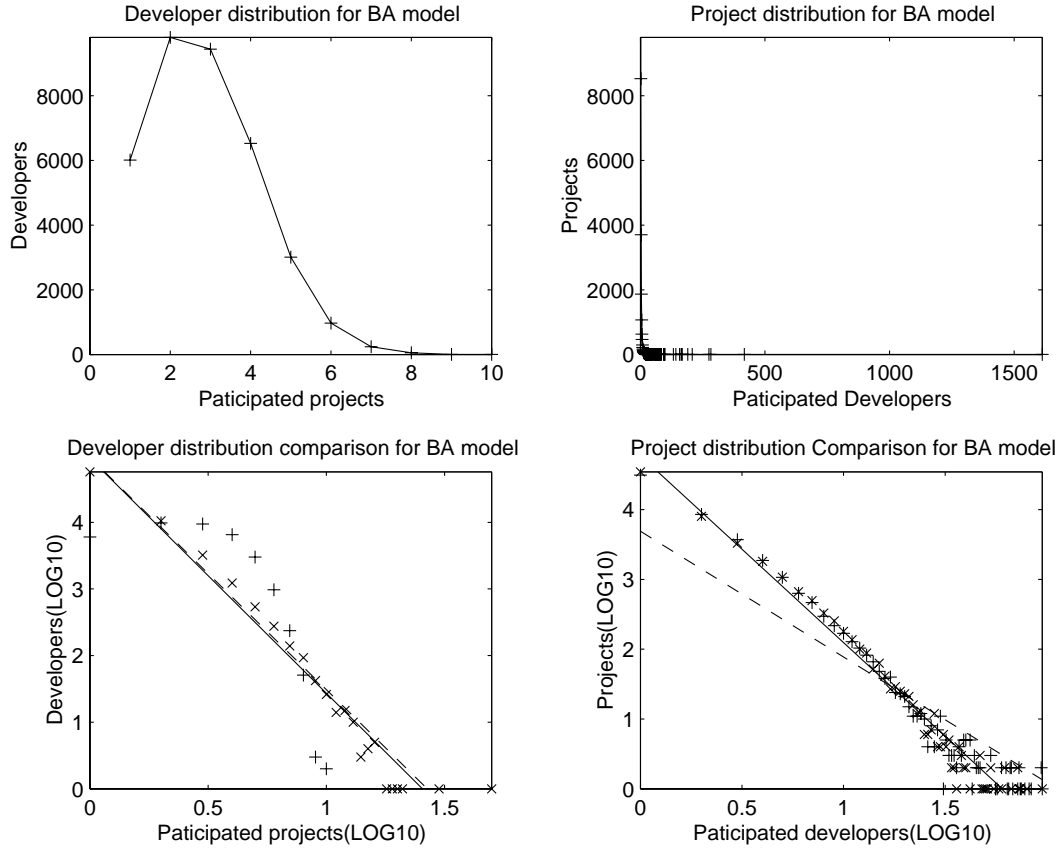


Figure 4.11. Developer and project distributions in partial BA model: “+” represents simulated data and “x” represents empirical data. There are no preference for developer, but there are preference for project

and the project distribution is almost identical to the project distribution in Figure 4.10, where both developer preference and project preference exists. But the developer distribution in this simulation is a normal distribution, due to the randomness of developer action selection (without the developer preference). This developer distribution is different from the one in the BA model shown in Figure 4.10, which shows significant power law. We have also found similar phenomena in other simulations. This is because our simulation was based on the bipartite graph. This structure distinguished the developer

groups from the project groups. Developer distribution was determined by the developer preference and the project was just the object in action selection. So the project distribution (or the project preference) will not interfere with the developer distribution for the same reason that the developer distribution (or the developer preference) did not interfere with project distribution. The independence between project distribution and developer distribution allowed us to investigate the two distributions separately, which made inspection of the network simpler.

4.6 Summary

In this chapter, we used simulation methods to assist in understanding the topology and evolution of the SourceForge developer collaboration network. First we proposed a framework for simulation related study. Following this framework, we iterated all the well-known models (ER model, BA model and so on) in complex network with similar parameters to generate the “virtual” SourceForge. By comparing the simulation outputs and the empirical data, we confirmed the results in Chapter 3 and also verified the correctness of our proposed model. Finally, in simulation, we also had some interesting discoveries about the SourceForge developer collaboration network, such as the independence of distributions, life-cycle of developers and projects and the evolutionary patterns of the network. The results in this chapter are summarized in Table 4.2.

Table 4.2. SUMMARY OF SIMULATION RESULT

Model	Parameter	Expected Pattern	Observed Pattern
ER	Developer distribution	Power law	Normal
	Project distribution	Power law	Normal
	Cluster distribution	Power law	Power law
	Average degree	Increasing	Decreasing
	Clustering coefficient	Decreasing (large value)	Decreasing (small value)
	Diameter	Decreasing	Increasing
BA	Developer distribution	Power law	Power law
	Project distribution	Power law	Power law (heavy tail)
	Cluster distribution	Power law	Power law
	Average degree	Increasing	Increasing
	Clustering coefficient	Decreasing (large value)	Decreasing (large value)
	Diameter	Decreasing	Decreasing
	“Young upcomer”	Existing	Not existing
BA with constant fitness	Developer distribution	Power law	Power law
	Project distribution	Power law	Power law (heavy tail)
	Cluster distribution	Power law	Power law
	Average degree	Increasing	Increasing
	Clustering coefficient	Decreasing (large value)	Decreasing (large value)
	Diameter	Decreasing	Decreasing
	“Young upcomer”	Existing	Existing
BA with dynamic fitness	Developer distribution	Power law	Power law
	Project distribution	Power law	Power law (small tail)
	Cluster distribution	Power law	Power law
	Average degree	Increasing	Increasing
	Clustering coefficient	Decreasing (large value)	Decreasing (large value)
	Diameter	Decreasing	Decreasing
	“Young upcomer”	Existing	Existing

CHAPTER 5

CONCLUSION

5.1 Summary

We have discussed how Open Source Software (OSS) development is a classic example and prototype of a collaborative social network. Based on the empirical data we collected from SourceForge over the last two years, we were able to investigate the structure and the dynamic mechanisms that determined the topology and governed the evolution of such systems. SourceForge is one of the largest online collaborators for open source development projects, hosting over 50 thousand projects and over 80 thousand developers. Our empirical data on projects and developers was collected monthly beginning in January 2001. In this thesis, we analyzed the empirical data we collected from SourceForge in order to obtain statistics and topological information of the OSS developer collaboration network. We extracted the parameters of the evolution by inspecting the longitudinal properties of the network. We generated a model that depicted the evolution of this collaboration network. Finally, we used simulation to verify and validate the models for the SourceForge community.

In the statistical study of the empirical data, we investigated not only 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. In the study, we calculated all of these properties of the empirical data and compared them with the properties of all existing models

(The results are shown in table 3.1 and table 4.2). The BA model with constant fitness could explain all the high-level statistics of the network, like power law degree distribution, relatively large clustering coefficient and relatively small diameter. But we also observed that even the BA model with constant fitness could not depict all the existing properties of the SourceForge collaboration network, especially the developing patterns of the individual entities (developer or project). Based on these observations, we proposed our model for the SourceForge collaboration network, which is based on dynamic fitness and the bipartite graph.

During the statistical study, we also observed that cluster related properties may have an important role in presenting the topology and evolution of the network. First, cluster distribution also fits the power law. Second, the relative size of major cluster and the relative size of isolated clusters may be a good indication of the network evolution.

In studying the simulation model of the SourceForge collaboration network, we proposed a framework for modeling and simulation related studies and applied this framework to our study of the SourceForge collaboration network. We used the parameters we extracted from empirical data as the simulation parameters and iterated from the ER model, BA model and BA model with constant fitness until finally arriving at the proposed BA model with dynamic fitness. Through simulation, we verified all the properties of these models that could be calculated by mathematical methods (degree distribution) and also obtained the other properties that were difficult for mathematics to calculate (diameter, clustering coefficient and developing patterns). Then by comparing the simulation output with the empirical data, we concluded that our proposed model, BA model with dynamic fitness, is the fittest model for the SourceForge collaboration network because it could reproduce all the properties investigated in this thesis.

5.2 Limitations

There were several limitations in our work, which are listed as follows:

- Our work was based on the empirical data we obtained from SourceForge. However this empirical data was incomplete, because 1) SourceForge is still active and we were not able to have a complete set of the empirical data covering all the possible evolution stages of the collaboration network. 2) We did not have the complete set of empirical data of the past years and the empirical data for several months was not available. This “incompleteness” may have biased the research results.
- Although SourceForge is one of the biggest hosting sites in the OSS community, this is only a single case. Studies of a single case could not reveal the common topology and evolution properties of the entire OSS community.
- Simulations of complex networks are too complicated for simulation toolkits to provide the user all the tunable parameters, so these toolkits always have some default setting for some simulation parameters that the user can not change. Thus using single simulation toolkits may also bring unexpected errors into the simulation output. We just used Swarm to run the simulation. This may cause some bias in the output.
- In the simulation iterations, we found that the parameters we deduced from the statistics of the empirical data did not exactly fit into our simulation. Thus we needed to adjust these parameters according to the simulation results. Unfortunately, these parameters (7 parameters) were dependent on each other and it was exceedingly difficult finding good parameters by a naive “guess and try” method. We cannot guarantee the final simulation we have is optimal.

5.3 Future work

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

1. We need more complete empirical data for experiments by continuing to collect data from SourceForge.
2. We may analyze other famous OSS hosting site, like Savannah, using the same methods.
3. We could migrate our simulations to Repast to verify the inference of simulation toolkits to the simulation output.
4. We could use more advanced methods like operation research optimization in fine-tuning the parameters to find the best solutions.

APPENDIX A

EXTRA SIMULATION RESULTS

Table A.1. ORIGINAL PARAMETERS FOR EMPIRICAL DATA, PART I

Month	Devs	Projs	New Devs	Retired Devs	New Projs	Discard Projs	Detachments	Attachments	Links
Feb-02	48640	33670							
Mar-02	48845	33248	1000	795	55	477	1303	1415	66085
Apr-02	52915	36731	4694	624	3816	333	1040	6868	66197
May-02	56144	39025	3620	391	2466	172	785	5342	72025
Jun-02	59531	41484	4462	1074	3051	592	2039	6717	76582
Aug-02	65761	46183	6939	709	4842	143	1350	10491	81260
Sep-02	66863	46932	1199	97	768	19	203	1758	90401
Oct-02	69312	48616	2728	279	1743	59	544	4016	91956
Nov-02	73367	51500	4600	545	2997	113	1041	6745	95428
Dec-02	73495	51676	1448	1320	884	708	2805	2204	101132
Jan-03	76271	53773	3128	352	2135	38	648	4805	104689
Feb-03	79701	56128	3838	408	2404	49	758	6670	110600

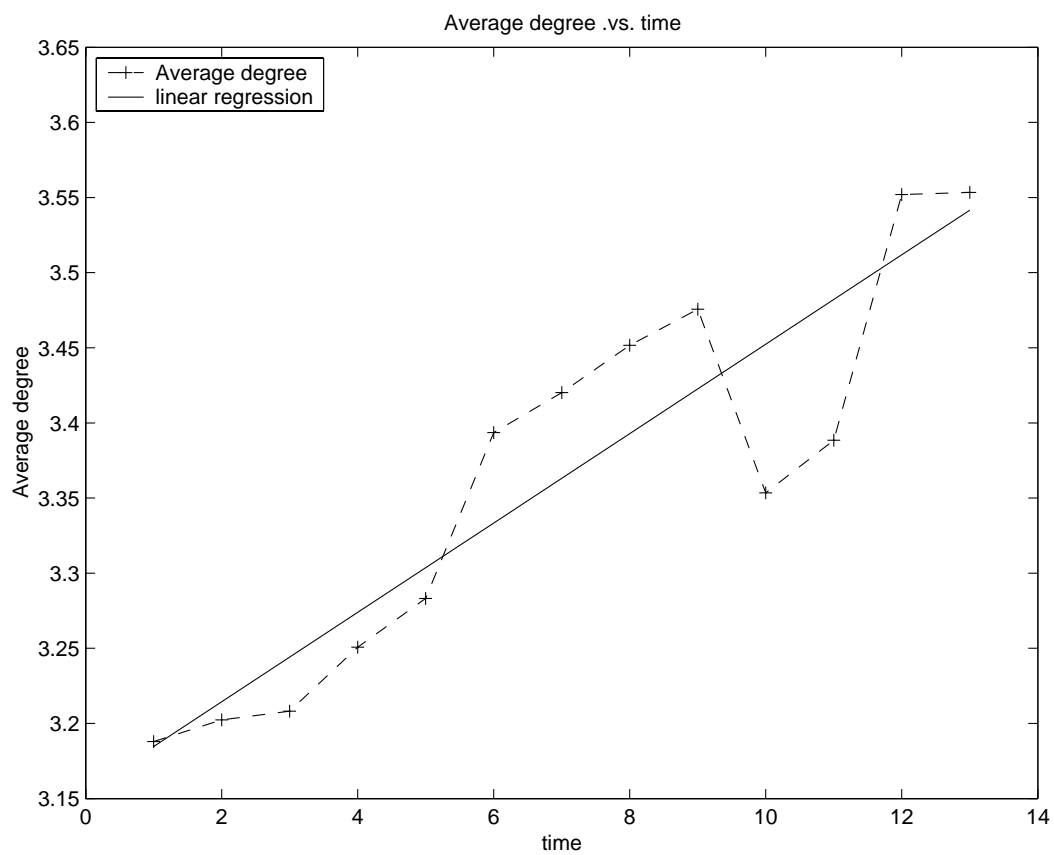


Figure A.1. Average degree of project network: unit of X coordinate is month and 0 represents January 2002

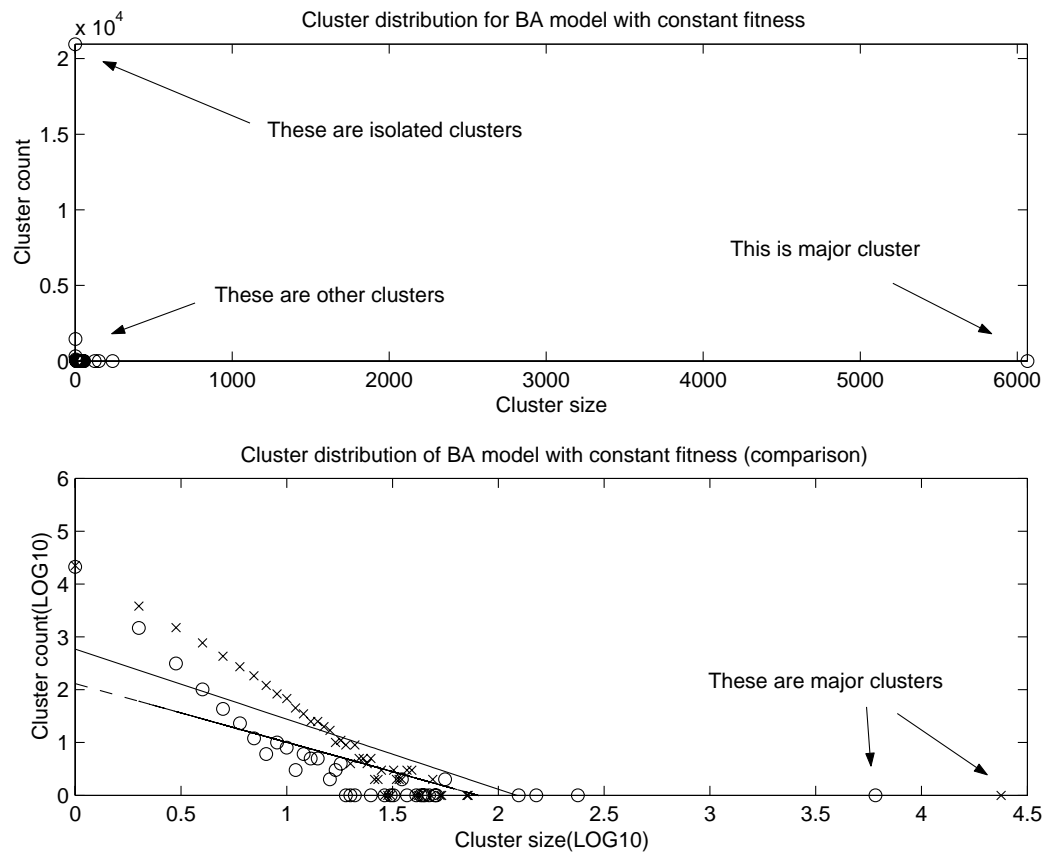


Figure A.2. Cluster distribution of BA model with constant fitness: 'o' represents simulated data and 'x' represents empirical data. The R^2 of the linear regression for simulated data is 0.5317

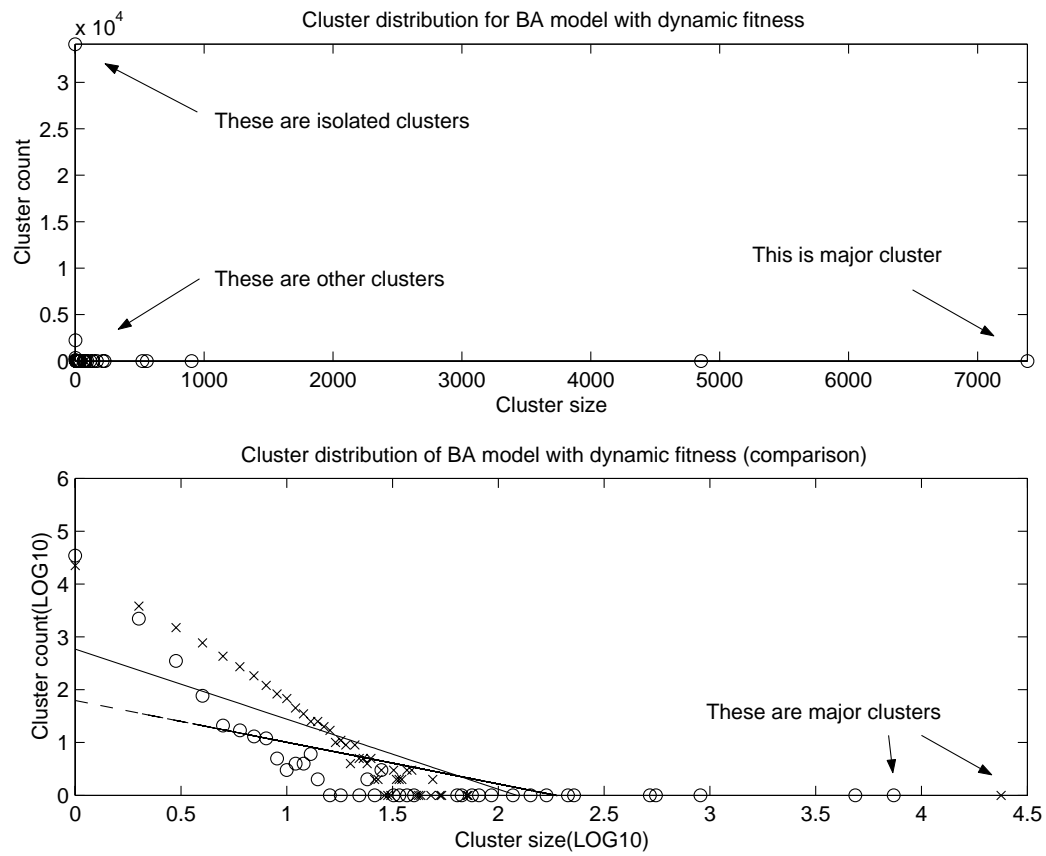


Figure A.3. Cluster distribution of BA model with dynamic fitness: 'o' represents simulated data and 'x' represents empirical data. The R^2 of the linear regression for the simulated data is 0.4490

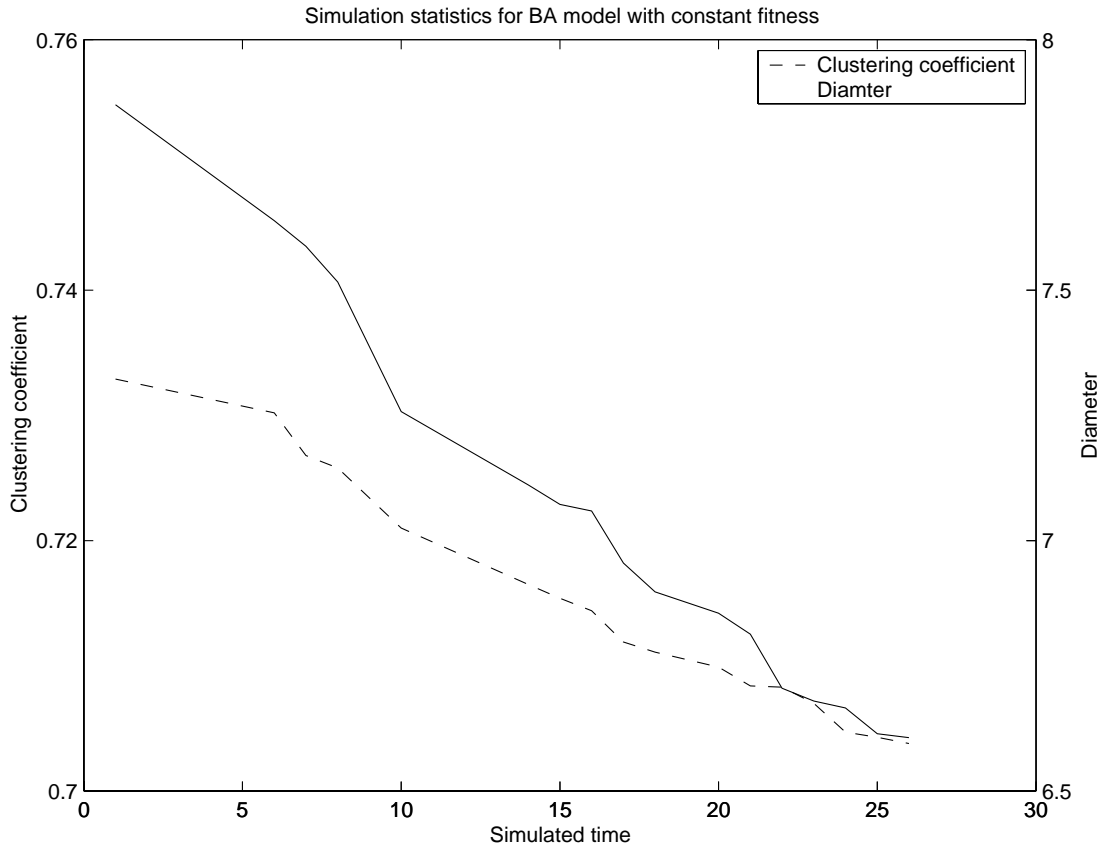


Figure A.4. Diameter and clustering coefficient for BA model with constant fitness

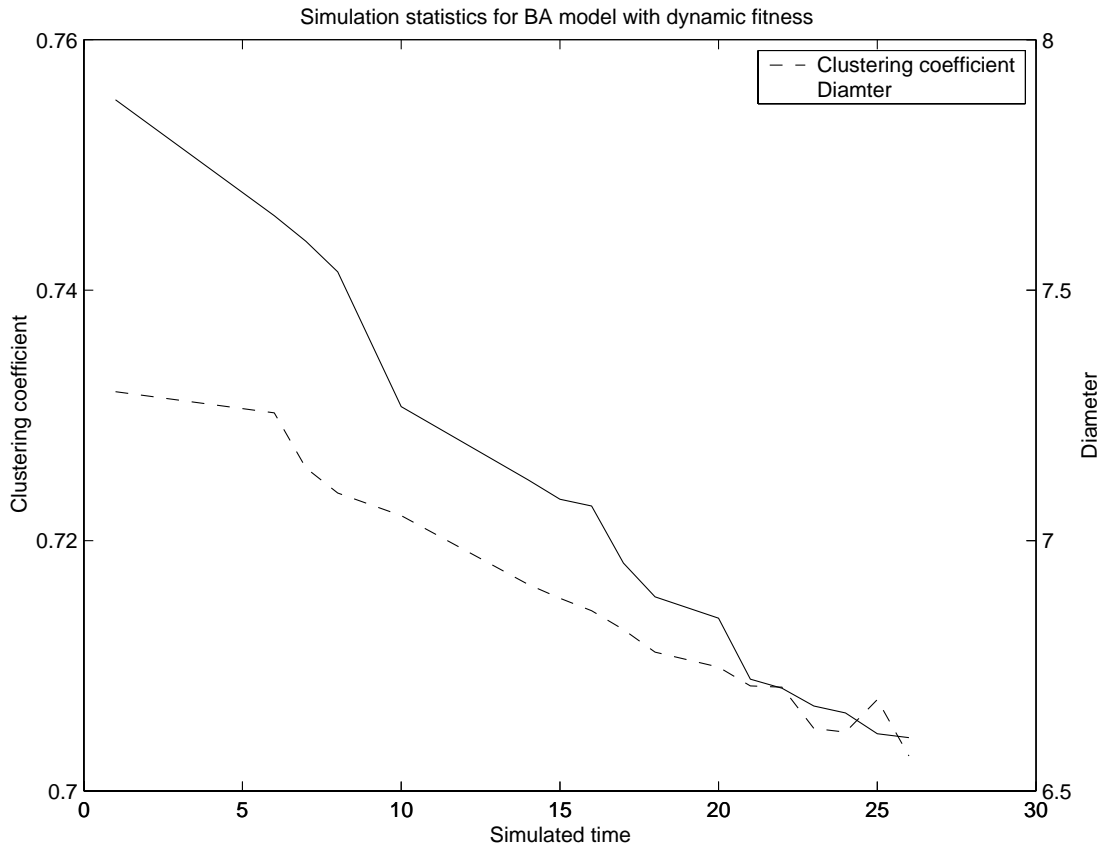


Figure A.5. Diameter and clustering coefficient for BA model with dynamic fitness

Table A.2. ORIGINAL PARAMETERS FOR EMPIRICAL DATA, PART II

Month	Devs	Newdevs	$\frac{Create}{New}$	$\frac{Join}{New}$	Olddevs	$\frac{Create}{Old}$	$\frac{Join}{Old}$	$\frac{Abandon}{Old}$	$\frac{No_action}{Old}$
Feb-02	48640								
Mar-02	48845	1000	63	937	47845	27	343	447	47028
Apr-02	52915	4694	4136	558	48221	1414	216	389	46202
May-02	56144	3620	2575	1045	52524	982	312	341	50889
Jun-02	59531	4462	3129	1333	55070	1232	414	860	52564
Aug-02	65761	6939	5105	1834	58822	2068	534	563	55657
Sep-02	66863	1199	712	487	65664	283	182	94	65105
Oct-02	69312	2728	1774	954	66584	675	312	224	65373
Nov-02	73367	4600	3095	1505	68767	1145	492	410	66720
Dec-02	73495	1448	749	699	72047	360	294	1241	70152
Jan-03	76271	3128	2007	1121	73143	836	422	252	71633
Feb-03	79701	3838	2719	1119	75863	1855	391	290	73327

APPENDIX B

DOCKING EXPERIENCE BETWEEN SWARM AND REPAST

This section describes docking of Repast and Swarm simulations on four OSS network models – ER, BA, BA with constant fitness and BA with dynamic fitness. There are several related docking experiences before [10, 50, 18]. I presents the results and comparisons. Furthermore, discussion of our docking process is given. In order to study the validity of our simulation, we compare attributes including degree distribution, diameter and clustering coefficient in Swarm and Repast simulations.

Degree distribution $p(k)$ is the distribution of the degree k throughout the network. The degree k of a node equals the total number of other nodes to which it is connected. Degree distribution was believed to be normal until Albert and Barabási found it fit a power law distribution in many real networks [6]. Figure B.1 gives developer distributions in the BA model implemented by Swarm and Repast. In the figure, both axes are represented with a logarithmic scale. From the figure, we can observe that developer distribution matches the power law distribution. We also found such a distribution in other three network models. However, there are slightly difference between Swarm results and Repast results. We believe these differences are caused by different random generators associated with RePast and Swarm. The diameter of a network is the maximum distance between any pair of connected nodes. The diameter can also be defined as the average length of the shortest paths between any pair of nodes in the graph. In this paper, the second

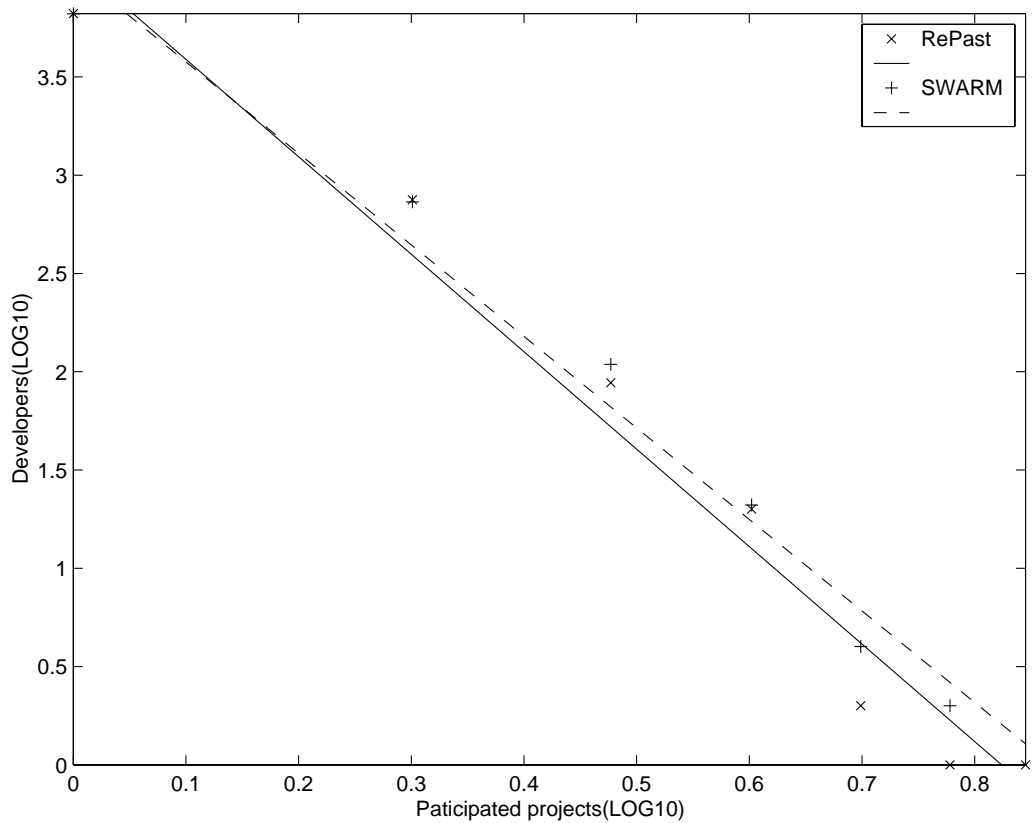


Figure B.1. Developer Distributions

definition is used since the average value is more suitable for studying the topology of the OSS network.

The neighborhood of a node consists of the set of nodes to which it is connected. The clustering coefficient of a node is a fraction representing the number of links actually present relative to the total possible number of links among the nodes in its neighborhood. The clustering coefficient of a graph is the average of all the clustering coefficients of the nodes represented. Recent research has found that real complex networks typically exhibit a large degree of clustering [4].

Figure B.2 shows the total number of developers and projects relative to the

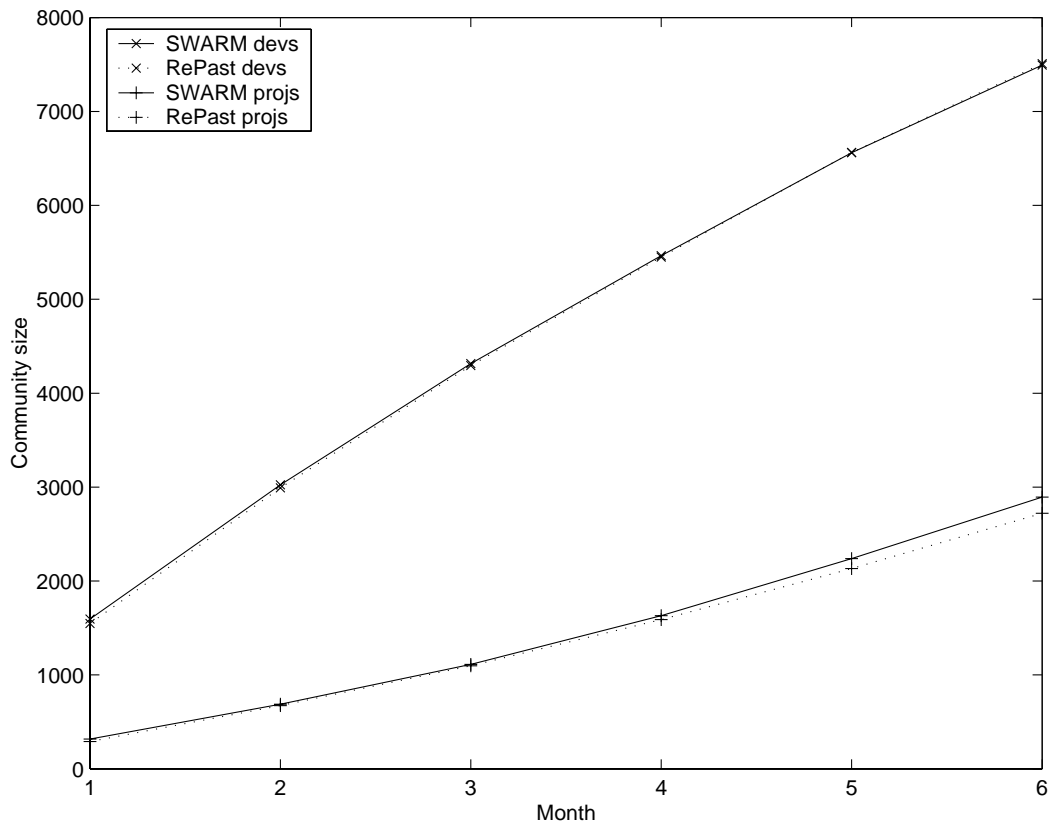


Figure B.2. Community Size Development

time period, which describe the developing trends of size of developers and projects in the network. The size of developers and projects are almost the same for Swarm and RePast simulation.

APPENDIX C

PROOF OF APPROXIMATE METHOD OF GRAPH DIAMETER

Given

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k \quad (\text{C.1})$$

as the generating function for p_k , which is the probability of a vertex have k degree.

we have the following properties:

$$\sum p_k = G_0(1) = 1 \quad (\text{C.2})$$

$$p_k = \frac{1}{k!} \frac{d^k G_0}{dx^k} \Big|_{x=0} \quad (\text{C.3})$$

$$z_1 = \langle k \rangle = \sum_k k p_k = G'_0(1) \quad (\text{C.4})$$

$$(\text{C.5})$$

We define p_k^1 as the probability of the degree of the vertices that we arrive at by following a randomly chosen edge. So

$$P_k^1 = \frac{k p_k}{\sum_k k p_k} \quad (\text{C.6})$$

Thus, we define the generating function for p_k^1 as

$$G_1(x) = \frac{\sum_k k p_k x^{k-1}}{\sum_k k p_k} \quad (\text{C.7})$$

We have the similar properties for $G_1(x)$ as for $G_0(x)$:

$$\sum p_k^1 = G_1(1) = 1 \quad (\text{C.8})$$

$$p_k^1 = \frac{1}{(k-1)!} \frac{d^k G_1}{dx^{k-1}} \Big|_{x=0} \quad (\text{C.9})$$

$$(\text{C.10})$$

Using power property of generating function, we can get the generating function for the distribution of the number of second neighbors of the original vertex as:

$$G_2(x) = \sum_k p_k [G_1(x)]^k = G_0(G_1(x)) \quad (\text{C.11})$$

Thus, by the moment property of generating function, we get

$$z_2 = \left[\frac{d}{dx} G_0(G_1(x)) \right]_{x=1} = G'_0(1) G'_1(1) \quad (\text{C.12})$$

Furthermore, we can get every z_m as

$$z_m = G'_0(1) [G'_1(1)]^{m-1} = \left[\frac{z_2}{z_1} \right]^{m-1} z_1 \quad (\text{C.13})$$

The overall network size, N , is

$$N = 1 + \sum_m z_m \quad (\text{C.14})$$

when $N \gg z_2 \gg z_1$, we can get

$$N \approx z_m \quad (\text{C.15})$$

Combining the Equation C.13 and Equation C.15, when $z_m \rightarrow N$ and $m \rightarrow D$, where D is the diameter of the network, we get

$$N = \left[\frac{z_2}{z_1} \right]^{D-1} z_1 \quad (\text{C.16})$$

So, we can get

$$D = \frac{\log \frac{N}{z_1}}{\log \frac{z_2}{z_1}} + 1 \quad (\text{C.17})$$

BIBLIOGRAPHY

- [1] J. Abello, P.M. Pardalos and M.G.C. Resende. In external memory algorithms. *MIMACS series in Discrete Mathematics Theoretical Computer Science*, page 119, 1999.
- [2] L.A. Adamic and B.A. Huberman. Scaling behavior of the world wide web. *Science*, 287(2115), 2000.
- [3] W. Aiello, F. Chung and L. Lu. Random evolution in massive graphs. *IEEE Symposium of Foundations of Computer Science*, 2001.
- [4] R. Albert and A.L. Barabási. Dynamics of complex systems: Scaling laws for the period of boolean networks. *Physics Review*.
- [5] R. Albert and A.L. Barabási. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [6] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(47), 2002.
- [7] R. Albert, H. Jeong and A.L. Barabási. Diameter of world-wide web. *Nature*, 401(130), 1999.
- [8] L.A.N. Amaral, A. Scala, M. Barthelemy and S. HE. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21), 2000.
- [9] R. Axelrod. The evolution of strategies in the iterated prisoner dilemma. *Genetic algorithm and simulated annealing*, pages 32–41, 1987.
- [10] R. Axelrod. Advancing the art of simulation in the social sciences. *Simulating Social Phenomena*, ed. Rosaria Conte, Rainer Hegselmann, and Pietro Terna., pages 21–40, 1997.
- [11] R. Axelrod. *The complexity of Cooperation: Agent based models of competition and collaboration*. Princeton University Press, Princeton, NJ, 1997.
- [12] A.L. Barabási, H. Jeong, Z. Neda, E. Ravasz, A. Schubert and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica*, 311(590), 2002.
- [13] J.M. Barrie and D.E. Presti. The world wide web as an instructional tool. *Science*, 274(371), 1996.

- [14] G. Bianconi and A.L. Barabási. Competition and multiscaling in evolving networks. *Europhys. Lett.*, 54(436), 2001.
- [15] B. Bollobás. Random graphs. *London:Academic*, 1985.
- [16] F. Brooks. No silver bullet: essence and accidents of software engineering. *IEEE Computer Magazine*, pages 10–19, 1987.
- [17] A. Bunde and S. Havlin. *Fractals and disordered systems, 2nd Edition*. Springer, Berlin, 1996.
- [18] R. Burton. *Simulating Organizations: Computational Models of Institutions and Groups*, chapter Aligning Simulation Models: A Case Study and Results. AAAI/MIT Press, Cambridge, Massachusetts, 1998.
- [19] J. Camacho, R. Guimerà and L.A.N. Amaral. preprint cond-mat/0102127. 2001.
- [20] G. Drummond. Open source software and documents: A literature and online resource review. [http : //www.omar.org/opensource/litreviewa](http://www.omar.org/opensource/litreviewa), 1999.
- [21] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.
- [22] M. Faloutsos, P. Faloutsos and C. Faloutsos. On power-law relationships of the internet topology. *Computer Communication Review*, 29(251), 1999.
- [23] Y. Gao, V. Freeh and G. Madey. Analysis and modeling of the open source software community. *NAACSOS, Pittsburgh*, 2003.
- [24] Y. Gao, V. Freeh and G. Madey. Conceptual framework for agent-based modeling and simulation. *NAACSOS, Pittsburgh*, 2003.
- [25] Y. Gao, V. Freeh and G. Madey. Topology and evolution of the open source software community. *SwarmFest, Notre Dame*, 2003.
- [26] Netlogo Development group. Netlogo introduction. [http : //ccl.sesp.northwestern.edu/netlogo/](http://ccl.sesp.northwestern.edu/netlogo/).
- [27] Swarm Development group. A tutorial introduction to swarm. [http : //www.swarm.org](http://www.swarm.org).
- [28] D. Hiebeler. The swarm simulation system and individual-based modeling. *Advanced technology for natural resource management*, 1994.
- [29] W. Howe. When did internet start? a brief capsule history. <http://www.delphi.com/navnet/faq/history>, May 1996.
- [30] Repast Information. <http://repast.sourceforge.net/>. 2002.
- [31] Starlogo Information. <http://education.mit.edu/starlogo/>. 2002.
- [32] S. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, Oxford, 1993.

- [33] P.J. Kiviat. Simulation, technology, and the decision process. *ACM Transactions on Modeling and computer Simulation*, 1(2):89, 1991.
- [34] F. Liljeros, C. Edling, A. LAN, S. HE and Y. Aberg. Hub caps could squash STDs. *Nature*, 411(907), 2001.
- [35] K. Carley M. Prietula and L. Gasser. *Simulating organizations: computational models of institutions and groups*. MIT Press, Cambridge, MA, 1998.
- [36] G. Madey, V. Freeh and R. Tynan. Agent-based modeling of open source using swarm. *Eight Americas Conference of Information Systems*, 2002.
- [37] G. Madey, V. Freeh and R. Tynan. The open source software development phenomenon: an analysis based on social network theory. *Eight Americas Conference of Information Systems*, 2002.
- [38] G. Madey, V. Freeh, R. Tynan and Y. Gao. Agent-based modeling and simulation of collaborative social networks. *AMCIS, Tampa*, 2003.
- [39] J. G. March. Exploration and exploitation in organizational learning. *Organizational science*, pages 71–87, 1991.
- [40] A. Border, R. Kumar, F. Maghoul, P. Raghavan, S. Rajalopagan, R. Stata, A. Tomkins and J. Wiener. Graph structure in the web: experiments and models. *Computer Networks*, 33(309), 2000.
- [41] J.M. Montoya and R.V. Solé. preprint cond-mat/0011195. 2000.
- [42] M.E.J. Newman. Clustering and preferential attachment in growing networks. *Physics Review*, 64(025102), 2001.
- [43] M.E.J. Newman. Scientific collaboration networks: I. network construction and fundamental results. *Physics Review*, 64(016131), 2001.
- [44] M.E.J. Newman. Scientific collaboration networks: Ii. shortest paths, weighted networks, and centrality. *Physics Review*, 64(016132), 2001.
- [45] M.E.J. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci*, 98:404–409, 2001.
- [46] M.E.J. Newman and D.J. Watts. Random graph models of social networks. *Physics Review*, 64(026118), 2001.
- [47] N.S. Padian. STD risks and latino adolescents’ sexual network. <http://aidsscience.org/Preventionproject.asp?ID=661>, 2002.
- [48] S.L. Pimm. *The Balance of Nature*. University of Chicago Press, Chicago, 1991.
- [49] W. Poundstone. *The recursive universe*. Contemporary books, Chicago, IL, 1985.

- [50] J. Epstein R. Axtell, R. Axelrod and M. Cohen. Aligning simulation models: A case study and results. *Computational and Mathematical Organization Theory*, 1(2):123–141, 1996.
- [51] E. Raymond. The magic cauldron. [http : //www.tuxedo.org/ esr/writings/magic –cauldron/magic – cauldron.html](http://www.tuxedo.org/esr/writings/magic-cauldron/magic-cauldron.html), 1999.
- [52] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, MA, 1994.
- [53] P. Riolo. The effects of tag-mediated selection of partners in evolving populations playing the iterated prisoner’s dilemma. *Santa Fe Institute working paper*, (97-02-016), 1997.
- [54] T. Schelling. On the ecology of micromotives. *The corporate society*, pages 19–64, 1974.
- [55] L. Steve and C.L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.
- [56] D.J. Watts. *Small world*. Princenton university press, NJ, 1999.
- [57] D.J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.
- [58] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world. *Nature*, 393(440), 1998.
- [59] H.S. Wilf and D. Zeilberger. Rational functions certify combinatorial identities. 3:147–158, 1990.
- [60] R.J. Williams, N.D. Martinez, E.L. Berlow, J.A. Dunne and A.L. Barabási. Two degrees of separation in complex food webs. *Santa Fe Institute Working Paper Series*, (01-07-036), 2001.