ANALYSIS OF THE STRUCTURE AND EVOLUTION OF AN
OPEN-SOURCE COMMUNITY

BY

HAOYUN HUANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

WASHINGTON STATE UNIVERSITY
Department of Mechanical and Materials Engineering
DECEMBER 2010
To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of HAOYUN-HUANG find it satisfactory and recommend that it be accepted.

______________________________
Jitesh H. Panchal, Ph.D., Chair

______________________________
Uma Jayaram, Ph.D.

______________________________
Sankar Jayaram, Ph.D.
ACKNOWLEDGMENTS

“Every man has a property in his own person. This nobody has any right to but himself. The labor of his body, and the work of his hands, we may say are properly his” – John Locke.

First of all, I want to thank my parents, for their selfless breeding and taking care of me through all these years. I want to thank them for giving me the property and unalienable right of living on this earth.

I want to whole hearted thank Dr. Jitesh Panchal. He is not only my tutor during my Master study but also serve as soul mate of mine. I can still remember every word in his advices about my career and future in these 2 years. I really appreciate his patience and tolerance on my research when the progress was delayed. Without his instruction and creative ideas, I cannot accomplish this thesis and two other papers: one published conference paper and one under-review journal paper.

I also would like to thank my committee members Dr. Uma Jayaram and Dr. Sankar Jayaram for their advices and comments on this thesis.

Secondly, I want to thank all of my lab mates, especially Qize Le and Bryant Hawthorne. Qize has provided me many valuable comments and thinkings on my research and the published paper. For Bryant, he sacrificed lots of spared time on polishing my English of this thesis in the final stage. And for all the other lab mates, we are always a big family of Collective Systems Lab.
In the end, I want to thank my country and all my friends no matter at China or U.S., especially Chong Ren for your trust and respect. I have a really wonderful time being with all of you.
Open-source processes are based on the paradigm of self-organized communities as opposed to traditional hierarchical teams. These processes have not only been successful in the software development domain, but are increasingly being used in the development of physical products. In order to successfully adapt open-source processes to product realization there is a need to understand how open-source communities self-organize and how that impacts the development of the products. Towards the direction of fulfilling this need, we present an analysis of an existing open-source community involved in developing a web-based content-management platform, Drupal. The approach is based on the analysis of networks using techniques such as social network analysis, degree distribution, and hierarchical clustering. Openly available information on the Drupal website is utilized to perform the analysis of the community. The data is transformed into two weighted undirected networks: networks of people and networks of Drupal modules. Both the structure of these networks and their evolution during the past six years are studied. Based on the analysis, it is observed that the structure of the Drupal community...
has the characteristics of a scale-free network, which is similar to many other complex networks in diverse domains. Key trends in the evolution of the networks are identified. Finally, a predictive model is presented to provide potential explanations for the observed structures and evolutionary trends.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 *Introduction to Mass Collaborative Product Realization* ........................................ 1
   1.1.1 *Traditional Collaborative Product Realization* ........................................... 2
   1.1.2 *Mass Collaborative Product Realization (MCPR)* ......................................... 4

1.2 *Open-source Software (OSS) and OSS Community Concept in Mass Collaborative Product Realization* .................................................................................. 7
   1.2.1 *Open-source Software and Hardware* ............................................................ 7
   1.2.2 *Open-source Processes - A Collective-Systems Perspective* ......................... 10
   1.2.3 *OSS Community in Mass Collaborative Product Realization* ....................... 12

1.3 *Review of Literature and the Identification of Gap* ........................................... 15
   1.3.1 *Review of Literature on Analysis of OSS* ................................................... 15
   1.3.2 *Identification Research Gap* ....................................................................... 17

1.4 *Research Objectives and the Focus of This Thesis* .......................................... 18

1.5 *Introduction to Drupal Community* .................................................................. 21

## 2 APPROACHES ADOPTED IN THIS THESIS

2.1 *Data Collection* ................................................................................................. 24

2.2 *Generation of the Network* .............................................................................. 26
   2.2.1 *Product Network and Social Network* ......................................................... 26
   2.2.2 *Social Network Generation* ....................................................................... 27

2.3 *Techniques for Network Analysis* .................................................................. 29
   2.3.1 *Social Network Analysis (SNA) Metrics* .................................................. 30
   2.3.2 *Degree Distribution and Scale-Free Network* ............................................ 35
   2.3.3 *Hierarchical Clustering* ............................................................................. 36
   2.3.4 *Network Visualization using Images* ......................................................... 37
# Table of Contents

## 3 Analysis of Drupal Network Structure

- 3.1 SNA Metrics Analysis of Social Network (Drupal Community) ............................................. 39
- 3.2 Degree Distribution of the Networks ...................................................................................... 44
- 3.3 Network Structure Analysis through Clustering ...................................................................... 47
- 3.4 Hierarchy Organization Analysis ............................................................................................. 51

## 4 Analysis of Network Evolution

- 4.1 Generation of Snapshots of Networks at Different Intervals .................................................... 56
- 4.2 Evolution of the Network Characteristics .................................................................................. 61
- 4.3 Evolution of Scale-free Property ................................................................................................. 67
- 4.4 Visualization of the Network Structure Evolution ........................................................................ 69

## 5 Analysis of Effort Network --- Commits .................................................................................... 76

## 6 Simulation of Drupal Network Evolution ..................................................................................... 80

- 6.1 Details of the Models .................................................................................................................. 81
- 6.2 Simulation Results ....................................................................................................................... 87

## 7 Closure ............................................................................................................................................. 92

- 7.1 Summary of the Thesis ............................................................................................................... 92
- 7.2 Answering the Research Questions ............................................................................................. 94
- 7.3 Future Work ................................................................................................................................ 95

## References .......................................................................................................................................... 97
LIST OF TABLES

Table 1 - Difference between Traditional and Mass Collaborative Product Realization ... 6
Table 2 - Characteristics of the People and Project Networks .......................................... 40
Table 3 - Highlights of Drupal Network Evolution .......................................................... 62
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Traditional Collaborative Product Realization [3]</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Mass Collaborative Product Realization</td>
<td>5</td>
</tr>
<tr>
<td>Figure 3</td>
<td>The Approach Used to Analyze the Structure and Evolution of Open-source</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Communities as Networks</td>
<td></td>
</tr>
<tr>
<td>Figure 4</td>
<td>Illustration of the Data-gathering Step</td>
<td>25</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Example of a Bipartite Network and the Two Derived Networks</td>
<td>28</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Different Types of Network Structure</td>
<td>34</td>
</tr>
<tr>
<td>Figure 7</td>
<td>People and Project Network Structure</td>
<td>42</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Degree Distribution of Two Types of Nodes in the Bipartite Network</td>
<td>45</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Degree Distribution of Projects (left) and People (right) Networks</td>
<td>46</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Adjacency Matrix Image of the Bipartite Network</td>
<td>47</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Two Hypotheses of the Curve in Figure 10</td>
<td>49</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Adjacency Matrix Image of People Network before and after Clustering</td>
<td>51</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Adjacency Matrix Image of Project Network before and after Clustering</td>
<td>51</td>
</tr>
<tr>
<td>Figure 14</td>
<td>The Scaling of C(k) with k in Different Real Networks [90]</td>
<td>52</td>
</tr>
<tr>
<td>Figure 15</td>
<td>The Scaling of C(k) with k in People and Project Networks of Drupal Community</td>
<td>54</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Number of Participants and Modules at Different Time</td>
<td>57</td>
</tr>
<tr>
<td>Figure 17</td>
<td>People Network Structure Evolution</td>
<td>58</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Project Network Structure Evolution</td>
<td>61</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Mean Degree of Project and People Networks with Respect to Time</td>
<td>63</td>
</tr>
</tbody>
</table>
Figure 20 - Degree Centrality of Project and People Networks with Respect to Time .... 64
Figure 21 - Average Density of Project and People Networks with Respect to Time ...... 65
Figure 22 - Clustering Coefficient of Project and People Networks with Respect to Time
........................................................................................................................................... 66
Figure 23 - Evolution of the Power-law Coefficients for the People Network .......... 68
Figure 24 - Adjacency Matrix Paradigm Ordered by Time Snapshots......................... 69
Figure 25 - Adjacency Matrix Image of Evolution of People Network ....................... 71
Figure 26 - Adjacency Matrix Image of Evolution of Project Network ....................... 75
Figure 27 - CVS Message from Drupal Community Recording Revisions on Every Module [60] ....................................................................................................................................... 77
Figure 28 - Commits Network Distribution ..................................................................... 79
Figure 29 - Diagram of Simulation Processes Flow ....................................................... 84
Figure 30 - Modeling Data from Drupal ......................................................................... 85
Figure 31 - Degree Distribution of the First Model ......................................................... 87
Figure 32 - Joint Degree Distribution of Drupal and Simulated Network .................... 88
Figure 33 - Degree Distribution of People of Drupal and Simulated Network ............ 89
Figure 34 - Evolution of the Simulation by Model 2 ...................................................... 91
1 INTRODUCTION

1.1 Introduction to Mass Collaborative Product Realization

Globalization is a process in which regional economics, societies and cultures are integrated through international networks of communication. We are in the third phase of Globalization as Thomas L. Friedman [1] defined in his book, The World is Flat. Globalization 3.0 is converging workstations with fiber optic cables and work flow software. This is unlike Globalization 1.0 in which countries and governments are protagonists, and Globalization 2.0 in which companies led the way in forming the global integration. In Globalization 3.0, a group of individuals collaborates with each other by sharing information and communications technologies globally. The age of Globalization 3.0 is an innovative paradigm or phenomenon, where numerous groups of individuals work together in the form of communities and loose networks to produce goods and services. This is referred to as Mass Collaboration, by Tapscott and Williams [2]. Instead of rendering user to passively receive information, mass collaboration provides them an interacting platform of sharing information and adding value to the Internet [3]. In the age of mass collaboration, open-source software (OSS) market leads an initial success among various industries. The success of Linux [4], Apache [5], Mozilla [6], Symbian [7], OpenSolaris [8], and other OSS are owed to a large number of participants, with diverse expertise and varied interests. Open-source operating systems such as Linux, Open Solaris and Symbian (operating system for mobile phones) validate a saying that “given enough eyeballs, all bugs are shallow” [9]. Mozilla provides a free Open-Source web browser and Apache is developed and maintained by an open community of
developers. One of the largest encyclopedias, Wikipedia.com, is built by over 10 million collaborating volunteers. It consists of over 15 million articles in 256 languages. Wikipedia is only one of several examples of Web 2.0 Services [10], along with YouTube [11], Flickr [12], Facebook [13] etc. A large number of users in these large communities participate in collaboration over Internet at massive scales. Other than OSS, some of the Open-Source projects involving physical products are following this trend closely. The Open-Source Car Project [14] provides an environment for a community of participants who are interested in developing a simple and innovative car. The Open Prosthetics Project [15] is sharing CAD models for designing prosthetic devices. Apart from these object-oriented platforms, InnoCentive [16] is the world’s first Open Innovation Marketplace, where over 165,000 creative thinkers provided more than 200,000 ideas for 800 challenges in 40 disciplines and 348 were solved only in 2008. More examples that utilize the mass collaboration paradigm are provided by Panchal and Fathianathan [3].

1.1.1 Traditional Collaborative Product Realization

“Throughout most of human history, hierarchies of one form or another have served as the primary engines of wealth creation and provided a model for institutions such as the church, the military, and government” [2]. Traditional collaborative product realization follows a hierarchical mode and it seems that there are no other alternatives. Hierarchical mode mainly lies in the process of traditional product realization. This traditional product realization is based on top-down decomposition where information flows logically from the desired functionality of a product to a design that satisfies the functionality. One of the dominating characteristics is that initial product realization
problems decomposes into several sub-problems. Sub-problems are decomposed until the problem is believed to be a basic problem. Paid academics and topic experts follow their professional disciplinary rules to provide solutions or validations which satisfy the requirements for the problem. In this process, strategic partners play an important role in collaboration and realizing the solutions for problems. In most cases, a company acquires the services of another company which in turn acquires the services of other companies. Of course, this type of information exchanged between strategic partners is various. This means the relationship can be dependent, independent or interdependent. Figure 1 shows the illustration of traditional product realization processes.

![Figure 1 – Traditional Collaborative Product Realization](image)

We can imagine the process of designing a new mobile phone as one example of traditional product realization. When Motorola wanted to develop its new mobile phone series Droid, one of its common strategic partners in telecommunications Verizon, was involved in the development of this phone. In fact, the brand name Droid is a trademark of Lucasfilm licensed to Verizon Wireless [17]. Droid’s operating system runs Google’s
Android, indicating Google as another strategic partner of this project. Design teams from Motorola labs carry on the main development activities which include the appearance, functionality design, and hardware selection etc. For the production of other parts, HTC takes on the production and distribution activities in the manufacturing phase. Sandisk memory cards and Sharp QVGA displays are used in the assembling phase after negotiation between HTC and Motorola. In this example, Motorola divides the problem: design and manufacture a new mobile phone “Droid Series” into several sub-problems. Different strategic partners or development groups take on each of the sub-problems and provide solution to it.

1.1.2 Mass Collaborative Product Realization (MCPR)

According to Ralph Szygenda, CIO of General Motors, “Most big companies are multinationals, not global, and increasingly that’s a big problem for all of us.” Like many multinational coorporations, GM is not only a collection of separate companies but also divided into geographically regional divisions. We are seeking another alternative other than the top-down hierarchical model of traditional product realizations for these kinds of multinationals.

MCPR can be explained by two concepts: open innovation and self-organized communities with large numbers of participants. On the other side of innovation, close innovation paradigm is an approach that is fundamentally focused and fits well with knowledge environments of the early twentieth century [18]. Unlike closed innovation, open innovation refers to a paradigm that many creative ideas from companies or individuals flow into the market for the purpose of product development. Competition and collaboration are two basic modes of open innovation. In the competition-based
mode, there is no collaboration between participants. Participants take part in one problem and provide solution for it based on their professional knowledge and expertise. Rewards for the solutions are one of their motivations. As a complement to InnoCentive, the attractive rewards which are ranging from one thousand to one million dollars, are one of the propellants for that incredible number of completed challenges. In the collaboration-based mode, participants join to their desired problems which are known by the whole community. Community members offer and amend solutions any time at their will. Compared to traditional top-down hierarchical mode, the self-organized community does not follow the restrictions of hierarchy. Participants do not have to be assigned a particular sub-problem and are free to work on sub-problems that they want. Participants have many different motivations for “jumping in”, ranging from fun and altruism, to achieving something that is direct value to them [3].

![Figure 2 – Mass Collaborative Product Realization](image)

A publish and subscribe (pub/sub) mode is usually adopted through the information flow of mass collaborative product realization. Publish and subscribe mode is a messaging paradigm where a publisher sends messages and subscribers receive messages
through the community platform. The community platform characterizes messages into classes, without knowledge of what subscribers there may be. Subscribers receive specific interest class of messages, without knowledge of what publishers there are. There are plenty of advantages in using the ‘pub/sub’ mode as information flow in mass collaborative product realization. The message filtering function of ‘pub/sub’ mode provides two common forms of selecting message processes, topic-based and content-based systems. The decoupling of publishers and subscribers can allow for greater scalability and a more dynamic network topology. In the traditional client-server paradigm, the client cannot post or receive messages while the server is not running. However, publishers are loosely coupled to the subscribers even without the existence of each other. ‘Pub/sub’ mode facilitates mass collaborative product realization in the following four aspects: a) formation of the community, b) online description of the product realization problem c) online submission of solutions and comments and d) notification of changes. The differences between traditional and mass collaborative product realizations are summarized in Table 1.

<table>
<thead>
<tr>
<th>Organizational Structure</th>
<th>Traditional Product Development</th>
<th>Mass Collaborative Product Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants</td>
<td>Guided by higher level organizational goals</td>
<td>Self-interested</td>
</tr>
<tr>
<td>Product</td>
<td>Systematically designed in a targeted manner</td>
<td>Product evolves over time</td>
</tr>
<tr>
<td>Task Assignment</td>
<td>Task assigned to participants</td>
<td>Participants choose to work on tasks on their own</td>
</tr>
<tr>
<td>Information Flow</td>
<td>Direct hierarchical flow</td>
<td>Publish and subscribe</td>
</tr>
<tr>
<td>Product Release</td>
<td>Product release after major revision</td>
<td>Product always under continuous development</td>
</tr>
</tbody>
</table>
MCPR provides opportunities to develop successful products in the marketplace, which requires various types of competencies. An organization or a single individual is not capable to satisfy all the needs in the marketplace. Organizations often cooperate through partnerships with other organizations. Individuals, on the other hand, rarely have opportunities to contribute their knowledge as a single unit other than through organizations. MCPR serves as a proper platform offering opportunities to organizations and individuals. First of all, MCPR turns ideas and technologies to products. Hundreds of thousands of good ideas are generated everyday but only a few of them succeed because of the lack of a proper team and a lack of resources. In MCPR, individuals who have innovative ideas can turn them into reality by working with other talents. Second, through MCPR, individuals also play an important role in identifying and matching new technologies into many areas where they can be appropriately used. Finally, MCPR helps solve difficult problems rapidly. The design process also involves analyzing alternatives generated from multiple perspectives and making decisions. Mass collaboration provides an opportunity to solicit ideas and solutions from large groups of people. The probability of solving the difficult problems is hence increased as a result of many minds trying to solve it.

1.2 Open Source Software (OSS) and OSS Community Concept in Mass Collaborative Product Realization

1.2.1 Open-source Software and Hardware

During the past two decades, Open-source processes have gained significant popularity in the software development domain. Various successful products such as
Linux, Apache, and Mozilla have shown that open-source processes can be as successful as the processes followed by traditional organizations. The concept of open-source has not only been used in the software development domain but also has been recently implemented in physical product development. Examples include Open-Source 3D printers [19], electronics prototyping platforms [20], cell phones [21], cars [14, 22], prosthetics [15], machine tools, robots, and other socially-relevant design projects [23]. In physical product development, open-source refers to the openness of information such as design details, schematics, CAD models, bills of materials, associated software, etc. The success of Open-source processes in physical product development is driven by the fact that physical products are information products during the design phase, and their development is accelerated by reducing the cost of 3D printing capabilities.

Open-source processes emerged in software products earlier than in physical products because software has characteristics particularly suitable for it. The development of both software and physical products can be divided into four phases: design, manufacturing, distribution, and upgrade. In the design phase, both software and physical products can be viewed as information-based products [24] defined by requirements, functions and detail designs, which can be recorded electronically. In the manufacturing phase, software products can be manufactured (programmed) by individuals using computers, while physical products need to be manufactured by specific physical tools and machines. In the distribution phase, software products can be shared through the Internet at (almost) zero cost, but physical products need to be transported from one site to another. In the upgrade phase, software can be upgraded by simply re-compiling the codes whereas physical products need to be re-manufactured. By comparing software and physical
products in the four product development phases, it is clear that software products are easier to manufacture, distribute and upgrade than physical products, even by individuals without many resources. Hence, the differences between software and hardware products are clear. However, there are still fundamental similarities in the way open-source processes are utilized both in software and hardware. We are particularly interested in the similarities in the way open-source communities emerge and the products gradually evolve.

Further, emerging technologies are making open-source processes more applicable to physical product development. For example, with the increasing availability of 3D printing capabilities, open-source processes have also started gaining popularity in physical product development [25]. In the design phase, physical products are also information-based products whose documents can be easily shared by individuals. So the open-source processes can be applied during the design phase, examples include Open Source Car [14, 22], and Open Prosthetics Project [15]. With the development of the rapid prototyping technology, open-source processes can also be extended into product manufacturing. A 3D printer is a convenient tool for individuals to prototype physical products using CAD models. Some physical products such as electronic hardware can be manufactured directly from electronic design documents. These designs can be downloaded from the Internet. In these cases, the manufacturing and upgrading phases can also be carried out through open-source processes. In a recent article, Anderson [26] projects that open-source will revolutionize the way in which innovative products are designed and developed. Given the increasing adoption of open-source as a new
paradigm for product realization, it is becoming important to understand its underlying dynamics.

1.2.2 Open-source Processes - A Collective-Systems Perspective

Open-source processes are significantly different from traditional product development processes because they are based on bottom-up design by self-organized communities as opposed to top-down design by hierarchical organizations. They are driven by participants who choose their activities based on their own goals and interests instead of being driven by top-down hierarchical control as in the case of traditional product development. Open-source products are always under continuous development and evolution. Similarly, the open-source communities are under continuous evolution. Hence, traditional hierarchical models of products and processes prevalent in systems engineering are limited in providing insights about the analysis and design of open-source processes. Further insights can be gained by viewing open-source process as collective systems.

Collective systems, according to Tumer and Wolpert [27]’s definition, consist of interacting “agents” who strive to maximize their private utility through their local behaviors. The system’s global performance measured in terms of a world-utility function. In open-source processes, the participants may be considered as agents with their own goals. These goals may range from the need to achieve certain functionality to gaining recognition [28]. Interactions between the agents are through the (software or hardware) products they work on. The world-utility function can be defined in terms of the development of the product. Tumer and Wolpert [27] and Namatame [29] highlight that collective systems are associated with two types of problems: forward (analysis)
problem and inverse (design) problem. The forward problem involves finding the system-level performance based on a set of agent behaviors and their interactions. On the other hand, the inverse problem is concerned with the determination of “suitable” agent behaviors and interactions to achieve desired system-level performance. Recently, significant efforts have been devoted by economists and social scientists on the forward problem related to open-source processes, with the goal of understanding the private utilities and interactions of agents in successful projects [28, 30-33]. Their focus is on answering questions such as: Why do people contribute for free? What motivates people to contribute? Why is open-source successful? However, the inverse (design) problem has received little attention. The inverse problem is particularly important for the successful implementation of open-source processes within product development, and hence, of relevance to design engineering research community.

One of the key factors in the inverse (design) problem in open source is the structure and evolution of the community and their effect on the evolution of products. It is known that “teams with the same composition of members can perform very differently depending on the patterns of relationships among the members” [34]. The effect of organizational structure on the product is well recognized in traditional product development, [35]. According to Conway [36], “any organization that designs a system (defined more broadly than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure”. Hence, the organizations strive to align the organizational structures with the product structures. However, in the case of open-source processes, no organizational structures are imposed at the beginning of the process. The structure of the organization evolves as new
participants join and collaborate with existing participants. The collaboration between different participants is based on the product structure and is driven by the dependencies between subsystems, implying the effect of product structure on community structure. Hence, in open-source processes, the products and communities undergo interdependent co-evolution. In order to successfully utilize open-source processes for product realization, we believe that the knowledge of this interdependent co-evolution is crucial. The knowledge can be gained by understanding a) the structure and evolution of communities, b) the structure and evolution of products, and c) the interdependence between structures and evolution of communities and products.

1.2.3 OSS Community in Mass Collaborative Product Realization

Weber states in his book ‘The Success of Open Source’ that “Open-source is a real-world, researchable example of a community and a knowledge production process that has been fundamentally changed, or created in significant ways, by Internet technology” [37]. As we discuss in Section 1.1.2, mass collaborative product realization as a paradigm enables organizations and individuals to develop innovative products. An OSS community is a self-organized platform where individuals work together to provide modifications to the product. Several successful OSS projects and their communities such as Linux [4], Apache [5] and Mozilla [6] attract attentions and are worth studying. Therefore, one key question left is why individuals voluntarily allocate their time, effort and mind to the open-source process rather than to something else? The answer to this question can be various but one is inevitable: the motivation of individuals.

We can see open-source process as amusement for enthusiasts or a game playing by groups of hobbyists. This is a labor of love. Hobbyists prefer to follow the motivation and
doing what they want to other than personal satisfaction. A survey by Boston Consulting Group (BCC) results four characteristic groups of developers with different insights on Open-Source process [37]. About one third of “believers” said that they were strongly motivated by the conviction that source code should be open. About a fifth of “professionals” admitted that working on Open-source helped them in their jobs. About one fifth of “skill enhancers” emphasized the learning and experience they got from open source programming. And one quarter of “fun-seekers” did it mainly for intellectual stimulation. Individual’s motivation on open source can be yielded in the following four elements.

First of all, solving challengeable programming problems serve to be an aesthetic pursuit for developers in the OSS communities. Torvalds, who initiated the development of the Linux kernel, has consistently said that his primary motivation for working on Linux is “the fun of programming”, which he often equates with artistic creativity. For primary developers, this kind of motivation about creativity is a good bargain to them. Showing the world an individual’s work and how creative one is the best encouragement for elementary developers among the community. Of course, one thing one needs to know is that good and bad comments on the work come at the same time. Everyone is growing up through humbling experience.

Second, for some of the developers, embedding in a battle with “enemies” is one of their “born duties”. Proprietary software such as Microsoft Windows, Adobe Flash Player, PS3 OS, iTunes, Adobe Photoshop, Google Earth, Mac OS X, Skype, WinZip and some versions of Unix are impenetrable products in open source developers. The BCC survey concludes that 11.3 percent of the developers said that “to beat proprietary
software” was a principal motivation for their work; and 34.2 percent chose “code should be open” as a primary motivation [38]. However, the “enemy” is not an ideological villain. It is a technical and business practice villain and that is what the conflict is about [38].

Moreover, everyone is an egoist under special conditions and environments. The raising egotism of individuals is a powerful motivating force for open source developers. Winning respect from others in the OSS community by presenting their own innovative and creative work is a source of satisfaction and self-recognition for the developers. Within this mechanism, ego as a driving force is openly acknowledged and accepted. However, it is criticism that makes the OSS better and better between different versions. The norm in the open source community is to criticize the code, which means that one person’s ego boosting is built on another person’s insult. Similar to the egotism, reputation is another issue. In the open source world, there is no authority, which means any developer can earn his/her reputation once showing their ideas and products or correct criticism on other’s works.

At last, shared identity and belief system is firmly established among OSS communities. Levy et al. [39] described several key tenets in his book, which continue to characterize the open source community to a surprising degree. Statements from Steven such as “access to computer should be unlimited” and “information should be free” indicated the future of open source process in 1984. Another important and foundational aspect of shared belief is the notion that personal efficacy not only benefits from, but positively requires, a set of cooperative relationships with others [38]. On the other hand,
doing new things that challenge the skills of others is a process of learning and development for individuals as well as a contribution to the whole community.

1.3 Review of Literature and the Identification of Gap

1.3.1 Review of Literature on Analysis of OSS

Existing literature on open-source processes is primarily focused on open-source software (OSS) development because of highly developed processes, large number of communities, and significant amounts of data on OSS development. A general discussion of the factors affecting the success of OSS development is presented by Weber [38]. OSS is a public good provided by volunteers – the “source code” used to generate the programs is freely available to read, use and modify [40]. An OSS project is typically initiated by an individual or a small group with ideas which can realize their intellectual, personal, or business interests [41]. Various researchers have presented empirical and quantitative studies on the structure of OSS communities based on the data from existing OSS projects. Raymond [9] describes the Linux development community as a “Bazaar”. Cox [42] presents initial thoughts of “town councils” structure in OSS community based on Linux 8086 project. The author conceptually illustrates the community structure for Linux 8086 project. Weber [38] discusses different types of organization structures in various OSS projects. For example, the community structure of the Linux project reflects a pyramid structure whereas the community structure of the BSD project is represented as concentric circles. The structures concluded by Weber are based on direct observation of communities without rigorous mathematical analysis. Crowston and Howison [32] discuss community centralization in OSS development communities by analyzing data
from the bug-tracking system in SourceForge [43]. The authors demonstrate that the community centralization or decentralization is not a characteristic of OSS projects. Crowston and Howison [44] later analyze hierarchy and centralization of the OSS communities of Apache, Savannah and SourceForge [43] by employing social network analysis (SNA) measures. They conclude that large projects are less centralized and hierarchical, as compared to smaller projects. Xu and Madey [45] discuss role distribution and degree distribution in the SourceForge community. Xu et al. [46], and Gao and Madey [47] study topological properties of open-source communities, including degree distribution, diameter, clustering coefficient, centrality and component distribution by modeling OSS communities as complex social networks. They also observe small-world [48] and scale-free [49] network properties in the SourceForge community. Xu et al. [50] present the structure of OSS communities by calculating the modularity of the network, which is defined as the fraction of edges within communities minus the expected value of the same quantity if edges fall in a random network, and analyzing the groups that exist in the SourceForge [43] network.

The studies discussed above are focused on analyzing the community structures. Some efforts have also been carried out on the evolution of the communities. White et al. [51] introduced the analysis of social structure over time using snapshots of data. Nakakoji et al. [52] discuss the evolution of communities in the form of role changes of the members in OSS communities, and conclude that there are two factors determining the evolution of OSS communities: the existence of motivated members, and the social mechanisms of communities. Weiss et al. [53] trace the evolution of a community by taking snapshots of its membership at regular intervals and establish a major hypothesis.
that OSS communities grow through a process of preferential attachment. De Souza et al. [54] represent a framework for software modules and software developers, and study software project communications at two points in time. The authors analyze the movements of developers across different modules of software systems. Howison et al. [55] investigate the structure of OSS development communities over time using snapshots of data to understand the dynamics of social structures in OSS development communities. They examine three properties of the social structures, namely, centralization, network center, and stability of participation. Wiggins et al. [56] analyze the dynamics of OSS development communities and find a variation in communication centralization and decentralization in the OSS development communities. Open-source software development is a special case of mass-collaborative product development [3]. Panchal [57] presents an agent-based model to model the evolution of products in such bottom-up processes. Panchal later extends the model to explore the co-evolution of communities and products [58]. Le and Panchal [59] study the effect of product architecture on the evolution of products in mass-collaborative processes.

1.3.2 Identification of Research Gap

The existing studies introduced above seem to cover the major analysis of OSS communities, which includes the snapshot structure of community, employing social network analysis measures, topological properties such as diameter, cluster coefficient and centrality etc. However, they are limited in analysis of open-source communities because of the lack of the following aspects:

- Simultaneous analysis of structure and evolution of the community,
- Comprehensive analysis of the different aspects of the community structure,
• Predictive models to explain the evolutionary behavior of the communities, and

• Integrated analysis of the evolution of communities and the products.

We perform a comprehensive study of the community structure and evolution in Chapter 3 and 4. The study is based on the data from Drupal [60], which is an open-source software for building community-based websites. The reason for studying Drupal is that there is freely available data, detailed documentation, and a highly developed community associated with this project. Besides, Drupal is widely used as a basic framework for web development and is a very successful community-based website building tool. The objective is to understand the evolutionary characteristics of open-source projects that span both software and physical products. The study of the OSS project will lead to a) fundamental knowledge which can be applied to both software products and physical products based on their commonalities in the design phase, and b) new techniques enabling individuals involved in manufacturing, upgrade and distribution phases.

1.4 Research Objectives and the Focus of This Thesis

In Section 1.2, we have discussed the Open-source software (OSS) from a collective-system perspective and OSS community from a mass collaborative perspective. We have also explained that motivation is one of the most important reasons individuals spend their time and effort to the community rather than something else. OSS communities bring together individuals into an integral organization with innovations and creativities, which presents the diversity of OSS community population. And the structure of OSS community influences the product characteristics and properties to some extents.
Therefore, when groups of people are involved in the process of production, it is important to first figure out their actions and the following essential problems of community.

**Q1: What is the structure of open-source community?**

The hypothesis to validate this question in this thesis is that community structure properties can be reflected from some social network metrics. Newman and Girvan present a set of algorithms for investigating the community structure in networks [65]. They demonstrate that the algorithms are highly effective at discovering community structure in both computer-generated and real-world network data. However, employing and then analyzing the social network metrics seems a better way for the reason that this kind of technique is mature and feasible. Xu et al. [46], and Gao and Madey [47] study topological properties of open-source communities, including degree distribution, diameter, clustering coefficient, centrality and component distribution by modeling OSS communities as complex social networks. Open-source is a real world and researchable example of communities have been fundamentally created in a significant way [38]. We propose a series of social network analysis methods for analyzing the structure of the Drupal [61] community, which serves to be a good representative example for other open source communities.

**Q2: What is the evolutionary behavior of these communities?**

The approach of answering this question is similar to that of Q1. We believe that social network analysis methods are appropriate to explore the properties of the OSS community. Therefore, evolutionary behavior of OSS communities can be separated into several snapshots as the community is formed. Kumar et al. made efforts on excavating
Our research on evolutionary behavior of OSS communities focuses on three aspects. First, the analysis of the evolution of network characteristics involves the social network metrics comparison and investigation of some snapshots of the community network. Second, the evolutionary data of degree distribution is aimed at verifying the scale-free properties of the OSS community network. And last, the innovative visualization method help in analyzing the network structure evolution.

**Q3: What are the underlying local behaviors that result in their evolutionary behavior?**

Our hypothesis is that preferential attachment mechanism underlies the growth of most of the OSS community network. Preferential attachment can be simply explained that new person connects to the old ones with probability proportional to their degree. It is a key concept in the field of scale-free networks. In order to validate the hypothesis and to predict the underlying dynamics of the network growth, we present a computational model whose input is the local behavior of the nodes (people) and the output is the structure of the networks at various time-steps. Existing literature on preferential attachment is focused on 1-mode networks. In this thesis, we extend the model to a bipartite network. This model is advanced because the mechanism is based on the observation of the micro-behavior of each developer in the OSS community and it included the strategy of selecting their preferred group to join. The result of the simulation and the validation of this hypothesis are presented in the Chapter 6.
The goal of this thesis is to take a step towards answering these questions. The developed insights can be used to address the inverse (design) problem in open-source processes. The approach used here is to focus on a specific open-source project, and to analyze the structure and evolution of the associated community. A predictive model is proposed for the underlying local behaviors of individuals that possibly explain the resulting structure and evolutionary behavior of the community.

1.5 Introduction to Drupal Community

Drupal is an open-source content-management system, which is used to create community-based websites. The standard release of Drupal, known as the Core, contains basic features of the content management system. The Drupal framework consists of a core and a large number of modules developed by users using open-source techniques. Drupal was also designed to allow customized modules to be added to extend Drupal’s core capabilities. Drupal has been chosen for this study because of its strong community and the easy access to participants and modules data for analysis. Drupal started in 2000 and currently has a large community of contributors. There are different types of users who interact with the Drupal community. Passive Users are users who only download and use the software but do not contribute to the code. This users group forms a large user base in the Drupal community but it is not considered in this social network analysis. Active Users contribute to the discussion board, suggest new features and identify bugs but do not modify the code. Co-developers modify the codes, add fixed bugs, and add new features to the software. Core developers contribute largely to the core of the Drupal code and coordinate co-developers’ work. Project leaders are the project administrators who manage the direction of the entire project. For the analysis presented in this thesis,
only co-developers, core developers, and project leaders are considered because the
database of Drupal only records the participant who develop the module or at least
modify the code once. Activities such as the identification of bugs and contributions to
discussions on the bulletin boards are not considered in this thesis.

In the case study of Drupal community, the network created for the analysis of Drupal
community consists of two types of entities: a) the people (participants), and b) the
projects (modules) they contribute to. The analyses discussed in Chapter 2 are carried out
for the Drupal data. The approaches discussed in the next chapter are utilized to analyze
the structure and evolution of Drupal community [61]. The analysis of the structure of
networks is discussed in Chapter 3 and the analysis of network evolution is presented in
Chapter 4.
2 APPROACHES ADOPTED IN THIS THESIS

The approaches adopted in this thesis are based on network analysis. The communities are modeled as social networks, defined by participants connected by collaboration links. Social network analysis approaches are referred to the steps, processes and methods which help researchers to investigate the natural properties of a social network and its community. Power-law distributions are reported on the existence of topology of large networks defined by the World-Wide Web [49]. Border shows the world-wide web structure from a connectivity point of view and demonstrates the existence of a large strongly connected component [63]. The discoveries above are from a static point of view. In fact, most of the real-world communities are evolving in nature. Social network analysis has been used to analyze evolution of diverse systems, such as author and paper networks [64], online communities on websites such as Yahoo and Flickr [62], and other OSS communities. The approach used in analyzing Yahoo and Flickr [35] first measures the reciprocity, density and degree distribution with an associated time stamp of both the networks. Then some network information was summarized and speculations were made based on the previous results. Finally, simulations were run as a confirmations for the speculations and used to prove some of the properties of both the networks. Similarly, the author and paper networks [34] is also following those steps but focusing more on the structure of the network. The overall approach adopted in this thesis is highlighted in Figure 3 beginning with a static analysis and then the evolution of the network.
2.1 Data Collection

A social network is a set of interrelated social entities. Social network analysis is defined as an investigation of the social network properties through the relationships between the entities in the network. Therefore, a data collection process must at least include the entities of the network and their relationships. The primary data required for this research is a table including the information of two aspects: participants and their product modules.

In the first data collection step, raw data about the participants and the product modules they contribute to are extracted from the database. The raw data are records of
initial information dug from the social network. And it is used to derive the information about relationships between individuals and their related modules. The raw data can be in the form of a simple table which shows the participants and the modules. In order to study the evolution of the community network, the following information is collected: a) the joining dates of individuals, b) the dates of individuals' first contribution, and c) their contributions to different modules.

Figure 4 (left) is a sample information table from www.drupal.org, which includes information about the participants' activities on a module named “Activity”. This committers table contains information about the user name ("User" column), the first and last time each user made a revision of this module ("First commit" and "Last Commit" columns), and the number of times each user revised this project (the “Commits” column). Each Drupal module has an information table similar to Figure 4 (left). After collecting the information from all the modules, the overall information table as shown in Figure 4 (right) is generated. For further analysis of the data, the overall information table usually includes a unique ID number for each separate module and individual. Some of the databases, such as SourceForge, provide the information table which consists of the name and its unique ID for entities in the network. However, in most cases, an automatic distributing unique ID program is developed at the end of the data mining process.
2.2 **Generation of the Network**

2.2.1 **Product Network and Social Network**

In the previous chapter, we introduced the first step of social network analysis processes – data collection. At the end of data collection, an overall information table was assembled from several “Committers table” of all modules in Drupal community. In network science, this “Committers table” is referred as “conventional” sociological data, which is always contain cases, subjects or observations in the rows of array, and scores on attributes, variables or measures in the column. Many network analysts apply techniques to the conventional data the same as other formats of data. However, rather than thinking of the attribute and measures between two entities, analyst would be more interested in seeing the structure of connections. Entities in the network are described in their relationships, but not their attributes. The main difference between conventional and network data is that conventional data focus on entities and attributes but network data on entities and relations.

A network is an interconnected system of things or people. This means the entities are usually connected to each other by some ties. Some terminologies need to be introduced before the analysis can begin. Nodes and edges are used to represent entities and relations in network analysis. In graph theory, a network is modeled as a graph which consists of a set of nodes and edges. A graph whose edges are without direction (an unordered pair of nodes) is called undirected graph. In the most cases, social networks are modeled as an undirected graph. We can define different types of nodes to represent different entities in the social network. An edge connected to two nodes in a network means that these two nodes are interacting with each other. Different strengths of connections between these
two nodes are needed to be identified in a social network. Strong connection is used on nodes with similar background, experience, interest, goals and resource. For example, strong connection exists between two good friends. Weak connections, by contrast, are formed under limited interactions. For example, a person you met in the supermarket may have a week connection with you rather than a strong connection. In social network, we can use weighted networks to signify the strength of an edge between two nodes. A weighted network is a network which there are values assigned on the link between two nodes. The next chapter presents how to generate a social network by extracting information from the conventional data.

2.2.2 Social Network Generation

From the information table generated in data collection, community networks are created to model the relationships within the community. First, we build a network consisting of two types of nodes – “people” and “projects”. The development of each module is a project in Drupal. Hence, each module represents a “project” node and each participant represents a “people” node in the network. Such a network is called a bipartite network, $G = \{S1 \cup S2, E\}$, which consists of two disjoint sets of nodes $S1$ and $S2$ and a set of edges $E$ such that each edge in $E$ connects a node in $S1$ to a node in $S2$. An example of a bipartite graph is shown in Figure 5, where $S1 = \{a, b, c, d, e, f, g\}$ and $S2 = \{1, 2, 3, 4\}$. In the case of the open-source software network, assume that $S1$ and $S2$ represent nodes of “people” and “projects” respectively.

Links in the network connect a person with a project. Hence, a link represents a person working on a module. The bipartite network can be weighted or binary. If a binary matrix is used, then the links only represent the presence of relationship between people.
and projects. However, if a weighted network is used, the weights on the links can be used to represent the amount of effort invested by the participants on corresponding modules. An indicator of the amount of effort is the number of commits by a person to a module.

In Figure 5, a binary bipartite graph is illustrated. The bipartite graph $G$ can be transformed into two weighted undirected graphs $G_1 = \{S_1, E_1\}$ and $G_2 = \{S_2, E_2\}$ consisting of people ($S_1$) and projects ($S_2$) respectively. Figure 5 provides an illustration of graphs $G_1$ and $G_2$ derived from a bipartite graph. Two people in $G_1$ are connected by an edge if both of them share at least one project. Similarly, two projects in $G_2$ are linked if they have at least one common participant. The weights associated with edges $E_1$ represent the number of projects shared by a set of people. Similarly, the weights in graph $G_2$ represent the number of common participants shared by projects.

The graphs $G_1$ and $G_2$ are used for analysis discussed in Section 2.3.

![Figure 5 - Example of a Bipartite Network and the Two Derived Networks](image)

The graphs can also be represented in matrix form as shown in the Figure 5. An adjacency matrix of a network with $n$ nodes is an $n \times n$ matrix, where an element $a_{ij}$ denotes the weight on the edge from node $i$ to node $j$ and $0$ denotes no connection.
between nodes i and j. The diagonal elements are conventionally set to 1. From the overall information table, we can obtain the matrix M shown in Figure 5 by adding 1 to the element $m_{ij}$ in M if there is a connection, where i is the module ID and j is the person ID. Once the matrix M is obtained, matrix $M_1$ and $M_2$ can be derived from M by following the description above.

At first, we mark down the column ID of the non-zero element in the first row and get a set: $X = \{j_1, j_2, j_3, \ldots j_k\}$. Then set element $b_{xy}$ to in the target adjacency matrix, where $xy$ are any combinations of the two numbers in the set X. At last, repeat the first and second steps for r times, where r is the maximum row number in the M matrix. Similarly, we can obtain the other adjacency matrix by marking down the row ID and repeating the maximum column number. For example, we aim at developing the matrix $M_1$ from the matrix M. In the first row of M, the non-zero elements are in the column 1 and 4. We get the set $X = \{1, 4\}$. Then we set the elements $b_{11}, b_{14}, b_{41}$ and $b_{44}$ to 1. After repeating the process 4 times, we can obtain matrix $M_1$.

2.3 Techniques for Network Analysis

Adjacency matrix serves as a basic input for the network analysis process. After creating the adjacency matrix, the network properties can be explored by using different approaches. In this thesis, the following approaches are used to determine the characteristics of the networks: social network analysis metrics, degree distribution, clustering and re-sequencing, and network structure image method. These approaches are discussed in Section 2.3.1 through Section 2.3.4.
2.3.1 Social Network Analysis (SNA) Metrics

As discussed earlier, the OSS community is modeled as a social network comprising of participants, projects, and the relationships among participants and projects. In order to characterize the key features of the OSS network, we use Social Network Analysis (SNA) metrics [65]. SNA is a theoretical and methodological paradigm for examining complex social structures [66]. Social networks can be either directed or undirected. The links may also carry weights to represent the strengths of the relationships between nodes [67]. The following SNA metrics are used in this thesis: degree, clustering coefficient, diameter, shortest path, density, connectedness, and degree centrality.

a) **Degree** is the number of nearest neighbors of a node [68]. In an undirected graph, the degree of a node \( v \) is the number of edges incident with \( v \) and is denoted by \( \text{deg}(v) \) or \( k_v \) [69]. The degree distribution, \( P(K) \), of a network is defined as the fraction of nodes in the network with degree \( k \) [70]. In a bipartite network, two degree distributions corresponding to both types of nodes are important. The joint degree distribution of a network, \( P(K_1, K_2) \), represents the probability that a randomly selected edge is connected to nodes with degrees \( K_1 \) and \( K_2 \) [71, 72]. The joint degree distribution is different from the conditional probability \( P(K_2 | K_1) \) which measures the probability that a given node of degree \( K_1 \) is connected to a node of degree \( K_2 \).

b) **Clustering** coefficient is the probability that two nearest neighbors of a node are also the nearest neighbors of one another [68]. It usually serves as an indicator of the connectivity which means the probability that nodes tend to cluster together. A cluster in a network is a group of nodes connected to each other by some links.
The nodes in the same cluster are connected by a link, but no link between nodes in different clusters [46]. The clustering coefficient reflects the “cliquishness” of the mean closest neighborhood of a vertex. The clustering coefficient of a social network is the average of all the clustering coefficient of the nodes. The general function of clustering coefficient of a node:

\[ C_i = \frac{2 \sum e_{jk}}{k_i(k_i - 1)} \]  

(1)

where \( \sum e_{jk} \) is the sum of the links between neighbors of node i, \( k_i \) is the number of neighbors of the node i.

c) Shortest path is the shortest path of nodes and edges that links two nodes in a network [73]. In graph theory, the shortest path between two nodes is defined as the distance of two nodes. Diameter is the largest distance between any two nodes of a connected network [69]. The diameter of a network tells us how “big” the network is. Calculating the diameter is computationally expensive, so the alternative way is to calculate the average shortest path. The average shortest path can describe whether there exists the “small world phenomenon” [74] within the networks. The average diameter approximation, developed by Newman and Watts, can be computed by:

\[ d = \frac{\log (N/z_1)}{\log (z_2/z_1)} + 1 \]  

(2)

Where, \( d \) is the diameter, \( N \) is the total number of nodes in the network, \( z_1 \) is the average number of neighbors 1 link away, \( z_2 \) is the average number of neighbors 2 links away [46].
d) *Density of a network* is the average proportion of links incident with nodes in the network [75]. If we are comparing two populations, and we note that there are many actors in one that are not connected to any other ("isolates"). In the other population most actors are embedded in at least one dyad -- we would likely conclude that social life is very different in the two populations. The density of a network ranges from 0 (if there are no links present) to 1 (if all possible links are present). A network with density of 1 is also called a complete network. The density of a network with n nodes and m links is:

\[
density = \frac{2m}{n(n-1)}
\]  

(3)

e) *Krackhardt’s Graph Theoretical Dimension (GTD)*: Social network structure of a population is always display “texture” (Figure 6a) when embedding the entities in neighborhood, cluster and groups. A very common form of embedding of actors in structures, though, does involve unequal rankings. Hierarchies, in which individuals or sub-populations are not only differentiated, but also ranked, are extremely common in social life. Krackhardt’s GTD of hierarchy is a definition of the meaning of hierarchy, and developed measures of each of the four component dimension of the concept, which is provided by Krackhardt in 1994. Krackhardt develops index numbers to assess the extent to which each of the four dimensions deviates from the pure ideal type of an out-tree (Figure 6b), and hence develops four measures of the extent to which a given structure resembles the ideal typical hierarchy.
i) **Connectedness** represents the ratio of the number of pairs in the directed graph that are reachable relative to the number of ordered pairs. The degree of connectedness is then defined as [76]:

\[
\text{Connectedness} = 1 - \frac{V}{N(N-1)/2}
\]

(4)

where \(V\) is the number of pairs of nodes that are not mutually reachable, and the maximum number of violation is the total number of pairs of nodes \(= N(N-1)\)

ii) **Hierarchy**: To measure the degree of hierarchy of digraph \(D\), a new digraph \(D_r\) must be created. \(D_r\) is defined as the reachability digraph of \(D\). Each point in \(D\) exists in \(D_r\); moreover the line \((P_i, P_j)\) exists in \(D_r\) if and only if \(P_i\) can reach \(P_j\) in \(D\). If \(D\) is graph hierarchic, then \(D_r\) will have no symmetric lines in it. That is, if the line \((P_i, P_j)\) exists in \(D_r\) then the line \((P_j, P_i)\) does not. A violation to this condition exists every time a symmetric line exists in \(D_r\). The degree of hierarchy, then, is defined as:

\[
\text{Graph hierarchy} = 1 - \frac{V}{\text{Max}(V)}
\]

(5)

where \(V\) is the number of unordered pairs of point in \(D_r\) that are symmetrically linked, and \(\text{Max}(V)\) is the number of unordered pairs of points in \(D_r\) where \(P_i\) is linked to \(P_j\) or \(P_j\) is linked to \(P_i\).

iii) **Efficiency**: In the underlying graph \((G_1, G_2, \text{etc.})\) of each component \((D_1, D_2, \text{etc.})\) of digraph \(D\), there are exactly \(N_n - 1\) links, where \(N_n\) is the number of nodes in the corresponding component \(D_n\). Because fewer than \(N_n - 1\) links is
not possible, violations occur to the extent that more than this minimum number of links is present. The degree of graph efficiency is defined as:

\[
\text{Graph efficiency} = 1 - \frac{E}{\text{Max}(E)}
\]  \hspace{1cm} (6)

where \(E\) is the number of links in excess of \(N_n - 1\), summed over all components, and \(\text{Max}(E)\) is the maximum number of links in excess of \(N_n - 1\) possible, summed over all components.

iv) Least upper bound (LUB): To be a pure out-tree, each pair of actors (except pairs formed between the ultimate boss and others) must have an actor that directs ties to both -- that is, command must be unified. The deviation of a graph from this condition can be measured by counting the numbers of pairs of actors that do not have a common boss relative to the number of pairs that could (which depends on the number of actors and the span of control of the ultimate boss) [67].

![Figure 6 – Different Types of Network Structure](image)

f) Degree Centrality measures the degree of inequality or variance in the network as a percentage of that of a perfect “star” network (Figure 6c) of the same size. For a network \(G=(V,E)\) with \(n\) nodes, the degree centrality \(C_D(V)\) for a node \(V\) is [77]:
The degree centrality of a network $G$ is:

$$ C_D(V) = \frac{\text{degree} (v)}{n - 1} \quad (7) $$

The degree centrality of a network $G$ is:

$$ C_D(G) = \frac{\sum_{i=1}^{n}[C_D(v \star) - C_D(v_i)]}{n - 2} \quad (8) $$

where $C_D(v \star)$ is the highest degree centrality of a node in the network.

$C_D(v_i)$ is the degree centrality of a node $i$ in the network.

Social network analysis tools are used for evaluating the metrics discussed above. The most widely used tools are Structure [78], Gradap [79], UCInet [80] and Network Workbench [81]. Other social network analysis tools are discussed by Huisman et al. [82] and Freeman [77]. UCInet is used for the results presented in this thesis.

### 2.3.2 Degree Distribution and Scale-Free Network

The degree of a node is the number of links connected to this node. The degree distribution is the possibility distribution of all the degrees in the entire network. The only way to obtain the degree distribution of a network is to calculate every degree of all the nodes in the network. The degree distribution reflects the frequency of the degree of nodes in the network. It is commonly believed that the degree distribution is a Poisson distribution, which means the two nodes establish their relationship randomly [46]. However, Barabasi and Albert [49] have found that degree distribution satisfies a power law in many real-world networks [83]

$$ y = \beta x^\alpha \quad (9) $$

where $\beta$ and $\alpha$ are constants and $y$ denotes the number of nodes with degree $x$. 

35
A network whose degree distribution follows a power law is called scale-free network. Scale-free networks have a property that only a few of nodes (called “hubs”) have a high degree, while most other nodes are only connected to a few nodes. Scale-free networks have different characteristics as compared to random networks. Scale-free networks present better anti-failures robustness characteristics than random network [46]. Random networks leave most of the nodes disconnected if multiple node failures happen in the network. However, a scale-free network is affected less by these random nodes failures because they have a high probability of only attacking the low connectivity nodes. Also, scale-free distributions have been studied in physics, such as analysis of fractals, which refers to objects appearing similar (at least in some statistical sense) at every lengthscale [84]. For more recent reviews of complex networks and scale-free network see [68, 85].

2.3.3 Hierarchical Clustering

Clustering is an approach for assigning a set of objects into subsets (clusters) such that the objects within a cluster are closer to each other as compared to the objects in different clusters [86]. Hierarchical clustering involves recursive clustering using previously assigned clusters [87]. At the highest level of the hierarchy, all objects are within one cluster. At the lowest level, each object is its own cluster and the number of clusters is equal to the number of objects. Between the highest and lowest levels, various intermediate levels of clusters are generated based on the similarity (or closeness) or distance between different objects. Various measures such as Euclidean distance, Manhattan distance, maximum distance, Mahalanobis distance, and cosine similarity are
commonly used. Hierarchical clustering is used in statistical data analysis, pattern recognition, and data mining applications.

A weighted network is a network where there are weights assigned on the link between two nodes. In weighted networks, the weights can be used to represent the similarity or dissimilarity between nodes. The larger the value of a dissimilarity coefficient, the less similar the two objects are. Conversely, the larger the value of a similarity coefficient, the more similar the two objects are.

Recall the approaches of generating network in Section 2.2, “1” denotes that there is a connection between two nodes. These numbers in the elements of the adjacency matrix are the weights of this network. Therefore, for the OSS social networks discussed in this thesis, the weights represent the similarity (closeness) between people and projects. Clusters of people represent participants working closely with each other. In hierarchical clustering process, we use the Euclidean distance to measure the literal distance between two objects when they are viewed as points in the two-dimensional space formed by their attributes [55]. The result of hierarchical clustering is a tree with closely related nodes closer to each other and the dissimilar nodes distant from each other. The relative sizes of clusters and their overlap convey significant information about the network structure.

2.3.4 Network Visualization using Images

For complex networks with a large number of nodes, it is difficult to observe nodes and links of clusters. An alternate visualization approach is based on the image representation of a matrix. The image corresponding to an adjacency matrix provides a convenient visual representation of the network structure. A network structure image is generated by mapping each element of the matrix into a point in the image. Hence, a
n × n matrix maps into an image with a size of n × n pixels. The color of a point in the image corresponds to the values of the elements in the matrix. Above all, it is beneficial to use image method in social network analysis. Firstly, the adjacency matrix, which includes all the network information, is a numerical expression of a network. Excavating messages from the adjacency matrix is an efficient and accurate way of exploring the properties of a network. Moreover, transforming the adjacency matrix by regulating the column and row sequence is the robust characteristic of image method. It is usually combined with the hierarchical clustering method to visualize the clusters within networks. At last, some of the images processing methods provide additional assistance in the network analysis, such as using the Spline interpolation and Bezier curve transformation.
3 ANALYSIS OF DRUPAL NETWORK STRUCTURE

3.1 SNA Metrics Analysis of Social Network (Drupal Community)

Drupal 5 and 6 cores have majority of applications released from Drupal community at present. The main changes between version 5 and 6 are evident in Drupal 6's major usability improvements, security and maintainability advancements, friendlier installer, and expanded development framework. Further, from bug fix to feature request, these issues follow-through on the Drupal project's continued commitment to deliver flexibility and power to themers and developers. However, the first version of Drupal 6 was released in February 2008 [60]. For the analysis purpose, longer history of a social network is better for exploring the evolution and analyzing the network structure. One of the most important reasons is that the population becomes remarkable diverse. The data used in this thesis was collected for Drupal 5.x in August 2009. The data consists of 1907 projects (modules) and 1217 participants who contributed the code during the nine years from the start of the project in the year 2000. During the nine years development of Drupal 5.x, it is believed that Drupal community has a diverse population, which consists of active users, co-developers, core developers and project leader. The boundaries of the population are those imposed or created by the participants themselves. Most commonly, these boundaries are defined by the personal interest, professional skills and the spare time of the participants. Someone who is interested in fashion will probably contribute to some nice theme and skins rather than some customized plug-in. Alternatively, a person who has a better professional skill in programming or more time spent on the module development may develop a more powerful application.
The data used to create the bipartite network consist of people and project from which two networks, discussed in Section 2.2 are generated. The two networks are referred to as people network and project network. People network is a combination of participants themselves and their relations in Drupal 5.x community. Similarly, each node in project network represents one module of Drupal 5.x community. Through the SNA metrics of people network, we can better understand the population diversity and structure, evolution process of the network and some of the Drupal community properties. On the one hand, project network provides alternative demonstration on the properties of Drupal community; on the other hand, it conveys a message about how the Drupal 5.x core is built up by these related projects. The characteristics of the two networks are listed in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>People Network</th>
<th>Project Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>24.626</td>
<td>42.14</td>
</tr>
<tr>
<td>Centralization</td>
<td>1.26%</td>
<td>2.71%</td>
</tr>
<tr>
<td>Average Distance</td>
<td>2.86</td>
<td>2.87</td>
</tr>
<tr>
<td>Average Density</td>
<td>0.0202</td>
<td>0.0221</td>
</tr>
<tr>
<td>Clustering Coefficient</td>
<td>0.74</td>
<td>0.827</td>
</tr>
<tr>
<td>Connectedness</td>
<td>0.4465</td>
<td>0.5454</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.9738</td>
<td>0.9708</td>
</tr>
<tr>
<td>LUB</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

In Section 2.3.1, all of the metrics in Table 2 were defined. It is observed that both the networks are similar in terms of the metrics listed in the table. For networks with over 1000 nodes, the average distance between the nodes of 2.86 and 2.87 are very low. The definition of degree centrality was depicted in Section 2.3.1. The degree centrality of a node represents how much different it is from a star network of the same size. In the equation (7), the degree centrality of a network is largely dependent on the sum of the
difference between the degree of each node and the highest degree in the network. The
degree centrality of both networks in Drupal community is low, 1.26% in people network
and 2.71% in the project network. This indicates that most of the nodes in the network
share similar value of degrees. The variance of the degree in the network is small. The
number of nodes with high degree is in a small proportion. Hence, from the Figure 7, we
can observe that the “star” network structure is not prominent in the Drupal community.
With the low average degrees of 24.62 and 42.14, we can assume that both networks are
in a low-scale unitary connection. The low average density in both networks implies that
different people develop most of the projects, and there are more co-developers than core
developers. The “ideal typical” hierarchy is an out-tree (Figure 6b) as discussed by
Krackhardt. The model of Drupal community will never be a pure out-tree model because
every link is reciprocated. This means reciprocal relation between two entities (nodes) is
equal and this obviates a pure hierarchy. This is why the value of hierarchy is zero in both
networks. In graph theory, one of the characteristics of a tree is that a tree always
connects to a single component. Drupal network is not even a tree structure. In fact, the
majority of the OSS community may not be a connected network and Drupal is one of
them. The connectedness value in Table 2 measures the proportion of nodes that cannot
reach other nodes. From the data, it seems that only half of the nodes are reachable in
both networks. However, with over a million possible links in people network and three
millions in project network, the connectedness, 0.4465 in people network and 0.5454 in
project network, show high extent of connectivity of the two networks. The extremely
high value of efficiency of both networks is incredible. The greater the inefficiency it is,
the bigger the difference the actual number of links and the maximum possible links in a
component has. This is another indication that Drupal network is in low-scale connection. Moreover, we can implicate that both the people and project networks of Drupal community are divided into several components or clusters, which we will introduce into a later section. These characteristics provide basic information about the network structure.

The combination of low average distance and high clustering coefficient denotes that this network is highly connected. In network analysis, this is called “small-world” phenomenon. In 1960s, Stanley Milgram, a social psychologist, conducted an experiment to trace paths through the social network of residents in the nation. In his small world experiment, randomly selected individuals from Nebraska and Kansas were told to send a packet to his acquaintance whose name was on the roster; or sent to one of his acquaintances who may know the person named on the roster. When and if the package eventually reached the contact person in the roster, the researchers could examine the roster to count the number of times it had been forwarded from person to person. All of the people named on the roster in this experiment were in Boston, Massachusetts because
it has a long distance from Nebraska and Kansas, both socially and geographically. As a result, Stanley Milgram found that the average path length was around 5.5 or six. Hence, the researchers concluded that people in the United States are separated by about six people on average, which is called “six degrees of separation”. Small world phenomenon means that any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances [74]. In network science, the average distance between any two nodes in a social network is small if the network presents a small world phenomenon. Based on the characteristics in Table 1, we assume both the people and project networks of Drupal community are “small world” networks. This means the distance between any two participants in the Drupal community is between 5 and 6. Here we speculate the Drupal network is formed by many circles. Assume that participants A, B and C in the people network, there must be paths ab, bc and ac connecting A, B and C according to the small world phenomenon. The average of nodes in paths ab, bc and ac is 5. Therefore, a circle is formed with average 18 nodes. In graph theory, if any two nodes in a network belong to a circle, this network is non-separable, which means that one cannot cut an edge to separate the network into two set of nodes. If some clusters of Drupal network are non-separable, it implies that all of the participants in the people network are working as a large group. They share their knowledge, and interests to develop module towards similar goal. The product they create functions coherently, thereby adding value for the customers. This property may be the one propelling the evolution of Drupal community.
3.2 Degree Distribution of the Networks

In Section 2.3.2, degree distribution and scale-free network properties have been introduced. In the process of analyzing the Drupal community, we want to verify whether the people and project networks of Drupal community are scale-free network or not. A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. The power law distribution highly influences the network topology. For example, it turns out that the major hubs are closely followed by smaller ones. These, in turn, are followed by other nodes with an even smaller degree and so on. Therefore, checking a network’s scale-free properties is another way to analyze its topology. To obtain the degree distribution of both people and project networks of Drupal community, we first use UCI\textit{net} [80] to calculate the degree of each node in the network. Then a program used to count the cardinality of the degree is developed by using Matlab [88]. At last, we need to find out the relationship between the degree cardinality and the amount of different cardinalities of a network.

The degree distributions are plotted in Figure 8 and Figure 9. Figure 8 shows the degree distributions of the nodes in the bipartite network whereas Figure 9 contains the degree distributions of the people and project networks. The points in Figure 8 represent the number of people working on different modules and the number of modules on which different people contribute to. On the other hand, the points in Figure 9 represent the number of projects linked to other projects through common contributors (left) and the number of people linked to other people (right). The X and Y axes in both the figures are the degree cardinality and the number of nodes in different degree cardinality respectively. It is observed that the degree distributions of the networks are linear on a
log-log scale indicating a scale-free topology of all three networks. Such a scale-free topology has been observed in many biological, technical, and social networks. The community of Sourceforge has also been shown to have a similar degree distribution.

![Figure 8 - Degree Distribution of Two Types of Nodes in the Bipartite Network](image)

In a scale free network topology, there is a small set of nodes with a large number of links with other nodes, and a large number of nodes with small number of connections. The nodes with a large number of connections are called the “hubs”. The hubs in people network are the small number of core developers who communicate with and support a large number of other participants. The hubs in the project network are the key projects that provide the core functions of Drupal. These key projects usually are large projects in the OSS community that need a group of people to work on. Also, in order to finish efficiently, this kind of group of people is well organized. It always includes a project leader, a couple of core developers and some co-developers. This kind of well organized group may develop not just one large project but many. Moreover, most of the developers might develop some patches or small projects independently in their spared time. This may be one of the hypotheses to explain the “hub” in the scale-free network. In the bipartite network, the “hubs” can be either core developers or key projects.
A widely accepted model for generating a scale-free network is the preferential attachment model. Preferential attachment is a key concept in the field of scale-free networks. In a wide range of graph models the basic mechanism behind the emerging power law degree distribution is that the new nodes attach to the old ones with probability proportional to their degree. Furthermore, in earlier works the occurrence of preferential attachment was directly demonstrated in several real-world networks with scale-free degree distribution [89]. In the case of degree distributions, it is conjectured that, for a variety of reasons, nodes accumulate new edges in proportion to the number they have already, leading to a multiplicative process which is known to give power-law distributions [90]. According to this model, networks grow through the addition of nodes. New nodes preferentially attach to other nodes with high degree. Hence, the probability of attachment of a new node to existing nodes is proportional to the degrees of the existing nodes. We believe that the model explains the emergence of scale-free networks in the open-source domain because the modules that have higher number of participants develop faster, thereby increasing the modules’ utility, and hence attract even more participants.

![Figure 9 - Degree Distribution of Projects (left) and People (right) Networks](image-url)
3.3 *Network Structure Analysis through Clustering*

In this Section, we analyze the structure of the network by combination the methods of clustering techniques and network visualization, which are introduced in Chapter 2.3. The original and clustered networks are visualized using images corresponding to the adjacency matrices of the networks. Figure 10 is a black and white image corresponding to the binary bipartite network with people and project nodes. The horizontal and vertical axes correspond to people and projects respectively. Each pixel in the image represents a pair of nodes. The weight of a link between the pair of nodes corresponds to value in the 3D plot image. For example, in Figure 10, black color represents no link between the nodes and white color represents a link with weight = 1, indicating a link between a person and a project. The people and projects are arranged in the increasing order of their IDs.

![Figure 10 - Adjacency Matrix Image of the Bipartite Network](image)

One might be curious about the shape of a “convex curve” in Figure 10 which divides the image into two areas. The top right corner area is pure black, while the other area has some bright pixels on the black background. Why the shape of this curve is convex but
not concave? Why the convex curve seems continuous? And why this curve can separate the image into two areas? Among these questions, some of the answers are valuable for the properties exploration of Drupal community. For the last question, there are two different areas in the image separated by the convex curve which is caused by the way where we generate the ID of people and project right after the process of data collection. After collecting all of the committer tables from Drupal, we assembled all the data into an overall information table and sorted the project’s name by the alphabet order. As shown in Figure 4, we gave an ID to each unique name in the project network as well as the people network from the beginning of the overall information table. The people developed project “1” is given a tag ID earlier than the person of project “2”. We replaced the project and people’s name with ID under this rule for the overall information table. The people and project sequence in Figure 10 are arranged in the increasing order of their IDs (1, 2, 3, 4, 5…). That is the reason why we can find any bright pixel on the right of the first line in Figure 10. The people corresponding to project “1” will not be given a large number IDs.

Another question we are going to answer is why the curve has a convex shape but not concave. Figure 11b is zoomed in from part of Figure 10. We can observe that the convex shape is an aggregation of many “horizontal ladder shapes” in a microcosmic view. The “horizontal ladder shapes” denotes that there are groups of people working in the same project. The people in the same group may work independently for their own part and assemble them together at the end; or they may cooperate with each other under the lead of a project leader. Furthermore, we can think of these “horizontal ladder shapes” in another view. Assume that one participant started a project at a time and this is a middle
size project which needs time to finish. The world is developing and so is the Drupal community. Before participant A finishes his project, there may be some new participant B or C coming to join the project which was started by A. This indicates that the Drupal community is growing fast in population of developers, or at least it is growing fast during the “horizontal ladder shape” period. On the contrary, we can assume that the Drupal community is not growing fast enough. From the time participant A started his project to the time he finished it, no new participant joins to this project. After finishing one, participant A started another project. If this happened again and again, one can imagine that the shape of the curve will not be ever “horizontal”, but “vertical” instead. In Figure 11a, we can observe that there are several “vertical ladder shapes”. This is another hypothesis of the development of Drupal community. Instead of the blasting of population, the “vertical ladder shape” phenomenon can be called the blasting of intelligence. Assume that the existing developers of Drupal community, which consist of co-developers, core developers and the project leaders, are all talented in developing projects. They usually develop a project independently. The Drupal community is supported by a small but intelligent group of people. Therefore, Figure 10 gives us some information about that how the Drupal community is organized.

Figure 11 – Two Hypotheses of the Curve in Figure 10

a) Zoom in of the concave curve

b) Zoom in of the convex curve
Moreover, we analyze the structures of networks using clustering techniques. 3D plots are used to visualize the adjacency matrices corresponding to the weighted people and project networks. The plots are shown in Figure 12 and Figure 13 respectively. The z-axis corresponds to the weights of links connecting nodes on x and y axes. The weights on the links are used as the similarity measure for clustering purposes. This is because the larger weights on people networks indicate that people are working together on greater number of projects. Similarly, larger weights on the project networks indicate that the projects share greater number of participants. Hence, the larger the weights, closer the nodes are.

The adjacency matrices are clustered and the corresponding plots are shown in Figure 12. From these plots of the clustered networks, it is clear that there are few participants near the top-right corner who have connections with a lot of other people. These participants are the core developers who contribute to the entire project and oversee the work of other participants. These participants are also highly connected with each other. The other participants are weakly connected and have a few links with each other. As we discussed in the previous Section, this kind of distribution is due to the scale free nature of the network. Hence, the plots provide an indication of the different roles of the individuals. Similar characteristic is also observed in the project network.

In addition to the highly skewed distribution of links between nodes, we also observe clusters of nodes appearing as blocks in the diagonal. The blocks indicate sets of participants or projects that are highly connected. The highly connected participants are similar to teams of individuals working together in traditional product realization. The
difference, however, is that these clusters (teams) in open-source are self-organized as opposed to being pre-defined as in traditional product realization.

3.4 Hierarchy Organization Analysis

In Chapter 3.1, we showed most of the metrics which measure the network properties for both people and project network. The two networks share zero value in the metrics “hierarchy” and high clustering coefficient. Also, we stated that the people and project network of Drupal community represents scale-free property in Chapter 3.2. To analyze the hierarchical structure of the network, we have discussed how to measure the hierarchy value by an extended Krackhardt’s “out-tree” model before. However, some reviews provide a different view in analyzing the hierarchical structure of a network [90]. In order
to bring the high degree of clustering and the scale-free topology under a single roof, we have to make a reasonable assumption that a network combines the two properties in a hierarchical way. The network generated under this hierarchical manner is called hierarchical network. The presence of a hierarchy and the scale-free property impose strict restrictions on the number and the degree of cohesiveness of the different groups present in a network, which can be captured in a quantitative manner using a scaling law, describing the dependence of the clustering coefficient on the node degree [91]. A quantitative manner using the recent finding of Dorogovtsev, Goltsev, and Mendes [68] that the clustering coefficient $C(k)$ of a node with $k$ links follows the scaling law

$$C(k) \sim k^{-1}$$

(10)

The following systems are investigated to validate that hierarchical organization is present in real networks [90].

![Figure 14 – The Scaling of $C(k)$ with $k$ in Different Real Networks [91]](image)
Figure 14 (a) is an actor network of 392340 nodes and 15345957 links. Any two actors in Hollywood if they acted in the same movie are connected. As Figure 14 (a) indicates, $C(k)$ scales as $k^{-1}$, indicating that the network has a hierarchical topology. At high $k$, the $C(k)$ curve splits into two branches, one of which continues to follow Equation (10), while the other saturates. Figure 14 (b) is the semantic web, which connects two English words if they are listed as synonyms in the Merriam Webster dictionary. The obtained semantic web has 182,853 nodes and 317,658 links, and it is scale-free with degree exponent $\gamma = 3.25$. Figure 14 (c) is the scaling of $C(k)$ of the World Wide Web based on the data collected in [92]. While the obtained $C(k)$, shown in Figure 14 (c), does not follow as closely the scaling law as observed in the previous two examples, there is a clear evidence that $C(k)$ decreases rapidly with $k$, supporting the coexistence of many highly interconnected small nodes with a few larger nodes, which have a much lower clustering coefficient. Figure 14 (d) depicts that Internet at the autonomous system level, each node representing a domain, is connected if there is a communication link between them.

From the above examples, we also want to use this scaling law to identify the presence of a hierarchical architecture in Drupal community. The people network has 1217 nodes and 19026 links and the project network has 1907 nodes and 60966 links. By inputting the network data into Ucinet 6, we obtained the degree and clustering coefficient of all nodes in both the networks. Figure 15 is the scaling of $C(k)$ with $k$ links in people and project networks of Drupal community. Each point in the figure indicates a node in the network (square ■ in people network and circle ● in project network). As Figure 15 (a) indicates, we find that $C(k)$ scales as $k^{-1}$, indicating that the network has a
hierarchical topology. Indeed, the majority of developers with a few links (small k) participate only in one project. Such developers have a clustering coefficient equal to one. These developers link to the same project, and therefore connected to each other. The nodes with high k value include many developers who joined in several project, and thus, their neighbors are not necessarily linked to each other, resulting in a smaller

C(k). In Figure 15 (a), the nodes whose degree is higher than 100 is following the scaling law more strictly than the low degree nodes. For the part which followed the scaling law, the nodes with higher degree result in higher clustering coefficients. From the equation (1) of clustering coefficient, clustering coefficient of a node will reduce largely even if only 1 link increasing on the degree of a node. One hypothesis for this scaling law phenomenon in the people network is that the people with high k value are clustered in huge projects, which usually include a large group of developers. Once this kind of people joined a new huge project group, their neighbor will enormously increase so that the clustering coefficient is also increasing. These people are usually considered to be the central in a hub in the people network. One can observe that the number of nodes
following the scaling law (the dash line) is a small proportion to the whole population of people network. And as introduced in the Chapter 1.5, these people may be in the role of core developers. Therefore, despite no hierarchical architecture throughout the overall people network, the high level of the population such as core developers and project leaders will display a hierarchical architecture. In Figure 15 (b), the project network does not follow as closely the scaling law as observed in the people network.
4 ANALYSIS OF NETWORK EVOLUTION

In Chapter 3, we discussed network structure of Drupal community using data up to August 2009. Drupal was founded in 2000, rapidly growing from 12 developers and 23 projects to 1217 developers and 1907 projects. The network evolution analysis is aimed at understanding how the bipartite, people, and project networks grow over time. Through the analysis of network evolution, our objective is to identify characteristics of the network that change over time and the characteristics that are invariant with time. The knowledge about the evolution of these networks can be used to better predict the later development of Drupal community.

4.1 Generation of Snapshots of Networks at Different Intervals

During nine years of development, Drupal community has grown by a factor of about 100, both in terms of the number of participants and modules. In order to analyze the evolution of the community, six snapshots are generated based on the time when people joined the community and time when projects were created. The snapshots are generated at intervals of one year, starting with Year 4, which means the fourth year after Drupal launched. Different snapshots are not generated for the first three years because most of the evolution in the network took place between years 4 and 9 of the project. Therefore, we set our initial analysis point at year 2003, three years after Drupal community was formed. The number of participants (people) and modules (projects) at different snapshots are shown in Figure 16. It is observed that the number of participants and modules has grown exponentially. The exponential curve fit for both is shown in the figure, which provides an important indicator of evolution of the Drupal community.
We found that there are exponential relations between the population and years of the people and project networks of Drupal community:

\[ y_1 = 19.116e^{0.7618x} \]  \hspace{1cm} (11)

\[ y_2 = 8.5157e^{0.8416x} \]  \hspace{1cm} (12)

where \( y_1 \) and \( y_2 \) denote the population of projects and people respectively, \( x \) denotes the number of years after the initial analysis point of studying Drupal case as stated above, which \( x \) is equal to 1 at years 4 in Figure 16.

One can observe that the populations over six years from Year 4 are closely following the exponential relationship above. Theoretical value can be calculated by using the two equations (11) (12). The errors between theoretical value and actual population of the two networks are acceptable. They are 9.5% in people network and 8.5% in project network. The error in both networks is under 10% through an over 6 year’s statistical study. During these six years, Drupal community was derived from a small group of OSS programmers, to a community with hundreds of developer, and finally became a famous free and open source content management system with thousands of users. The discovery of exponential relationship serves to be a way to predict the blast of population in Drupal.
community. This prediction is one of the most valuable results in this research, especially for the simulation of the evolution process of Drupal community.

Figure 17 and Figure 18 are two digraphs shown the structure evolution of the people and project networks. In each figure, there are six sub-graphs which are snapshots of people or project network structure. Nodes in the graphs denote one participant in people network or one module in project network. The link in the graphs means the cooperation between two developers or two projects sharing a same developer. If two developers A and B of Drupal community take part in a same project, two arrows are pointed to A and B respectively because they are reciprocal from each other. It is the same convention in the project network. If two projects C and D share a common developer, two arrows are pointed to C and D respectively because they may have a same programming structure or similar function for users. In fact, all the links in both people and project networks are reciprocal links, which means that the nodes at the end of the link are always pointed to
each other. As discuss in Section 2.3.1, these reciprocal links and the direction they pointed to affect the format of the out-tree model, which will finally influence the Krackhardt’s hierarchy of a network.

In the previous section, we have many hypotheses on the structure of both people and project networks, some of which can be proven by observing the graph of the two networks from Figure 17 and 18. First, we speculate both the networks do not have a prominent “star” network model. This can be told apart from the network structure of early stage, such as in the Year 4 structure in both networks. Second, the scale-free property of both the networks indicates that there are some “hubs” in the networks. From Figure 17 and 18, we can observe that some of the nodes at the center of the graph have more connections to others than the peripheral nodes in the graph. We can consider this kind of nodes as the “hubs” of the networks. Third, in Chapter 3.1, we showed that both networks have zero value in “Hierarchy” item in Krackhardt’s hierarchy analysis. This means the internal networks are not distributing a “tree” structure and the networks are not hierarchical organized. However, it is not precise to judge a network is whether hierarchical or not only by one measurement. We present our hierarchical organization analysis in Chapter 3.4, whose result is referable for considering hierarchical structure of a network. Our hypothesis is that both the networks are presenting a partial hierarchical architecture. We can observe from Figure 17 and 18 that high degree nodes are those in black background, which is formed because of the overlap of thousands of links. These high degree nodes connect to each other and to some of the low degree nodes at the same time. At last, we find an interesting phenomenon in both figures. A cluster is a set of one or more nodes that are linking to each other. We can observe that a “big cluster” in the
middle of Figure 17 and 18. In fact, these are several overlapping clusters. We made a deep exploration and found out that these are small clusters which contained only two or a few nodes linking to each others. These two “big cluster” are where small projects (modules) or co-developers gather around. For example, developer ID 442, who independently finished project (module) ID 360 and 1446, was cooperated with developer ID 453 and 454 on project ID 374. The three projects were completed in less than one year. The function of project ID 360 is allowing comment follow up e-mail notification for anonymous as well as registered users. If you write a comment to some post, the follow up comments for your comment will be informed to you through your email provided in the comment. The function of project ID 374 is that it implements mechanisms for posting contributions (papers) and managing the reviewing process from assigning the reviewing task up to deciding about acceptation. Once the conference administrator decides what content type corresponds to contributions and reviews, the conference module completely controls access to these content types. One of the similar characteristics between the functions of these two modules is that they provide a customized, convenient and instant way for users to manage and distribute their information. Moreover, the program structure, the interface and port are parallel. These are the common things which actors in the same cluster will probably share.
4.2 Evolution of the Network Characteristics

In Chapter 3.1, we have discussed the analysis of network structure by using the social network metrics. The social network metrics shown before depict the structure of Drupal community up to Aug 2009. However, as we stated before, Drupal have been launched since 2000 and we have setup six different points to analyze the evolution of the community efficiently and typically. The following social network metrics are used to analyze the characteristics of the six snapshots of the people and project networks. The data collecting process has been introduced in Chapter 2.1. With the overall information sheet of the whole Drupal community, we spilt it up to six copies by the different time intervals discussed above. The results are shown in Table 3. The trends in the evolution of the networks are analyzed using average degree, degree centrality, average density, clustering coefficient and connectedness.
### Table 3 - Highlights of Drupal Network Evolution

<table>
<thead>
<tr>
<th></th>
<th>People Network</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Years</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Degree</td>
<td></td>
<td>24</td>
<td>32.734</td>
<td>32.8</td>
<td>30.392</td>
<td>24.853</td>
<td>24.636</td>
</tr>
<tr>
<td>Centralization</td>
<td></td>
<td>14.05%</td>
<td>5.61%</td>
<td>3.41%</td>
<td>2.27%</td>
<td>1.57%</td>
<td>0.0126%</td>
</tr>
<tr>
<td>Average Density</td>
<td></td>
<td>0.48</td>
<td>0.3031</td>
<td>0.1155</td>
<td>0.0553</td>
<td>0.0277</td>
<td>0.0202</td>
</tr>
<tr>
<td>Clustering Coef.</td>
<td></td>
<td>0.842</td>
<td>0.855</td>
<td>0.816</td>
<td>0.787</td>
<td>0.754</td>
<td>0.74</td>
</tr>
<tr>
<td>Connectedness</td>
<td></td>
<td>0.742</td>
<td>0.6959</td>
<td>0.5685</td>
<td>0.5084</td>
<td>0.3943</td>
<td>0.4465</td>
</tr>
<tr>
<td>Hierarchy</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td>0.753</td>
<td>0.827</td>
<td>0.8944</td>
<td>0.9383</td>
<td>0.9591</td>
<td>0.9738</td>
</tr>
<tr>
<td>LUB</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Project Network</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Years</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Degree</td>
<td></td>
<td>54.519</td>
<td>91.723</td>
<td>57.904</td>
<td>45.82</td>
<td>39.207</td>
<td>42.136</td>
</tr>
<tr>
<td>Centralization</td>
<td></td>
<td>20.34%</td>
<td>15.13%</td>
<td>8.1%</td>
<td>4.29%</td>
<td>3.07%</td>
<td>2.71%</td>
</tr>
<tr>
<td>Average Density</td>
<td></td>
<td>0.7174</td>
<td>0.5559</td>
<td>0.1392</td>
<td>0.0517</td>
<td>0.0262</td>
<td>0.0221</td>
</tr>
<tr>
<td>Clustering Coef.</td>
<td></td>
<td>0.868</td>
<td>0.912</td>
<td>0.853</td>
<td>0.838</td>
<td>0.841</td>
<td>0.827</td>
</tr>
<tr>
<td>Connectedness</td>
<td></td>
<td>0.8254</td>
<td>0.7116</td>
<td>0.6148</td>
<td>0.5981</td>
<td>0.4945</td>
<td>0.5454</td>
</tr>
<tr>
<td>Hierarchy</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
<td>0.4838</td>
<td>0.6027</td>
<td>0.8707</td>
<td>0.9429</td>
<td>0.9627</td>
<td>0.9708</td>
</tr>
<tr>
<td>LUB</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

In Figure 19, the mean degree of the nodes in the networks is plotted at different time steps. The mean degree of project network is generally two times higher than that of people network. However, we can observe that the typical trend of these two curves displayed in similar. Figure 19 shows three stages of the networks’ evolution - the first stage where the mean degree increases, the second stage during which the average degree reduces, and finally the third stage where the mean degree remains relatively constant. The three stages are observed in both people network and the project network. We believe that this trend can be explained as follows. During the first phase, which is also the start up stage of Drupal community, most of the contributions to the projects were from the core developers and founders, who are going to become the main project leader.
in the community later in most cases. All the developers closely collaborate with each other. Any additional number of core developers results in the increase in the degree of all the nodes in the network. Hence, the average degree increases during this first phase. This phenomenon has also been discussed as an initial euphoria among the co-developers and core-developers [62]. Especially, we can observe that the mean degree of project network in Years 5 is prominent high. To some extent, this measurement can demonstrate that a group of developers were ambitious and enthusiastic to their work and contributed a lot to the community at that time. Both the two networks reach the peak of mean degree 5 years after the community launched. During the second phase, one of the reasons that the mean degree statistical reduces is that as more developers join the network, the number of participants and modules (i.e., nodes in the network) increase at a rate faster than the number of links. We can check from Figure 16 about the apparently increasing rate of the community population began from Year 5. The other reason may be concluded that the fade of enthusiasm of the initial members in the community. Most of the OSS as well as online platform will face this kind of dilemma. During the third phase, the existing participants explore other modules and initiate collaborations with other participants. This stage is usually seen as the true organic properties of the networks.

![Figure 19 - Mean Degree of Project and People Networks with Respect to Time](image)
Figure 20 shows the degree centrality of the network as a function of time. The degree centrality of both the networks shows a steep decline during the initial phase, indicating the significant difference between the nodes with highest degree and the other nodes. During the initial years of an open-source project, core-developers play an important role in the development of the project. At that time, some project leaders start lots of projects while others start only a few. This causes the significant difference seen in the figure. During the later phases, there are more choices of projects for people to contribute to, which results in a gradually decreasing curve. Degree centrality also provides a comparison between the network structure and the “star network” (Figure 6c). We can recall Figure 17 and 18, the structure of people and project network. The bigger Drupal community grows, the more “texture” (Figure 6a) structure it presents. This is how the network goes the opposite way of “star” network structure.

![Figure 20 - Degree Centrality of Project and People Networks with Respect to Time](image)

The average density of the networks is plotted in Figure 21. The trend of the average density is similar to the degree centrality. It reduces steeply after the initial phase and then reduces gradually during the later phases. During the initial phase, there is a closely connected group of core-developers and project leaders, which means that the nodes are highly connected in both project and people networks. In order to maintain the same
density of a network, the number of links must increase at a rate equal to the square of the number of nodes. However, we observe that rate of increase of the number of links is almost equal to the rate of increase in nodes. In Figure 16, we have shown that the population of project and people network is growing closely to the exponential equation. And we recalled the definition of average density in Section 2.3.1. Form the equation (3), if the average density of Drupal community was standing at still level, no mention the increasing level, the link between nodes should at least increase in square exponential relation. For example, if we want to keep the average density stay at 0.7, the people network needs more than 14000 new links when only 100 new nodes added in the network. Hence, the density reduces as shown in Figure 21 from the Year 4 to the Year 6. However, in the stage from Year 7 to Year 9 in both networks, the level of average density stands still. This implicates that the link is growing rapidly despite a sharp increasing population.

The clustering coefficients of the people and project networks are plotted in Figure 22. After reaching their peaks in a short time, both the curves undergo a steady decline. We believe that the reason for the downtrend is the exponential increase in participants in the Drupal network. Like the demonstration of average density, we need to begin with the
definition of clustering coefficient in Section 2.3.1. To keep clustering coefficient in the same level of one node, if a new neighbor is adding to one node, all its n original neighbor need to create \( \frac{n+1}{n-1} \) times of new links in the neighborhood. For these \( \frac{n+1}{n-1} \) times of new links, it possibly happened in a professional group of neighborhood because this kind of professional group is well organized and target aimed. The robust developing ability is driving the creation of new links in this group. However, abundant of professional groups is not a common case existing in the social network. Therefore, it is reasonable to believe that the clustering coefficient decreasing because of the increasing population. Further, the core modules of Drupal were developed during the first phase, resulting in the high clustering of the people network. The closely related core modules result in higher initial clustering coefficients. Despite its decline, the clustering coefficients in both networks are still high compared to random networks. Both the networks keep a high clustering coefficient high than 0.72 over the statistical 6 years. The low average node distance in both networks combined with the high clustering coefficient is a characteristic of small-world networks.

![Figure 22 – Clustering Coefficient of Project and People Networks with Respect to Time](image-url)
4.3 Evolution of Scale-free Property

In Chapter 3.2, we discuss the degree distribution of the two networks. In this section, we explore how the degree distributions of the Drupal network change over time. The degree distributions of the people network and project network for six snapshots of data are shown in Figure 23.

If the degree distribution is following the power law, one can said that the network has scale-free properties. We have discussed the scale-free network in Section 2.3.2 and 3.2. From Figure 23, we can observe that the degree distribution of the bipartite network of Drupal community is closely following the power law over all the six snapshot data. This implies that the Drupal community maintains scale-free properties, started from the date it launched. Scale-free networks have robustness characteristics because of they formed “hubs”. In the “hubs”, a large number of nodes with only a few connections connected to a very small number of highly connected nodes. So the random node failures have high probability in disconnecting one of the low connectivity nodes. We have discussed the “hubs” of Drupal community in Section 4.1 by observing the structure of people and project networks. It is easy to observe the “hubs” in a network with less than 100 or 200 nodes. However, it is hard to notice the number of “hubs” only by observing in a network with more than 1000 nodes. One of our future objectives from analyzing the evolution of scale-free properties of Drupal community is to find out some alternative ways to deciding approximate level of numbers of “hubs”.
It is observed that during each data snapshot, the form of the degree distribution remains the same – power law. The only difference is in terms of the parameters of the power law. The coefficient increases while the exponent decreases with time as shown in the figure. Faloutsos [93] has shown that the frequency, $f$, of an outdegree, $d$, is proportional to the outdegree to the power of a constant $k$. From Figure 23, we have obtained the relationship from our data, the frequency and degree:

$$f = \beta \times d^\alpha$$

where $d$ is the degree and $f$ is the frequency of a degree, $\alpha$ and $\beta$ are constants.

In Figure 23, it is observed that there is an approximate exponential relationship between $\alpha$ and $\beta$ in bipartite network. This helps in eliminating one constant $\alpha$ or $\beta$ in the equation of frequency and degree. Hence, the evolution of the Drupal network far from arbitrary and follows some common trends observed in complex networks [93].
Currently, we have obtained the equation of frequency and degree with only one constant by elimination either $\alpha$ or $\beta$. In future work, we will determine whether there is a relationship between these constants and other parameters of the network, such as the number of nodes or links of the network.

4.4 Visualization of the Network Structure Evolution

In this section, we construct an adjacency matrix for the snapshots of network at different times. In both people and project networks the row and column sequences, which stand for participants and projects are ordered for the past six years. Figure 24 shows how the Adjacency matrix is reordered by sequencing the node ID at different times. The sign “Years 4” means the nodes in this square joined Drupal 4 years after it launched and the same convention for the other “Years 5, 6, 7, 8, 9”. This method is oriented and objective. In the micro-behavior level, we can easily observe how one node is related to others in different time periods. In the people network, we can find out how the old developers cooperated with the other new comers only by checking the first few rows in the “Year 4” intervals. Moreover, this method is effective for observing the changes in overall network structure. In the general observing process, this adjacency
matrix is hierarchical with time snapshots. With the image visualization introduced in Section 2.3.4, this adjacency matrix ordered by time snapshots indicates the evolution of Drupal community.
Figure 25 – Adjacency Matrix Image of Evolution of People Network

(b) People Network (4 years ago)

( d) People Network (2 years ago)

(g) People Network (5 years ago)

(e) People Network (3 years ago)

Figure 25 – Adjacency Matrix Image of Evolution of People Network
Figure 25 shows the adjacency matrices of the people network at different times in an image form. The participant IDs are organized in an increasing order based on the time when they joined in Drupal. For example, the developer who joined Drupal two years ago has an ID greater than the participant who joined four years ago. Similarly, the projects’ IDs are also arranged in increasing order. The color bar indicates the values in the adjacency matrix. A point in carmine means the value of this position in the adjacency matrix is higher than other points in other colors. For example, in Figure 25 (a), a point in carmine indicates two participants collaborating on 6 projects, which is also the highest among those who joined people network in the 5 years ago. In fact, as the population of a network growing, different color in the color bar is implicating different value levels between nodes. In Figure 25 (d), we have a more than 500 people network. If a point is in the color of red or orange, the value in the same position of the adjacency matrix should higher than the one in green or blue color.

Figure 25 (a) describes the people network which is formed 4 years after Drupal community launched. There are 17 people in this network at that time. We observe that almost every participant has strong collaborations with others. In Figure 25 (b), more participants are added. The strengths of collaboration between newly added participants are weaker and the strengths of collaboration between existing participants become stronger, as indicated by the change in the color of pixels corresponding to existing participants from blue to sky blue, yellow to red, red to carmine. This means the initial participants of Drupal community make big contributions to the community by developing new modules cooperating with other newcomers. As we stated before, this group of people might play a very important role in the evolution of Drupal community.
This phenomenon is one of the proofs to our hypotheses that most of initial participants are the founders or became project leaders at the end. Another interesting phenomenon is that there is a colored “cross” in Figure 26 (c). Also, we can check out that in Figure 26 (d), this colored “cross” extended to ID 200. These are small group of participants (ID55-57) who collaborate with almost everyone in the network, which signifies that these are participants who joined later but became project leader for the whole community. Another hypothesis for this colored “cross” is that at the beginning of Drupal development, new modules are those most needed by users which were used to carry out the basic function of this content management system. This small group of participants (ID55-57) might dedicate to these kinds of basic modules and cooperated with others in a certain time period.

Figure 26 shows the adjacency matrices of project network at different times. A same trend happens in project network similar to that in the people network. The color of first sets of projects changed from blue to sky blue, yellow to red, red to carmine. This is because more and more developers revised these projects. We have stated that the early modules of Drupal network shared similar basic functions in the system. And this may again became one of the reasons that colors of these projects were changing time by time. The developers who revised on these basic projects are oriented. As we stated about the preference of developers before, these participants might be the person who are interested in strengthen the power of the system other than making it more customized and convenient, or a nicer theme and skin. Similar to people network, there is a big “uncolored cross” (ID 17-19) in Figure 26 bottom right. Also, except for the big “uncolored cross”, there are a lot of small “uncolored cross”. These “uncolored cross”
displays in blue color while most of the adjacency points are in red or orange. One of the assumptions is that these uncolored projects are not a similar type to the others for they might not share same developers.
Figure 26 – Adjacency Matrix Image of Evolution of Project Network
5 ANALYSIS OF EFFORT NETWORK --- COMMITS

In the previous network analysis of Drupal community, we focus on the cooperation between developers in the people network and relationships between modules in the project network. Once two participants join in a same project, we set the value “1” in the adjacency matrix of people network to represent the fact that they cooperated with each other one time, and similarly for the project network. This network analysis method is widely used in OSS community, such as in the SourceForge community, Yahoo and Flickr community and paper-author network. However, if the value set in the adjacency matrix of a bipartite network is related to how much effort a participant contributed to a module, the social network analysis would be more convincible and convictive.

Drupal is a web-based content management system written in PHP that uses a relational database for storage. PHP, which stands for Hypertext Processor, is a high-level language designed for developing web applications. PHP offers a flexibility that supports both procedural and object-oriented approaches to software development. The Drupal core is meticulously written in procedural-style PHP. In such an open source environment, Drupal was designed to allow new features and custom behavior to be added to extend Drupal's core capabilities. As a result, the Drupal development community uses the Concurrent Versions System (CVS) to manage all the revisions of every file that makes up the Drupal system. Typically, CVS uses client-server architecture: a server stores the current version(s) of a project and its history, and clients connect to the server in order to "check out" a complete copy of the project, work on this copy and then later "check in" their changes. With the record of CVS, we obtain all the data about revisions of developers in the community. We can observe from Figure 27, the
CVS of Drupal records every daily change in the community. It provides the information of user’s name of the reviser, what module the reviser run into and what did he/she modified, and the time and date of the revision. Remind that the committer’s table in Figure 4, it simplified the CVS message in Figure 28 but specified to certain modules. The “Commits” item in Figure 4 is telling how many times a developer of Drupal community accessed to the code of the module and made revision on it.

**CVS messages**

CVS (Concurrent Versions System) is a code management system used by developers to collaborate and track modifications of code.

April 21, 2010

- **Commit #357716** by dww at 19:47
  Drupal.org Testing: /profiles/drupalorg_testing/drupalorg_testing.cvs_deploy.make 1.1
  
  
- **Commit #357714** by jamestelliott at 10:47
  Rotating Banner: /modules/rotating_banner/rotating_banner.css 1.25
  
  Created a Fallback for when JS is disabled.

- **Commit #357712** by psynaptic at 10:46
  Clean: /themes/clean/template.php 1.1.2.7.12.17.2.35 © DRUPAL-6--2
  
  Tidied up clean_menu_local_tasks.

- **Commit #357710** by kienn at 19:45
  Extra meta tags modules: /modules/metatags_extra/metatags_extra_operations
  /directory NONE

  Directory /cvs/drupal-contrib/contributions/modules/metatags_extra 
  /metatags_extra_operations added to the repository.

---

**Figure 27 – CVS Message from Drupal Community Recording Revisions on Every Module [60]**

In this Section, we aim at creating an effort network which can represent developer’s different efforts to their modules. In other words, we want to find out what is the difference among the contribution between two developers who joined in the same project. For the effort mentioning in this section, we make a hypothesis that the higher commits of one participant, the more effort he/she contributed to the network. As we stated above, the commit is the number of times one accessed to the code. Our assumption is following the convention of our human behavior: If someone is focusing on something more times, he/she might be interested in it; and if someone is interested in something, the possibility that the efforts they made on it might higher than the one who
is not interested in. Based on this assumption, our goal is to generate the distribution of commits to explore characteristics of the effort network.

At the beginning of the analysis, we have created an information sheet by gathering data about developers, modules, the data and time that each module stated and finished, and commits that each developer spent on their modules as shown in Figure 4. In Section 2.2.2, we generate the regular adjacency matrix of people network by counting how many times two developers joined in a same project before. Different from that method, we use another method which is more appropriate for generating adjacency matrix of effort network. At first, we create the bipartite network. Instead of setting 1 in the Matrix M in Figure 5, we put in the commits corresponding to the developer and modules. Second, when generating the adjacency matrix of people network, each value of elements in the adjacency matrix is the dot product of the two vectors, which stands for two different developers in the network. The value in each vector is the commits corresponding to different modules and the length of the vector is the number of modules in the network. For example, if Matrix M in Figure 5 is the adjacency matrix of bipartite commits network, the value of adjacency matrix of people commits network is:

\[ P(a, d) = [1,0,0,0] \cdot [1,1,1,1] = 1 \]  

where a and d is the ID of two different developers. P is the value of adjacency matrix in people effort network.
Figure 28 is the commits distribution of people and project network. The commits network distribution can be separated into two parts, the commits > 10 and commits < 10. It follows the power law relationship when the commits >10. However, the initial section whose commits < 10 do not obey the power law. We can observe that there are less than 10 points in the initial section both the people and project commits networks. The problem is that the network is short of number of people or projects in the low-commit section. We can notice that the number of people who have 1 commit is highest in the commits distribution of people network. But apparently, the value is not high enough to become the “commits scale-free network”. The people in the low commits section are co-developers in the Drupal community, who modify the code occasionally. However, by investigating the evolution of number of co-developers of Drupal community, we found out that this group of people is extending rapidly. Other analysis will be catching up on this issue in the future research on Drupal community.
6 SIMULATION OF DRUPAL NETWORK EVOLUTION

In previous sections, the case studies reveal the properties of network structure and evolution of Drupal network. The important properties include scale-free characteristics, the clusters, and the evolutionary features of average degree, degree centrality, average density, clustering coefficient, connectedness and degree distributions. However, we still don’t know the mechanism inside the network evolution which produces the results shown in the case study. In this section, two computational models based on the preferential attachment rule are built. The target of the models is to simulate the evolution of Drupal network and reveal the mechanism which produces results displayed in Chapter 4. The main idea of the models is to set up micro behaviors of individuals in the network including: a) how does individual join? b) how does individual create new projects? c) which projects does individual works on? Based on the given micro behaviors of individuals in the network, the macro structures of the entire network and macro behaviors of the network evolution can be displayed by executing the models over time. By executing the models, it can be known whether different micro behaviors of the individuals result in different macro network structures and macro behavior of network evolution. Hence, the mechanism inside the network evolution can be concluded from the micro behaviors of individuals. In Chapter 6.1 the details of two models will be introduced. In Chapter 6.2 the results of the macro structures of the network will be displayed by executing two models and the discussion will be made to analyze the results from the models.

Our hypothesis is that preferential attachment mechanism underlies the growth of the Drupal network also. In order to validate the hypothesis and to predict the underlying
dynamics of the network growth, we present a computational model whose input is the local behavior of the nodes (people) and the output is the structure of the networks at various time-steps. The local behaviors in the network include activities such as new individuals joining the network, creation of new projects, and participation of existing individuals on existing projects. Existing literature on preferential attachment is focused on one mode networks. In this thesis, we extend the model to a bipartite network. Based on specified local behaviors of individuals in the network, the global structures of the entire network and process of network growth can be modeled. Using the model, the effect of different local behaviors on the global network structures can be studied.

6.1 Details of the Models

The model is based on the following assumptions: i) only one new participant joins the network in each time-step, ii) the probability that an existing participant create a new project is based on a constant project/people ratio, iii) the number of links added between existing participants and existing people is constant in each time-step. The inputs of the model are the numbers of people (M), projects (N), and the links between people and projects (L). The model is initialized by creating a small set of people and projects and linking them as a random initial network. The initial network is grown through the gradual addition of nodes (people and projects) and links (the participation of individuals on projects) in discrete time-steps. During each time-step, the network grows in three ways:

a) Participation of new individuals: One new individual is added to the model during each time-step. The new individual either initiates a new project with probability
pₐ, or contributes to an existing project with probability (1-Pₐ). If the new participant creates a new project, a link between the new participant and the new project are created. Otherwise, the new participant preferentially links to an existing project based on their degree (i.e., the number of people working on the project). We utilize the model proposed by Dorogovtsev and co-authors [94] where the probability of linking to a project is equal to the degree plus a constant c₀, normalized over all the projects. Hence, the probability of a new individual i linking to a project j (with degree kⱼ) is:

\[ P_j = \frac{(k_j + c_0)}{\sum_{r=1}^{n_t} (k_r + c_0)} \]  

where \( n_t \) is the number of projects at time-step t.

b) Creation of new projects by existing participants: The probability that a new project is created by existing participants \( p_b \) is determined by the ratio of total number of people and projects (M/N). This is a key parameter used to ensure that the targeted number of people and projects are achieved at the end of the process. It is assumed that the choice of the individual who creates a new project is again based on preferential attachment. Hence, the number of projects created by an individual is proportional to the number of projects that he/she is currently working on (i.e., the individual’s degree) plus a constant c₁. The probability that a new project j is created by an individual i (with degree kᵢ) is thus:

\[ P_i = \frac{(k_i + c_1)}{\sum_{r=1}^{m_t} (k_r + c_1)} \]  

where \( m_t \) is the total number of participants at time-step t.
c) Participation of existing participants on existing projects: Existing participants may also choose to work on projects created by other participants. In the bipartite network, this can be viewed as addition of links between the existing participant and project nodes. The number of links to be added is determined based on the total number of links ($L$) to be obtained at the end of the process. As a simplifying assumption, equal numbers of links are added throughout the process. Again, the links are added between projects and participants using the preferential attachment mechanism. An existing project is selected using the probability given in expression (15) and an existing participant is selected based on the probability given in expression (16). The process is repeated for the number of links to be added during a time-step.

The inputs to the model are $N = 1907$, $M = 1217$, $L = 4614$. The output of the model is bipartite network of people and projects generated over predefined time-steps. The objective is to simulate the global network structure simply based on the local behaviors. The preliminary focus is on evaluating the degree distribution of the global network structure. It is observed that the overall degree distribution follows a scale-free degree distribution. The parameters $p_a$, $c_0$, and $c_1$ are calibrated to bring the degree distribution of the bipartite network close to those in the Drupal network. The values of the parameters that resulted in the degree distribution that is closest to the real network are: $p_a = 0.15$, $c_0 = 0.55$ and $c_1 = 0.75$. Figure 30 displays the flow chart of the preliminary model.
In the first model, there are three limitations. 1) In each period, only one new participant joins in the network. However, the time period that we analyze of the Durpal network is one year. So within a year, new participants joining in the network are always more than one. 2) The probability that an existing participant create a new project is based on the project/people ratio which is a constant. However, due to the evolutionary properties of the network, it can be understood that this probability will always change at different period. 3) The number of links added between existing participants and existing people is based on the expected number of link which increase linearly. However, in the
analysis of the Drupal network, it reveals that the increase of number of links is always followed by the exponential law. Hence a refined model is built to solve these limitations.

The refined model shares the same process in each time period as the preliminary model. But it has three modifications. In each period, the number of new participants is pre-determined from Drupal data.

![Figure 30 – Modeling Data from Drupal](image)

\[ y = -0.253\ln(x) + 0.9917 \]  
\[ y = 0.06e^{0.1842x} \]  

**a)** Here we define four probabilities \( P_1, P_2, P_3, P_4 \) which are used in the second model.

- \( P_1 \) (Figure 31 a) is the probability that developer creates a new project in a certain time period.
- \( P_2 \) (Input) is the probability that a new developer creates a new project in a certain time period.
- \( P_3 \) is the probability that an existing developer creates a new project in a certain time period.

\[
P_3 = \frac{P_1 \times \text{Total Num of People} - P_2 \times \text{Num of New People}}{\text{Num of Existing people}} \quad (17)
\]

- \( P_4 \) (Figure 30 b) is the probability that an existing participants won’t work in any projects any more after they first join the community.
b) At each time period, the probability is changing instead of a constant value. In the Drupal data, at each period, the number of new project can be counted. In the model, the number of new projects created by new participants can be calculated before analysis of existing participants creating new projects. Hence, the number of new projects created by existing participants can be known by subtract the number of new projects created by new participants from the number of new projects created. The probability that existing participants create new projects can be expressed as the number of new project created by existing participants divided by the number of existing participants.

c) The probability that existing participants work on existing projects is calculated from the analysis of Drupal data to replace the expected number of links used in the preliminary model. In the analysis of Drupal data, it is found that the probability of existing participants won’t work in any projects any more after they first join the community (Figure 30 b). So the probability the existing participants work on existing projects will be expressed as:

\[ P_5 = 1 - P_3 - P_4. \]

By executing the second model, the evolutionary degree distribution properties of people and projects can be obtained as the results of outputs of the second model. At each period, the degree distributions can be created and validated whether they match the Drupal data.
6.2 Simulation Results

In this section, we are going to present the simulation results of both the two models. The first model is aimed at verifying our hypothesis on the micro behavior of individuals --- the preferential attachment rule. The second model is modified to simulate the evolution process of Drupal community by adding some of the investigated Drupal parameters.

In the previous Chapter 3.2, we stated that preferential attachment is one of the general models for creating scale-free network. This is one of the reasons we used the preferential attachment kernel in both models to keep the scale-free property of the simulated network. In Figure 31, we show the degree distribution of the first model and compare the result to the Drupal original data. We can observe that both the two trend lines of the people and project network distribution are almost overlapping. The numbers in the power equations are closed with average 4% errors on the intercepts and 3.75% on the slopes in both networks.

The two parameters \( c_0 \) and \( c_1 \) are used to adjusting the degree sequence of the network. By changing the values of \( c_0 \) and \( c_1 \), we can get different distributions through...
the simulation. With $c_0 = 0.55$ and $c_1 = 0.75$, we get a set of similar result of degree distribution above. However, other than the scale-free property, does the simulation meet other properties of Drupal community?

The Joint Degree Distribution is a matrix that calculates the possibility of cooperation between nodes with different degrees of a bipartite network. For example, in the Joint Degree Distribution, $F(2,3)=0.0043$ means that the possibility that people with degree 2 will work on projects with degree 3 is 0.43%. In Figure 32, we plot the Joint Degree Distribution in 3 dimensional spaces. The Z axis stands for the value of possibility and the X and Y axles are the degree sequences. From Figure 32 (a), we know that in the Drupal community, people with low degree are more willing to join a low degree project. This marco-behavior of thousands of nodes is not a conflict with what we introduced about the preferential attachment. Instead, with $c_0 = 0.55$ and $c_1 = 0.75$ and the preferential attachment kernel, our simulation result of the Joint Degree Distribution is almost the same with the Drupal data. By comparing Figure 32 (a) and (b), we find that not only the low-low (people with low-degree and project with low-degree) side of the matrix has a similar trend, but the position and heights of the ripples are incredibly parallel.

Figure 32 – Joint Degree Distribution of Drupal and Simulated Network
From these two verifications, our hypothesis that preferential attachment serves to be the micro-behavior rule of Drupal community is proven correct to some extent. With the preferential attachment kernel, we execute the other model to simulate the evolution of Drupal.

We have described the different between the first and second model. The modification of the second model is mainly focus on adding specific inputs which were investigated from the previous analysis of Drupal community. Figure 33 shows the result of degree distribution of participants for Year 7 to Year 9 of Drupal history. Similar to the property of Drupal community, our simulation also resulted in 3 scale-free networks from Year 7 to Year 9. If by only observing the intercept and slope at different times, a
huge error exists in the power law equations. However, by comparing the parameter from Year 4 to Year 9, we found that there is some cross matching in the degree distributions. The simulated parameters of Year 8 match the Drupal data of Year 7 and the simulated parameters of Year 7 match the Drupal Data of Year 6. This phenomenon can be explained. We speculate that the cross matching is due to the structure of the initial network. We formed the initial network by linking a person to a project randomly with a certain possibility. In fact, this may not the case of what the Drupal community is. Founders of Drupal might take on more responsibilities than other co-founders or assistants.

In addition to degree distribution, there are various indicators of networks, some of which are discussed in Section 2.3.1. In Figure 34, we can observe that the clustering coefficient evolutions between Drupal and simulated network are following the same trend closely. One of the reasons that the clustering coefficient values in a different range may be the mismatching on the two initial values. From the average density evolution in Figure 35, we can find out the value of simulated data in Year 5 is close to the value in the Year 6 of Drupal data because of the mismatching at the beginning. Except for the initial network, we can optimize the simulation by changing the parameters \( c_0 \) and \( c_1 \) to get a convergent result. For the second model, the complexity is \( O(N) \) because we only optimizing the parameters of Year 9 network. However, we believe that the complexity will be over \( O(N^6) \) if trying to narrow the error of all parameters at 6 different networks, which is a large computational consumption.
Figure 34 – Evolution of the Simulation by Model 2
7 CLOSURE

The structure of an organization not only affects the efficiency through the flow of information between individuals, but also has an impact on the final product. Hence, it is important from a design engineering standpoint to gain an understanding of the structure and evolution of open-source communities, and, ideally, to design mechanisms to direct their evolution.

7.1 Summary of the Thesis

The first contribution of this thesis is an approach for analyzing the structure and evolution of open-source communities and its application to a specific OSS community (Chapter 3). The objective of this analysis is to extract patterns in community structures and trends in their evolution. An understanding of such patterns and trends, and their impact on the product evolution can potentially be used for guiding open-source projects. Despite the differences between software and hardware development, the underlying mechanisms of community formation and evolution in open-source software and hardware are similar. The analysis is carried out using an existing open-source software development community because of the availability of data from successful projects. It is observed from the analysis that the specific community structure also displays a scale-free topology, which has been observed in a large number of other social, technical and biological networks. It is emphasized that the conclusions about the community are valid for the Drupal community. Whether the patterns observed in this community represent all open-source communities is still an open question that requires further investigation.

The second contribution is a predictive model that simulates the evolution of the community (Chapter 4). The model is solely based on the predicted local behaviors of the
individuals to generate global community structures. The local behaviors are modeled based on the hypothesis that the observed scale-free network topology is a result of preferential attachment of individuals to high-degree projects. A variation of the basic preferential attachment model for bipartite networks serves as a good starting mechanism for modeling the evolutionary dynamics of the network. The generated networks closely approximate the actual Drupal networks in terms of basic network characteristics such as degree distributions, connectedness, and clustering coefficient. The model is applicable, and can be calibrated, for other networks with similar degree distributions.

We acknowledge that open-source processes are complex in nature. Each project has its own peculiar characteristics. Hence, just like any other model, we envision various levels of abstraction of the model – from very general to very specific. The general models are good for a broad range of projects but fail to capture the specific details of a project. The appropriate granularity of a model depends on the goals of the modeling activity. The specific models contain specific knowledge that can be used for detailed modeling of a specific project. In this thesis, the focus is not on modeling a particular project in complete detail but to extract general underlying mechanisms that drive the bottom-up evolution of these communities. It is these general underlying mechanisms that can be used to guide a wide range of such projects. These mechanisms can range from providing incentives to the participants to change their behavior to supplementing the self-organized teams with traditional structured teams.
7.2 Answering the Research Questions

In Chapter 1.4, we proposed three research questions which led the whole work in this thesis. For each of the research questions about open-source community, one hypothesis was made based on our speculation and the support of previous work done by other researchers. After processing different methods on the research, the answers of the three research questions are included in the thesis and the summary of each is as follow:

**Q1: What is the structure of open-source community?**

The structure of open-source community is reflected through the social network metrics. In Chapter 3, we concluded most of the important characteristics of the structure of Drupal community. The combination of low average distance and high clustering coefficient denotes that Drupal network is highly connected, which is also call “small world” phenomenon. The scale-free property is presenting in the fact that the degree distribution follows the power law. This mean only a few nodes have a high connectivity to others. Through the study, we also list the possibilities that the Drupal community has different hierarchy organization of developer roles.

**Q2: What is the evolutionary behavior of these communities?**

We analyzed the evolutionary behavior through six snapshots of Drupal community in 9 years time. In Chapter 4, we gave social network metrics for each of the snapshots. From the evolutionary view of some metrics, we depicted the development of the structure of Drupal community. For example in Chapter 4.2, the decline of degree centrality indicated that some project leaders start lots of projects while others start only a few in the initial stage of Drupal. Moreover, we found out that the Drupal community has
been continuously presenting scale-free property during 9 years. Finally, the visualization method gave us the information of the macro-behavior of Drupal community.

**Q3: What are the underlying local behaviors that result in their evolutionary behavior?**

In Chapter 6, we developed a preferential-attachment-rule based model to verify our previous hypothesis. We found out that the simulation results are similar to part of our original analysis. The simulated network follows the power-law all the time. The joint degree distribution of the simulated network matches the original one, which also indicates that large amount of people with low-degree join projects with low-degree. Also, we simulated the same trend for some of the social network metrics, such as average density of people network. All these results strongly support that preferential attachment rule is one of the underlying local behaviors of Drupal community.

### 7.3 Future Work

There are various avenues for further extension of this work. First important aspect is the analysis of other projects to determine similarities and differences between different open-source projects. As more data about open-source hardware becomes available, the similarities and differences between open-source software and hardware can be explored. The model can be further improved by refining the local behavior of individuals. The information about the amount of contributions from each participant is available in many projects. This information can be used to further refine the model. The use of other network growth models such as rate equation method, probabilistic, and generation function techniques [95] can be investigated. As discussed in Chapter 1, the co-evolution of community structure and product structure is an important avenue for further research.
As a final comment, Social network analysis (SNA) is a fast growing field with a focus on analyzing structures of communities and their impact on processes on these networks such as transmission of information. Design research community can benefit from the tools developed in SNA to understand and account for the organizational (community) structures within product development, particularly in emerging bottom-up self-organized approaches such as open-source.
REFERENCES


50. Xu, J., Christley, S., and Madey, G., 2005, "The Open Source Software
Community Structure", in NAACSOS2005, Notre Dame, IN.

Multiple Networks I. Blockmodels of Roles and Positions," American Journal of
Sociology, 81(4), pp. 730-780.

52. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y., 2002,
"Evolution Patterns of Open-Source Software Systems and Communities", in
IWPSE ’02: Proceedings of the International Workshop on Principles of Software

Communities", in Proceedings of the International Conference on Open Source

Source Code as a Social and Technical Artifact", in GROUP ’05 ACM, Sanibel

55. Howison, J., Inoue, K., and Crowston, K., 2006, "Social Dynamics of Free and
Open Source Team Communications", in Proceedings of the IFIP 2nd
International Conference on Open Source Software, Boston, USA.


