

# **OPEN SOURCE SOFTWARE: ECONOMIC AND SOCIAL ANALYSIS**

**WU JING**

*(M.Sc, Hong Kong University of Science and Technology*

*B. Eng, Northwestern Polytechnical University, China)*

**A THESIS SUBMITTED**

**FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**DEPARTMENT OF INFORMATION SYSTEMS**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2008**

## Acknowledgements

When I am writing acknowledgements, the thesis writing will be finished soon. I am not sure whether I am happy or sad. I am happy that the thesis will be finished soon, but I am sad that PhD life will be over! Most of my PhD friends told me that pursuing PhD was very boring and tough. Many old friends do not understand why I am still a student, an old student! Every time chatting with me through MSN, they always begin at “Hi, did you graduate?” Is PhD study boring? No, I do not think so! Although it is difficult, it is not boring!

Four years ago, I came to Singapore to pursue PhD. It is for my boyfriend (husband now), who was also studying PhD in Singapore. I came into a new area again: information systems. I studied electrical engineering in undergraduate, and switch to economics in master. Of course, the first year was very tough because I had to sit in many courses which I was not familiar with. Finally, the grades were very bad. However, I still want to remember those days: at central library, my husband and I were studying together. At the beginning of second semester, Candy became my supervisor. She is very kind and optimistic. When I met difficulties, she always helped and encouraged me. When I was confused with research topics, she inspired me to find what I was interested in. Later, I chose the topic about open source. Since I learned a little knowledge on economics, I tried to use some economics methodology

to solve research questions in information systems field. Then, I started to investigate the competition issue between open source and proprietary software. The qualify exam was coming. Because of poor presentation and lack of preparation, I failed. Candy did not blame me, but supported me to revise the model and applied for the next QE. Of course, I passed the QE on the second time; otherwise, I could not sit here to write these acknowledgements. During this period, Candy and my husband gave me much support. They let me feel safety when I met difficulties. Therefore, although it was very tough in this period, I was still happy, and enjoyed the life.

At the end of second year, Candy and I submitted a paper to a conference: ECIS. I was very lucky that this paper was accepted. It was my first paper. I cannot use words to describe how excited I was. Thanks to Candy, for your effort in this paper and your effort in instructing me! In June 2006, I went to Europe to attending conference and see my husband who was at Paris at that period. Thanks to IS department and NUS, for the financial support for the travelling! Because of ECIS, NUS, IS department and Candy, I realized my dream in advance: having a trip to Europe! Is PhD boring? No. after a tough period, I got more happiness. I am enjoying PhD life!

Later, Candy left Singapore to US. I totally understood how desirable she wanted to be together with her husband. In order to go on PhD study, Candy introduced Ivan as my temporary supervisor. Ivan is an amazing gentleman. He is serious and strict, but kind and warmhearted. Although he was very busy and could not instruct me too

much time, he can give me much helpful advice in every meeting. During this period, I smoothly passed the thesis proposal exam. Thanks to Ivan and Candy, for your kind supervisions!

In March and May 2007, I submitted one paper to PACIS and two papers to ICIS. Fortunately, one was accepted by PACIS and one was accepted by ICIS. I was so lucky! I had a chance to go to New Zealand and Canada, which I did not image before! Is PhD life boring? No. Thanks to IS department and NUS, for the financial support for the travelling again! During this period, Khim Yong becomes my supervisor, and Ivan and Bernard become my thesis committee members. Khim Yong is a young and smart guy. Although he is very thin, he is full of energy. He is an expert in econometrics in our department. He gave me much helpful suggestions in the research, especially in analysis of the econometrics models. When I prepared the presentation in ICIS, he squeezed his valuable time to listen to my rehearsal. Khim Yong, I am always appreciating your kindly help! Bernard, the head of our department, is very amiable and always has smile on the face. He is very very busy, but still can squeeze time to meet with me to discuss my research. Thanks to Bernard, for your kind support to my research.

So far, June 2008, I still believe that my PhD life is rich, meaningful and full of surprise. I am very happy during these four years. Besides Candy, Ivan, Khim Yong, Bernard, IS department, and NUS, I also thank professor Teo Hock Hai. Your course brings me to

IS area, and lets me know what is IS, and how to do IS research. I thank my best friends: Qihong, Guo Rui, Shaomei, and Yang Xue. Our friendships make me much happier and more optimistic.

To my family, thanks to mother and father, you always give me everything selfless. It is you who give me such a happy and wonderful life!

At last, to my sweet heart, Kang Kai, I do not thank you here by words, but I would like to use my whole life to love you, care you and be together with you!

Wu Jing  
June 2008

# Table of Contents

<b>ACKNOWLEDGEMENTS .....</b>	<b>I</b>
<b>TABLE OF CONTENTS.....</b>	<b>V</b>
<b>SUMMARY .....</b>	<b>VIII</b>
<b>LIST OF TABLES.....</b>	<b>XI</b>
<b>LIST OF FIGURES .....</b>	<b>XII</b>
<b>CHAPTER 1. INTRODUCTION.....</b>	<b>1</b>
1.1 General Background.....	2
1.2 Three Studies.....	4
1.2.1 Evaluating Longitudinal Success of Open Source Software Projects .....	5
1.2.2 Optimal Software Design and Pricing .....	6
1.2.3 Partially Opening Source Code.....	8
1.3 Contributions.....	9
1.3.1 Evaluating Longitudinal Success of Open Source Software Projects .....	10
1.3.2 Optimal Software Design and Pricing .....	10
1.3.3 Partially Opening Source Code.....	11
References .....	12
<b>CHAPTER 2. EVALUATING LONGITUDINAL SUCCESS OF OPEN SOURCE SOFTWARE PROJECTS: A SOCIAL NETWORK PERSPECTIVE .....</b>	<b>14</b>
2.1 Introduction.....	14
2.2 Theoretical Background.....	19
2.2.1 Communication Pattern of Open Source Project Teams.....	19
2.2.2 Success of Open Source Projects.....	24
2.3 Research Model.....	26
2.3.1 Communication Pattern and Project Success.....	28
2.3.2 Project-Specific Characteristics and Project Success .....	35
2.4 Research Method.....	38

2.4.1	Project Selection .....	38
2.4.2	Measures .....	43
2.5	Results and Analysis .....	45
2.5.1	Econometric Models .....	48
2.5.2	Robustness Checks .....	54
2.5.3	Hypothesis Test.....	66
2.6	Concluding Remarks.....	71
	References .....	77
	Appendix .....	84

**CHAPTER 3. OPTIMAL SOFTWARE DESIGN AND PRICING IN THE PRESENCE OF OPEN SOURCE SOFTWARE..... 95**

3.1	Introduction.....	95
3.2	Literature Review.....	99
3.3	Model 1 .....	102
3.3.1	Market is fully covered.....	105
3.3.2	Market is not fully covered.....	106
3.3.3	Analysis of results.....	107
3.3.4	The impact of network effect.....	110
3.4	Model 2 .....	113
3.4.1	Model Setting.....	114
3.4.2	Optimal Pricing of Commercial Software .....	120
3.4.3	Optimal Design of Commercial Software .....	124
3.4.4	Comparative Static Analysis.....	128
3.4.5	Welfare Analysis.....	132
3.4.6	Overall Analysis.....	133
3.5	Concluding Remarks.....	134
	References .....	138
	Appendix .....	141

**CHAPTER 4. PARTIALLY OPENING SOURCE CODE: A NEW COMPETITIVE TOOL FOR SOFTWARE FIRMS..... 147**

4.1	Introduction.....	147
4.2	Literature Review.....	149
4.3	The Model.....	151
4.3.1	Case 1: Duopoly Market Dominated by Firm A and Firm B.....	153
4.3.2	Case 2: There is a Competing Pure Open Source Product.....	167
4.4	Concluding Remarks.....	175
	References.....	179
	Appendix.....	182
	<b>CHAPTER 5. CONCLUSION AND FUTURE WORK.....</b>	<b>186</b>
5.1	Evaluating Longitudinal Success of Open Source Software Projects.....	186
5.2	Optimal Software Design and Pricing.....	188
5.3	Partially Opening Source Code.....	189



## Summary

This thesis applies social network analysis and economic theory and methodology in Information Systems research to study three issues associated with open source software projects and their applications in the software industry.

The growing popularity of open source software has been garnering increasing attention not only from practitioners in the industry, but also from many academic scholars who are interested in examining this phenomenon in a rigorous in-depth manner. To date, as a testament to the popularity of open source software, there are also numerous open source projects being hosted on many large online repositories. While some of these open source projects are active and thriving, some of these projects are either languishing or show no developing activities at all. This observation thus begs the important question of what are the influential factors that impact on the success or failure of open source projects. As such, to deepen our understanding of the evolution of open source projects, the first study aims to analyze the evolution of open source projects from inception to success or failure by using the theoretical lens of social network analysis. Based on extensive empirical data collected from open source development projects, we study the impact of the communication patterns of open source projects on the outcomes of these projects, while accounting for project-specific characteristics. Such an approach thus incorporates both the supply side (developers) and the demand side (end users) factors. Since communication patterns may change

with time, success or failure of open source projects is transient. Therefore, we observe the changes in communication pattern of each project team over extended periods.

Open source software has become an increasingly threatening competitor to traditional proprietary software. In the second study, we examine the competition between proprietary and open source software by considering consumer's taste. In order to capture the effect of consumer's taste on the firm's strategy, we first use a one-dimensional Hotelling model, and then analyze a two-dimensional vertical differentiation model. In particular, we seek to answer how commercial software vendors should optimally set the price and design its product when competing with the open source product.

The popularity of open source not only poses competition to proprietary software producers, but also brings to light a new competing strategy: opening part of the source code. Many industry practices suggest that participating in open source projects may bring profit to software firms. In the third study, we model the competition between two profit-oriented firms, and analyze the optimal strategy of the firm that uses open source as a competing strategy. We seek to answer: Why does a for-profit firm open up its commercial product? How much should the firm open to achieve most profit? What is the best competition structure of the market when both firms choose their best competitive strategies? Furthermore, we consider the impact of the presence of a

competing pure open source product. We seek to find how the presence of open source affects the firms' strategies in the duopoly competition model.

## List of Tables

Table 2.1 Descriptive Statistics of All Variable .....	46
Table 2.2 Main Estimation Results .....	53
Table 2.3 Robustness Check: Simultaneity Bias .....	54
Table 2.4 Robustness Check: Endogeneity Bias .....	57
Table 2.5 Robustness Check: Development Activity .....	62
Table 2.6 Robustness Check: Project Popularity .....	63
Table 2.7 Summary of Hypotheses Test .....	71
Table 2.8 Examples of Centrality .....	85
Table 2.9 Communication Pattern Data of Project “YUL_Library” .....	88
Table 2.10 Full Estimation Results .....	88
Table 3.1 The Optimal Price, Demand and Profit of Commercial Firm .....	124
Table 3.2 Boundary Solution of the Software Firm .....	126
Table 3.3 Comparative Statistics of Optimal Solutions .....	129
Table 4.1 Pre-optimal Strategy in Duopoly Market .....	157
Table 4.2 Pre-optimal Strategy in Duopoly Market .....	159
Table 4.3 Relationships of Optimal Profit With Benefit ( $s$ ) And Cost ( $c, d$ ) .....	162
Table 4.4 Relationships of Optimal Price With Benefit ( $s$ ) And Cost ( $c, d$ ) .....	163
Table 4.5 Relationships of Optimal Demand With Benefit ( $s$ ) And Cost ( $c, d$ ) .....	165
Table 4.6 Optimal Strategy When There is a Competing Pure Open Source Product .....	170
Table 4.7 Comparative Static Analysis .....	171

## List of Figures

Figure 2.1 Research Model .....	27
Figure 2.2 Data Extraction .....	40
Figure 2.3 Project Selection .....	42
Figure 2.4 Communication Graphs of Project “YUL_Library” .....	47
Figure 2.5 Fluctuation of Communication Pattern Measures .....	48
Figure 2.6 Histograms of Selected Variables .....	50
Figure 2.7 Communication Graphs of Project “YUL_Library” .....	86
Figure 2.8 Communication Graphs of Project “YUL_Library” (cont’d) .....	87
Figure 3.1 The Hotelling Model .....	103
Figure 3.2 Locations of Open Source and Proprietary Software .....	116
Figure 3.3 Product Space (left) and Consumer Space (right) .....	118
Figure 3.4 Demand of Commercial Product ( $\alpha > 45^\circ$ ) .....	121
Figure 3.5 Demand of Commercial Product ( $\alpha < 45^\circ$ ) .....	123
Figure 3.6 The range of optimal location when $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ .....	125
Figure 3.7 When OSS Locates at the Shaded Area, Boundary Solution Achieves .....	126
Figure 3.8 Relationship Between Profit and Functionality of OSS .....	132
Figure 3.9 Optimal Location Curve of Proprietary Software .....	145
Figure 4.1 Distribution of Users .....	155
Figure 4.2 Market Share of Firm A and B .....	156
Figure 4.3 Relationships of Firm A’s Optimal Degree of Openness ( $\alpha_A^*$ ) With Benefit ( $s$ ) And Cost ( $c, d$ ) .....	159
Figure 4.4 Market Share of Firm A, B and OSS .....	169
Figure 4.5 Comparison of Consumer Surplus and Profits of Firms .....	174

## **Chapter 1. Introduction**

---

This thesis applies social network analysis and economic theory and methodology in Information Systems (IS) research to study issues associated with open source software (OSS) projects and their applications in the software industry. The popularity of the OSS phenomenon has been attracting more and more attention from both industry and academia. Many traditional software companies have either enrolled themselves in OSS development or applied OSS strategy. Meanwhile, academic researchers have also paid great attention to the OSS phenomenon. They have examined various aspects of OSS, social, economic and organizational. These studies have made use of different theories and methodologies in its field to explain the OSS phenomenon. This thesis will examine interesting OSS issues from social and economic theoretical perspective.

## 1.1 General Background

IS discipline is broad and has been defined in different ways. It has been depicted as “the study of the interaction of development and use of IS with organizations” (Cushing 1990), and “understanding what is or might be done with computer and software technical systems, and the effects they have in the human, organizational and social world” (Avgerou and Cornford 1995). Since IS research is a relative new research area, the theories and methodologies from other fields such as economics, psychology, social science, and computer science have been widely applied in IS.

The application of social network theory or social network analysis (SNA) in the field of IS can help to better understand the impact of social factors on IS applications. SNA has emerged as a key technique in many fields such as sociology, anthropology, statistics, mathematics, information sciences, education, and psychology. SNA aims to understand the relationships between people, groups, organizations, and other types of social entities (Granovetter 1973; Wasserman et al. 1994; Wellman et al. 1998) by description, visualization, and statistical modeling.

Economics has been widely accepted as one of the main IS research disciplines. It has been deemed as one of the four reference disciplines of IS together with computer science, management science and organization science (Benbasat and Weber 1996).

Various economic theories, such as game theory and economic models of organizational performance, have been applied to explain, predict and solve IS

problems.

Recent years have seen a rapid growth of OSS. OSS refers to those programs “whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or the modified program without having to pay royalties to previous developers” (Wheeler, 03). OSS involves a copyright-based license to keep private intellectual property claims out of the way of both software innovators and software adopters, while preserving a commons of software code that everyone can access (O’Mahony 2003). It is typically created within OSS projects, often initiated by an individual or a group that wants to develop a software product to meet particular needs.

Since the first OSS was developed by Richard Stallman (GNU) in the 70’s, there have been a large number of open source applications, ranging from common office suites such as StarOffice, to database (mySQL) and thousands of specialized scientific applications. Nowadays, OSS has been widely adopted for different purposes, including, for example, web servers (Apache, iPlanet/Netscape), e-mail servers (Sendmail), programming languages (Perl, Java, Python, GCC, Tk/TCL), and operating systems (Linux, BSD Unix). More than 65 percent of all public websites are operated on the open-source Apache web server; 80 percent of the world’s e-mail traffic is managed by Sendmail; and nearly 40 percent of large American corporations make use of the open-source GNU/Linux operating system (Weber 2004). Not only popular



in the software market, OSS phenomenon has also attracted greater attentions from academia, especially from the IS field. IS researchers have applied different theories and methodologies to investigate various issues of the OSS phenomenon, including competition between OSS and proprietary software, licensing problems of OSS, coordination in OSS, and survival of OSS projects. They have already achieved many results which are helpful for industry and research.

This thesis applies social network analysis and economic theory and methodology to study issues associated with OSS projects and their applications in the software industry. I will briefly introduce them one by one in the following section.

## **1.2 Three Studies**

It is a fact that OSS exists and is popular in the software market. It is also a fact that only a small proportion of OSS has survived in the market. This phenomenon attracts us to investigate the survival of the OSS projects in the evolving periods. However, the existence of OSS must affect the profitability of proprietary software, which spurs us to examine the competition between OSS and proprietary software. The software companies not only face the competition from OSS, but also from their colleagues. The software firms may use open source as the competitive strategy to compete with others. How can the firm use the open source as a competing weapon?

## 1.2.1 Evaluating Longitudinal Success of Open Source Software

### Projects

Although a few OSS projects, such as Linux, Apache, MySQL and PHP, have achieved extraordinary success and are among the most prominent software used in the technology industry, there are lots of OSS projects which are lackluster with no developing activity at all. Many die at the beginning, while others survive, but with little momentum behind them (Thomas and Hunt 2004). This begs questions of how to deal with the growing pains for the OSS projects: Why do some OSS projects achieve success while many others don't? What are the factors that could influence the success or failure of the OSS projects? To deepen our understanding of the OSS, it is essential to explore the factors that have contributed to its success or failure. In the first study of this thesis, we will examine OSS success through the social network perspective. The main objective is to identify the presence and significance of factors in predicting the success of an OSS project. We seek to provide insights to the following questions: (1) whether the success of open source projects is correlated to the social structure of the development teams, i.e. the communication pattern of the project team; and (2) what is the impact of communication pattern on the survival of open source projects in a long term. Based on real-world empirical data, we study communication pattern of open source project team, as well as considering project-specific characteristics, on the project success. We collect data from SourceForge.net, the largest repository of open source projects, which is widely used in most OSS studies. The details of this study are

described in Chapter 2.

### **1.2.2 Optimal Software Design and Pricing**

With the free of charge open source products available in market, many commercial firms have been dealing with continued pressure and competition from the open source world. OSS makes source code publicly available for free usage and modification, including bug fixing and customizing features. Ever since the burgeoning of OSS, it has attracted more and more attention from individual users and organizations due to its “free of charge” and “freedom of distribution and modification”. Without a doubt, the profitability of a commercial software publisher is affected (if not threatened) when the consumers are offered with an alternative free option other than the proprietary software. In order to make profit and maintain their dominance in the software market, the commercial software publisher must design different business and economic strategies to respond to the emergence of open source software. The second study in this thesis is to answer the key question about how a profit-seeking software firm should compete with open source software. Although competition has been the classic research topic in economic literature, the competition between open source and proprietary software has the following distinct features that deserve further analysis: (i) traditional duopoly competition model studies the equilibrium of two profit-making firms while open source software is free of charge and can't be made for profit by itself; (ii) traditional competition models normally study the optimal pricing while in case of

software competition, the software producers has two arms to fight with competition – pricing and product design. For instance, if the commercial product is quite similar to open source products, the commercial firm faces fierce competition; but if the proprietary software is highly differentiated, the product might appeal to a certain part of the market; (iii) software products exhibit positive network externalities, which further complicates the decision of optimal price and product design.

We adopt two models to analyze the competition between open source and proprietary software. We first employ a one-dimensional stylized Hotelling model to study the optimal pricing and design of proprietary software in the presence of competitive open source software. We address the following research questions: (1) what is the impact of open source software's positioning (design) on the optimal price, design and profit of the proprietary software; (2) how is social welfare affected by the positioning of open source software; (3) what are the firm's optimal strategy and profit when there's positive network effect. In this model, we use one dimension to represent consumer taste. We did not give the details of the consumer taste. In the second model, we try to analyze the consumer taste in a specific way: functionality and usability. In this model, we study the optimal design and pricing strategies for a monopoly commercial software firm to compete with open source software. The commercial software producer has to invest in a certain amount cost to achieve a certain level of usability and functionality for its product. We establish a two-dimensional vertical differentiation model to derive the optimal price and design of the commercial software product given the

characteristics of the open source software. The details of this study are described in Chapter 3.

### **1.2.3 Partially Opening Source Code**

With regard to the continuous competition between the open source and the proprietary camp, the age-old saying still works: if you can't beat him, join him. For the proprietary software publishers, it is not advisable to treat open source only as the competitor. Instead, proprietary firms can learn from it, absorb the advantages of it, and make use of it. Some industry practitioners have come to realize that proprietary software can leverage the open source idea and profit from it (Taft, 2005). Adam Fitzgerald, director for developer solutions at BEA Systems Inc., of San Jose, California, said at the panel at the BEAWorld conference: "You need to start thinking about what an open-source solution can do for you and identify best practices and best-of-breed open-source technology. This notion of blending open source solutions is what we see customers already using." "Combining the best open source software and the best commercial software will give you the best solution," said by Zhongyuan Zheng, vice president for R&D at Beijing-based Red Flag Software Co. Ltd., China's premier Linux vendor and maker of Red Flag Linux. More and more commercial firms have realized that the adoption of an open source strategy can bring strategic advantage in the aggressively competitive environment. Netscape, for example, open up its web browser and give out of the code for free as the Mozilla open source project. The other big firms like IBM

and Sun also keep up with this trend and open part of their commercial software codes. The open source movement in the software industry, in which commercial software publishers open part of their source codes, attracts a lot of attention from academia and industry. Among those papers discussing the competition between OSS and proprietary software, although some researchers looked into the incentives for commercial firms to participate in OSS development (Lerner and Tirole 2001), few studies examined the open source as the commercial firm's competing strategy to maximize profit. Thus, in the third study of my thesis, we will study the competition between two profit-oriented firms and analyze the model that when open source is as a software company's competing for-profit strategy, (1) why a for-profit firm opens up its commercial product; (2) how much the firm should open to achieve most profit; (3) what the equilibrium and best competition structure of the market are when both firms choose their best competitive strategy. The details of this study are described in Chapter 4.

### **1.3 Contributions**

This thesis applies social network analysis and economic theory and methodology in Information Systems research to study issues associated with open source software projects and their applications in the software industry.

### **1.3.1 Evaluating Longitudinal Success of Open Source Software**

#### **Projects**

This study is among the first to explore open source project success through the lens of social network perspective. Through social network analysis of empirical data collected from open source projects, we study the impact of the communication patterns of open source projects on the outcomes of these projects, while accounting for project-specific characteristics. Such a novel approach incorporates both the supply side (developers) and the demand side (end users) factors. We observe the changes of communication pattern of each project across extended periods, and investigate the evolving success of open source projects by looking at the dynamic impacts of communication patterns.

### **1.3.2 Optimal Software Design and Pricing**

The objective of this study is to answer the key question about how a profit-seeking software firm should compete with open source software. Although competition has been the classic research topic in economic literature, some distinct features are examined in this study. Traditional competition models normally study the optimal pricing while in case of software competition, the software producers has two arms to fight with competition – pricing and product design. This study not only investigates the optimal pricing of the software firm, but also finds the optimal product design.

### **1.3.3 Partially Opening Source Code**

In this study, instead of focusing on the competition between open source and proprietary software, we study the competition between two profit-oriented firms, and analyze the model that when open source is as a software company's competing for-profit strategy. There are very few papers discussing the situation when some software firms open part of their code for profit reasons to actively compete with other software firms. This study gives us the idea that software firm can improve its competing advantage by using open source strategy.



## References

- Avgerou, C., Cornford, T. "Limitations of information systems theory and practice: A case for pluralism," In Falkenberg et al., *Information Systems Concepts: Towards a Consolidation of Views*, London: Chapman and Hall, 1995, 130-143.
- Benbasat, I., Weber, R. "Research commentary: rethinking 'Diversity'," *Information Systems Research*, 7(4), 1996, 389-399.
- Cushing, B.E. "Frameworks, paradigms, and scientific research in management information systems," *The Journal of Information Systems*, 4(2), 1990, 38-59.
- Granovetter, M. "The strength of weak ties," *American Journal of Sociology*. 78, 1973, 1360-1380
- O'Mahony, S. "Guarding the commons: how community managed software projects protect their work," *Research Policy*, 32, 2003, pp, 1179–1198.
- Wasserman, S. and Galaskiewicz, J. *Advances in social network analysis: research in the social and behavioral sciences*, SAGE Publications, Thousand Oaks, Calif, 1994.
- S. Weber. *The Success of Open Source*, Harvard University Press, Cambridge, 2004.
- Wellman, B., and Berkowitz, S.D. *Social Structures: A Network Approach*, Cambridge University Press, Cambridge, 1998.

Wheeler, D. A. "Why open source software/free software (OSS/FS)? Look at the number!" Online resource: [http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html), December, 2003.

D. K. Taft "The key to open-source success," eWeek.com article. December, 2005.

Thomas and Hunt "Open source ecosystems," *IEEE Software*, (32:1), 2004.

# Chapter 2. Evaluating Longitudinal Success of Open Source Software Projects: A Social Network Perspective

---

## 2.1 Introduction

Recent years have seen a rapid growth of open source software (OSS). Ever since the first OSS was developed by Richard Stallman (GNU) in the 1970's, a multitude of open source applications have been developed, ranging from office productivity software such as StarOffice, to database and thousands of specialized scientific applications. Nowadays, OSS has been widely adopted for different purposes, including web servers (Apache, iPlanet/Netscape), e-mail servers (Sendmail), programming languages (Perl, Java, Python, GCC, Tk/TCL), and operating systems (Linux, BSD Unix). It is reported that more than 65 percent of public websites are now backed by the open-source Apache web server; 80 percent of the world's e-mail traffic is managed by Sendmail;

and nearly 40 percent of large American corporations make use of the open-source GNU/Linux operating system (Weber 2004).

What is OSS? OSS refers to those programs “whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or the modified program without having to pay royalties to previous developers” (Wheeler 2003). OSS involves a copyright-based license to keep private intellectual property claims out of the way of both software innovators and software adopter, while preserving a commons of software code that everyone can access (O’Mahony 2003). It is typically created within OSS projects, often initiated by an individual or a group that wants to develop a software product to meet their own needs.

The growing popularity of OSS has garnered increasing attention not only from practitioners in the industry, but also from academic scholars who are interested in examining this phenomenon in a rigorous in-depth manner. Various case studies have contributed to a better understanding of the OSS phenomenon. Lakhani and Hippel (2003) considered the nature and the functioning of the community of developers of the Apache software. Hertel et al. (2003) focused on factors determining the level of engagement in the Linux project. Krogh et al. (2003) analyzed the strategic process by which new individuals joined the community of developers of FreeNet, a peer-to-peer network of information distribution. These studies shed new light on how large

communities of developers arise, work and coordinate to achieve the success of an open source project. However, previous case studies are limited to large and popular projects only. While in-depth examinations on such large and popular projects are crucial to better understand how communities work effectively, findings from such studies may not be sufficiently representative of the open source community in general.

Several large open source projects have achieved extraordinary success and are among the most prominent software used in the technology industry. However, many open source projects have been lackluster with few or no development activities at all. Many flounder at the beginning, while others survive, but with little momentum behind them (Thomas and Hunt 2004). The failure of a large number of open source projects begs the following key question: What factors could influence the longitudinal success of open source projects? Specifically, since communications among developers are essential to the survival of the project, how does the communication pattern of the development team affect the evolving success of an open source project? In addition, the definitions and measurements of project success from the developers' and the end users' perspectives are different, how does this difference affect the impact of the other influential factors on a project's success? To deepen our understanding of OSS, it is essential for Information Systems (IS) researchers to study these questions theoretically and provide insights to the business world.

The open source community is characterized by the voluntary participation of software

developers collaborating over the Internet with the aim to produce license-free software. The developers have been creating value through developing and spreading new knowledge and capabilities, fostering innovations, and building and testing trust in working relations, relying heavily on information and communication technologies to accomplish their tasks (Powell et al 2004). For the development teams, to achieve their objectives and successfully complete their tasks, information must be effectively exchanged. Thus, communication and coordination have been found to be two major aspects that significantly affect the performance of such teams (Johansson et al. 1999; Maznevski and Chudoba 2001). OSS development is a complex socio-technical activity, requiring people to interact with each other. Thus, it is interesting to study the communication patterns of open source development teams to investigate the relation between coordination and communication characteristics (i.e., the social network attributes) of OSS project teams and the evolving outcomes of open source projects.

While others have studied the determinants of open source success (e.g., Fershtman and Gandal 2004; Comino et al. 2005; Sen 2005; Colazo et al. 2005; Stewart et al. 2006; Grewal et al. 2006), this study is among the first to explore open source project success through the lens of social network perspective. Through social network analysis of empirical data collected from open source projects, we study the impact of the communication patterns of open source projects on the outcomes of these projects, while accounting for project-specific characteristics. Such a novel approach thus incorporates both the supply side (developers) and the demand side (end users) factors.

As we know, communication patterns may change with time and thus success or failure of OSS projects is transient. It is therefore important to examine the dynamic impacts of communication patterns on project success such that we can assess the long term sustainability of OSS projects. Thus, in this study, we observe the changes of communication pattern of each project across an extended period of 13 months, and investigate the evolving success of open source projects by looking at the dynamic impacts of communication patterns.

Following the panel data analysis methodology, we obtain model estimation results from Three-Stage Least Squares accounting for both period and project fixed effects, as well as carry out several robustness checks of different models. The effects of communication pattern, i.e., project centrality, project density, and leadership centrality, on project development activity and popularity respectively are examined and uncovered by our research model. Based on our results, the impacts of communication patterns on project success considered from the demand side and the supply side are different. It implies that project managers can reap the benefits if they can structure their project teams with care. Therefore, according to the objectives of projects, a proper and planned control for the communication among team members is crucial for the survivability of the open source projects.

This study is organized as follows. Section 2.2 introduces the theoretical background of communication patterns and explains why and how it can be applied to open source

project studies. We provide definitions of key concepts such as the success of open source projects and the communication pattern. Then we propose the research model and the hypotheses in Section 2.3. We describe the operational details of our empirical research, such as criteria for project selection and measures of constructs in Section 2.4, followed by discussions of the results in Section 2.5. Finally, Section 2.6 concludes this study with directions of future research.

## **2.2 Theoretical Background**

In this study, we propose that the social structure of open source project teams may play a critical role in the success of open source projects. Based on social network theory, we investigate the interactive communications among open source contributors in order to find the impact of communication patterns on open source project success. In this section, we define key concepts such as success, social structure, social network analysis, and communication pattern in the open source environment.

### **2.2.1 Communication Pattern of Open Source Project Teams**

Open source developers collaborate mainly over the Internet. The advent of information and communication technologies provides instantaneous global accessibility for the open source community. Software development is a complex socio-technical activity. The developers of an open source project collaborate via



interactions or communications in the form of email exchange, message boards, etc. (Sawyer 2004). The communication and interaction among individuals and groups form the network of relationships inside the project team. To better understand the impact of such communications on the success of open source projects, we employ the social network analysis (SNA) method, which helps to identify the prominent patterns in such networks, trace the flow of information (and other resources), and discover potential relationships between the social structure and the final product, i.e. the software system (Kidane and Gloor 2007).

SNA (also called social network theory) has emerged as a key technique in many fields such as sociology, anthropology, statistics, mathematics, information sciences, education, and psychology. SNA aims to understand the relationships between people, groups, organizations, and other types of social entities (Granovetter 1973; Wasserman et al. 1994; Wellman et al. 1998) by description, visualization, and statistical modeling. It models social relationships in terms of nodes and ties. Nodes represent the individual actors or groups within the network, and ties or links show interactions or exchange of information flows between the nodes. In the context of open source projects, nodes are the developers, and ties are the interactions (i.e., communications) between the developers. In the field of Information Systems, previous literatures which focused on OSS research, have shown that social networks operate on many levels and play a critical role in determining the way of solving problems, running organizations, and the degree to which individuals achieve their goals. Hippel and Krogh (2003) argued that

open source development has become a significant social phenomenon, and that developers and users form a complex social network via various electronic communication channels on the Internet. Madey et al. (2002) conducted an empirical investigation of the open source movement by modeling OSS projects as a collaborative social network and found that the open source development community can be modeled as a self-organizing social network. Xu et al. (2005) explored some social network properties in the open source community to identify patterns of collaborations.

Social structure, a term frequently used in social theory, refers to entities or groups in definite relation to each other, to relatively enduring patterns of behavior and relationship within social systems (Scott 2002). The social structure of an open source development team describes how people interact, behave and organize in the community. Investigating social structure is a useful way to understand team practice such as coordination, control, socialization, continuity and learning (Freeman 1979; Scacchi 2002). Software engineers have realized that there are inevitable linkage between the group performance and the social structure of the development team. Therefore, a better understanding of the social structure can help with the development planning (Scacchi 2002). Crowston and Howison (2005) interviewed a member of the Apache Foundation's incubator team at ApacheCon 2003<sup>1</sup>. The incubator team

---

<sup>1</sup> The Apache foundation is a prestigious umbrella organization for teams developing free and open source software. It has created an incubator to ensure that the projects which seek to join the Foundation are of sufficient quality and longevity. <http://incubator.apache.org>

indicated that they were concerned that overly heavy reliance on a small number of (possibly corporate funded) developers was a major threat to the sustainability of the project and thus to the suitability of the project for Apache incubation (Crowston and Howison 2005). The study of social structure helps to identify the reasons for such concerns since it provides an assessment measure of finding the crucial members as well as their importance with regard to the project.

The communication pattern describes the structure of interactions during communication. It can be characterized by several attributes. According to social network theory, the centrality and density of a group are related to its efficiency of problem solving, perception of leadership and the personal satisfaction of participants (Scott 2002). The concepts of density and centrality refer to different aspects of the overall “compactness” of the network (Scott 2002). Density describes the general level of cohesion in the network while centrality describes the extent to which this cohesion is organized around particular focal points. Centrality and density, therefore, are important complementary measures (Scott 2002) of the communication pattern.

Density measures how closely a network is connected, which in turn determines the readiness of a group in response to changes in processes and outcomes. It is defined as the percentage of ties that exist in a network out of all possible ties.

Centrality<sup>2</sup> can be defined on an individual or overall level for a network. The centrality of an individual node refers to the number of direct links to other nodes in a network. If we define the link between nodes as communications, a person with a high centrality represents a major channel of information exchange. In some sense he is a focal point of communication, at least with respect to others who has contact with him. At the opposite extreme is a point of low centrality degree. The occupant of such a position is likely to be seen as peripheral. His position isolates him from direct involvement with most of the others in the network and cuts him off from active participation in major communication processes. Thus, the centrality measure indicates whether a group member is “in the thick of things” (Freeman 1979; Mullen et al. 1991). In order to track the influence of the project leader(s), we examine the individual centrality measure of project leader(s) since the centrality of the leader(s) indicates the prestige and influence of the leader(s) in the project team (Hanneman and Riddle 2005).

One can also define the centrality of a network as a whole. Project centrality, centrality of an entire project team, captures the inequality of the developers’ contributions to the project: high score of project centrality implies that the power of individual developers varies rather substantially, and overall, positional advantages are rather unequally distributed in this network. Social network theory (Leavitt 1951) suggests that the speed and efficiency of a network in solving problems are related to the inequality of the developers’ contributions to the project.

---

<sup>2</sup> The detailed (mathematical) definitions and examples of centrality are given in the Appendix.

### **2.2.2 Success of Open Source Projects**

Apart from licensing terms, OSS has other distinct features that are not seen in proprietary software. OSS development frequently depends on volunteers coordinating their efforts without the governance of a common organizer, and the end product is often provided for free (Feller and Fitzgerald 2000). Therefore, unlike traditional firm-driven endeavors, open source projects are not always driven by direct profit motives (Lakhani and Wolf 2003). The success indicators of commercial software such as market share, on time and on budget delivery cannot be readily applied in the OSS setting. In the OSS environment, there is usually no pre-determined deadline, a priori budget, or a set of specifications (Scacchi 2002), and market share of OSS is difficult to assess. Therefore, a set of different indicators are necessary to define the success of open source projects.

Success is a subjective concept and therefore it is not always clear on how to define success. Raymond (1998) defined successful OSS projects as those characterized by a continuing process of volunteer developers fixing bugs, adding features and releasing software “often and early”. Since a large number of OSS projects are abandoned by their developers, it is critical to attract contributors on an on-going basis to keep the project sustainable (Markus et al. 2000). Crowston et al. (2003) explored success measures in the Information Systems literature and suggested a portfolio of success measures, including measures of the development process. Subsequently, Crowston et

al. (2004) analyzed four success measurements by using data from SourceForge.net and suggested that a project that attracts developers, maintains a high level of activity, fixes bugs and has many users downloads can be described as successful. There are some other scholars advocating different success measurements. For example, Colazo et al. (2005) singled out two particular items from those success measures: the number of developers joining in a project and the relative level of the developers' productivity while they were engaged in the project (i.e., contribution). Comino et al. (2005) utilized the development stage (i.e., planning, pre-alpha, alpha, beta, stable and mature) of a project as the representation of the level of success of a project. Fershtman and Gandal (2004) considered an alternative definition of system success based on output per contributor. They examined how the type of license, the programming language, the intended audience and other factors affect the output per contributor in OSS projects. Sen (2005) made use of project popularity (defined by Freshmeat.net) as the measure for OSS's installation base. Stewart et al. (2006) adopted user interest as the measurement of OSS project success. In particular, they used the development activity to measure the development-oriented success. Grewal et al. (2006) adopted two kinds of success measures: the number of CVS<sup>3</sup> commits as an indicator of successful technical refinement, and the number of downloads over the life span of a project as the indicator

---

<sup>3</sup> Concurrent Versions System (CVS) is a program that lets a code developer save and retrieve different development versions of source code. It also lets a team of developers share control of different versions of files in a common repository of files. This kind of program is sometimes known as a version control system. CVS was created in the UNIX operating system environment and is available in both Free Software Foundation and commercial versions. It is a popular tool for programmers working on Linux and other UNIX-based systems.

of market or commercial success.

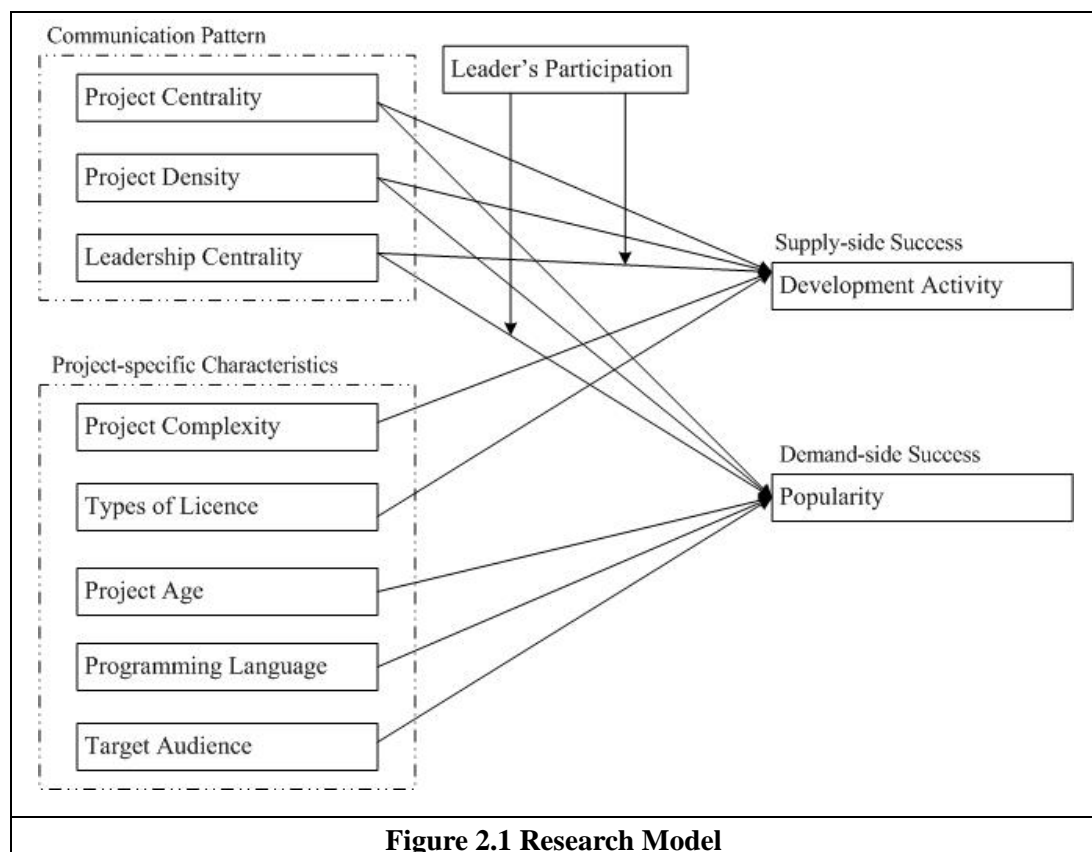
In our study, we consider success from both the supply side (developers) and the demand side (end users). Since open source development relies on voluntary input, attracting and motivating contributors are key factors for its success. In other words, development activity is a key indicator of project success: high development activity shows that the developers in the project continuously contribute to the project; the project will evolve until it has no development activity at all. On the demand side, project popularity is a key measure of the project's success: high popularity shows that there are many users using or are interested in using the open source software. On the other hand, an OSS project will cease to exist or progress if there is no demand or if no one makes use of the end product for an extended period.

In summary, our research is based on the theoretical fields of social network analysis, and we measure OSS success on both the developer and the end user side. To the best of our knowledge, we are among the first to simultaneously study the success of OSS projects from both the supply side and the demand side, while exploring the determinants of open source project success through a social network perspective of the communication patterns within OSS projects.

## **2.3 Research Model**

This study focuses on the communication pattern of open source development teams.

Specifically, we propose hypotheses with regard to how communication patterns may affect the success of open source projects. We define the following constructs that capture the communication pattern of an open source project: (1) project centrality, which measures the inequality of the developers' contributions in the project, and (2) project density, which measures the closeness of a network and its readiness to respond to changes, and (3) leadership centrality, which measures the influence and prestige of the project leader(s). In addition, we use the level of development activity and project popularity to measure the degree of success from the supply side and the demand side respectively. Our research model is shown in Figure 2.1.





### **2.3.1 Communication Pattern and Project Success**

*Project Centrality and Development Activity:* Project centrality measures the difference or inequality of contributions among developers, i.e., it examines whether there is an outstanding group of contributors in the project (Freeman 1979). Past research in social networks has shown that centrality is an important measure of group performance (Freeman et al. 1980). The investigation of project centrality can shed light on whether the inequality of the developers' contributions affects the success of the project. When the project centrality is high, the power of individual developers varies rather substantially. Social network theory suggests that networks with high centrality have the advantage of speedy and flexible information diffusion within the network (Cummings and Cross 2003). In a network with high centrality score, there are certain developers who have access to more resources of the network than others in the network. These "core" developers are responsible for exchanging information and allocating resources among the team members. In most cases, these core developers are the most capable developers in the team. They filter out less meaningful messages while distributing the most useful information and allocate resources to its best usage. This increases the efficiency and quality of the communication among the developers and enhances the team members' access to resources and information. Therefore, the development process, which involves collaborative tasks such as debugging, document writing, upgrading, patching, and consulting, can be better handled with better resource allocation and higher quality delivery. In the strategic management

literature, good coordination within development teams (which helps to attract users, resources, and collaborators) is considered as a key determinant of software development success (Kidane and Gloor 2007). Since higher project centrality is associated with better organization and more efficient information exchange, we propose that a more centralized project is likely to achieve higher level of development activities:

*Hypothesis 1: Project centrality will positively affect the level of project development activities.*

*Project Centrality and Popularity:* In the high centrality project, the power of individual developers varies rather substantially. The higher the project centrality, it is more likely that there are more linkages between the core developers and the other contributors (Hanneman and Riddle 2005). The core developers take charge of exchanging most information or resource among the project team members. Thus, the communications are largely dependent on the core developers. Oftentimes however, open source developers simultaneously participate in more than one open source project or are affiliated to other commercial projects. The developers usually have a limited amount of time to contribute the project (Hinds and Lee 2008). Once the core developers lose interest or pay insufficient efforts to the projects, it will greatly affect the coordination of the development team as well as the response to the end users' requests. Open source users, especially new users, usually put more emphasis on the

continual support and maintenance provided by the project teams. Heavy reliance on a single developer or a few developers threatens the continual adoption or usage of OSS from end users who seek continuous support and maintenance from the OSS project. Inconsistent, insufficient maintenance and support of open source projects by developers typically generate negative word of mouth which reduces the reputation of such projects. Given the increased likelihood of negative word of mouth and reputation in the OSS community, it is thus likely that the demand popularity of such projects decrease over time. Therefore, we propose that highly centralized OSS projects will be less likely to attract end users, i.e., less likely to increase the level of popularity of such projects:

*Hypothesis 2: Project centrality will negatively affect the level of project popularity.*

*Project Density and Development Activity:* In a network of software developers, a higher density indicates a greater degree of interaction among the members and thus closer collaboration among members (Hanneman and Riddle 2005). However, a higher project density makes the dissemination of knowledge more time-consuming, because information and knowledge needs to travel through the extended hierarchies of the project team. The higher density, the more information is repeatedly communicated within the project team. An example of a three-developer project illustrates this problem: suppose developer A has communications with B, B with C,

and C with A, then developer C may obtain the same information from both A and B. In projects with many developers, there may be more occurrences of repeated information exchange. The efficiency of the communication is therefore substantially reduced in projects with high density. Previous literature indicates that effective communications among the team members is a key factor to project success (Suchan and Hayzak 2001). For open source development teams, to achieve the objectives and to successfully complete their development tasks, information must be effectively exchanged (Powell et al. 2003). Thus, we propose that the density of a project will negatively affect the level of development activities of a project:

*Hypothesis 3: Project density will negatively affect the level of project development activities.*

*Project Density and Popularity:* According to the definition of density, project density determines the readiness of a group in response to changes (Hanneman and Riddle 2005). Higher density indicates a greater degree of interaction among the members in the process of making decisions (Hanneman and Riddle 2005; John Scott et al. 2005). In the case when a specific member cannot contribute, for e.g., the developer could not promptly respond to the feedbacks from end users such as bug fixing, support or feature requests, other members can take charge of his or her responsibility. The coordination of the development team as well as the response to the end users' requests will not be greatly affected. Open source users usually put more attention on the continual

support and maintenance provided by the project teams. Sufficient maintenance and support of open source projects by developers typically generate positive reputation of such projects. Hence, the software users might also favor a project with high density. Therefore, a high density project will attract more end users and subsequently leading to an increased the level of popularity of the project:

*Hypothesis 4: Project density will positively affect the level of project popularity.*

*Leadership Centrality and Development Activity:* A project manager plays the key role of coordinating overall project development activity. As Lerner and Tirole (2002) pointed out, the project manager should carry out some critical tasks such as attracting new programmers, ensuring an efficient division of the project into modules, allowing contributors to perform their tasks independently from the rest of the contributors, and managing conflicting views and approaches among participants. In a project with higher leadership centrality, the manager has higher stature and more influence on the project team (Mehra et. all 2006). In high manager centrality projects, the managers can attract and communicate with many developers, and those developers may actively exchange information with their managers because of the manager's stature and influence. Under such communication structures, developers derive the majority of necessary information and resources from the managers, and can perform their tasks independently from the other developers. The project manager thus serves as a pivotal conduit for information communication among the project team (Mehra et. al

2006). Mehra et. al (2006) also suggest that “when group leaders are centrally located within their groups, they can more successfully mobilize and direct group action toward the accomplishment of important group goals, and thereby enhance the objective performance of their groups.” Therefore, we propose that project leadership centrality will positively influence the level of development activity of the project:

*Hypothesis 5: Project leadership centrality will positively affect the level of project development activities.*

Furthermore, a project manager tends to simultaneously participate in more than one open source projects or is affiliated to other commercial projects. Through exposures in the multiple projects, she may get peer recognition or signal her talent to different colleagues. This is the so-called ego gratification incentive, which is one kind of signaling incentive defined by Lerner and Tirole (2002). Economic theory (e.g., Holmstrom 1999) suggests that the stronger the signaling incentive, the more visible the performance to the peers; the higher the impact of effort on the performance; and the more informative the performance about talent (Lerner and Tirole 2002). Thus, the more projects the project manager participates in, the more visible the manager’s performance will be to other developers. As stated above, when the manager’s centrality is high, the manager has high stature and influence in this project (Mehra et. al 2006). The visibility of such positive reputation of managers to developers in other projects is thus enhanced if a manager participates in multiple projects. Therefore, we

propose that a leader's heightened participation in other projects will yield a positive effect on the level of development activities for project teams with high leadership centrality:

*Hypothesis 6: Leader's heightened participations in other projects will yield a positive effect on the level of project development activities for projects with high leadership centrality.*

*Leadership Centrality and Popularity:* In a project with higher leadership centrality, the manager has higher stature and more influence in the project team (Hanneman and Riddle 2005), so that the manager takes charge of exchanging most information or resource among the project team members. However, heavy reliance on a single developer threatens the continual adoption or usage of OSS from end users who seek continuous support and maintenance from the project. A project manager typically has a limited amount of time and efforts to contribute to the projects (Hinds and Lee 2008). Once the manager could not focus attention to the project, it will greatly affect the coordination of the development team as well as the response to the end users' requests. These outcomes typically generate negative word of mouth which reduces the reputation of such projects. Given the increased likelihood of negative word of mouth and reputation in the OSS community, it is thus likely that the demand popularity of such projects among OSS end users decrease over time. Therefore, we propose that project leadership centrality will negatively influence the level of project popularity:

*Hypothesis 7: Project leadership centrality will negatively affect the level of project popularity.*

Since the open source managers (including the developers) usually simultaneously participate in more than one project or are affiliated to other commercial projects, there are effort and time constraints on part of the individual developers' fixed amount of resources and effort (Hinds and Lee 2008). The more projects the manager participates in, the more time and efforts are needed to contribute to the different projects. Thus, as much anecdotal evidence has indicated, heavily reliance on a single project manager threatens the survival of the open source projects. This may largely deter the users who want to seek continual consistent support and maintenance from OSS project teams. Therefore, we propose that a leader's heightened participations in other projects will yield a negative effect on the level of project popularity for projects with high leadership centrality:

*Hypothesis 8: Leader's heightened participations in other projects will yield a negative effect on the level of project popularity for projects with high leadership centrality.*

### **2.3.2 Project-Specific Characteristics and Project Success**

Apart from the communication pattern, project-specific factors may also influence the success of OSS projects. Project-specific characteristics considered in this study



include type of licenses, project complexity, programming language, project age, and target audience.

Lerner and Tirole (2005) suggested that the restrictiveness of license protects the developers from being exploited by the commercial software firms by limiting the privatization of their intellectual products. From the viewpoint of the open source developers, commercialization of open source projects is undesirable because it can reduce the visibility of the developer's contribution and reputation, which have been discovered to be one key incentive to participate in open source development (Lerner and Tirole 2005). Therefore, commercialization may drive away developers. Thus, we propose the restrictiveness of licenses may play a positive role on the level of OSS development activities. In addition, complex projects may deter some developers who would not like to pay much time on this project. OSS project developers, especially in the small OSS project, usually did not get monetary compensation from the projects. If the project is too complex, they need to spend much time on familiar with the project. Thus, we expect that a more complex project is likely to achieve a less level of development activities.

*Hypothesis 9A: A project with a restrictive license will be likely to achieve a higher level of development activities.*

*Hypothesis 9B: A more complex project will be likely to achieve a lower level of development activities.*

From the end users' perspective, software written in more popular programming languages may be more popular among end users, since more people can modify (or customize) the open source software. In addition, the popularity of an OSS project is likely to increase with a longer history of existence, because the longer the project has been developed and distributed in the open source community, the more users can find and adopt the OSS software. Finally, some open source projects are specifically developed for particular user groups. Apparently, software targeted at the general end users may appeal to more users, and thus leading to increased project popularity. Thus, we propose the following hypotheses:

*Hypothesis 10A: A project written in a popular programming language will be likely to achieve a higher level of popularity.*

*Hypothesis 10B: A project with a longer time history will be likely to achieve a higher level of popularity.*

*Hypothesis 10C: A project targeting at end users will be likely to achieve a higher level of popularity.*

## **2.4 Research Method**

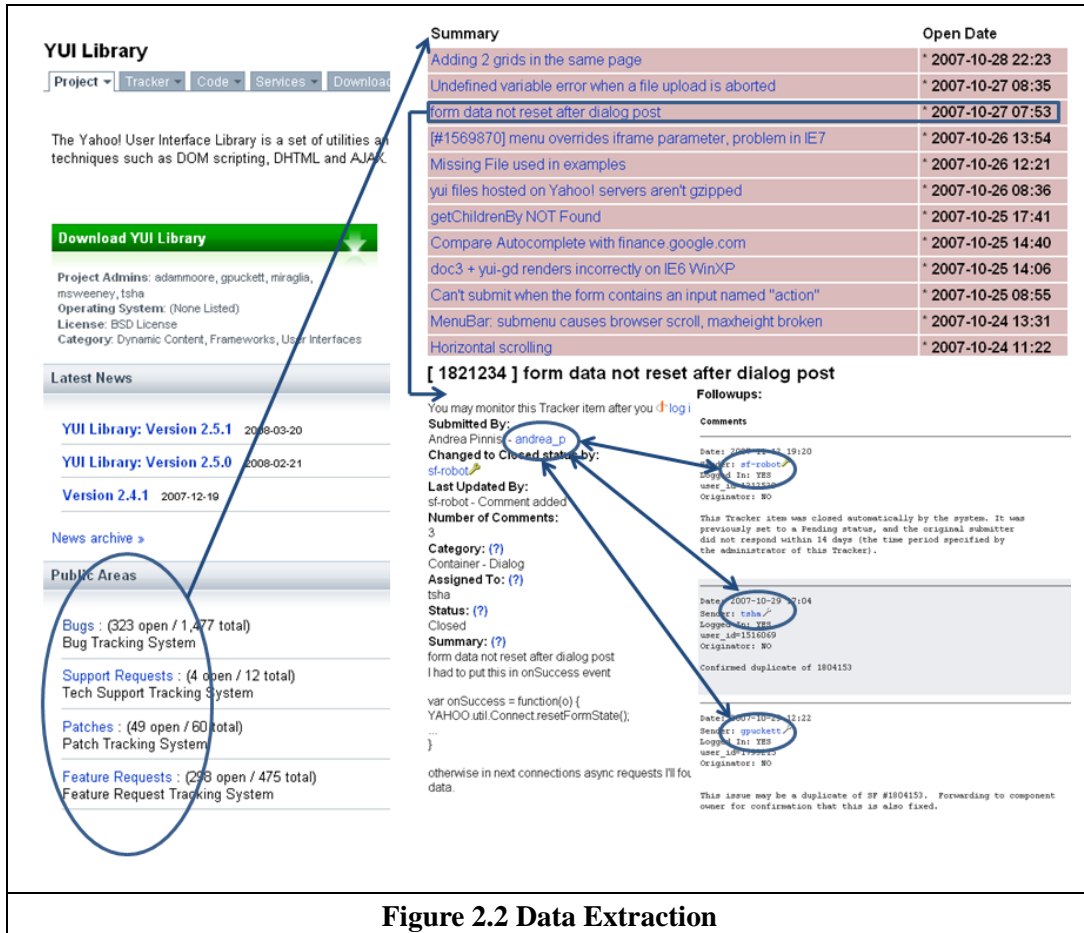
The objective of this study is to investigate the longitudinal impacts of communication patterns of open source teams on project success. We collect data of various OSS projects over an extended period of 13 months and utilize cross-sectional time-series panel data analysis methods (Wooldridge 2003, Greene 2003). In this section, we elaborate the operational details such as project selection and measurement of indicators.

### **2.4.1 Project Selection**

As with most empirical studies on open source projects, the data is collected from SourceForge.net, which is the world's largest online repository of open source applications. At the start of our data collection (in November 2006), SourceForge.net hosted 133,029 open source projects on a wide variety of areas, and had 1,425,354 registered developers. SourceForge.net also provides useful tools to control and manage open source development. It offers a variety of services including hosting, mailing lists, bug tracking, patch tracking, support request tracking, feature request tracking, message boards, file archiving, and other project management tools. SourceForge.net provides a large sampling population of open source projects with extensive details, and thereby is the best site to collect data on open source projects' development activities and attributes.

In order to investigate the communication patterns of the project teams, we observe and analyze the developers' interactions through bug, patch, support request, and feature request (BPSF) tracking systems hosted on SourceForge.net. These tracking systems enable users and developers to report and discuss BPSF. Each report includes basic information about the BPSF as well as their correspondences that deal with bug fixing, patches updating, support and feature requests responding in time sequence.

Figure 2.2 presents an example of the procedure for collecting data on the pattern of communication. First, on project's summary page, we can see the links of bugs, patches, feature requests or support requests systems. Second, by clicking the link (e.g. bugs), we can see the list of bugs. Third, checking the details of each bug, we can find who submitted the bug and who responded to this bug. We define that the submitter and respondents have the interaction, which will be recorded in the sociomatrix. Each sociomatrix contains interaction information of each project per month. A sociomatrix is a standard data representation for a network analysis (Wasserman and Faust 1994). The sociomatrix has a row and a column for each individual, and the cells of the matrix count the number of interactions from one individual to another. It can be constructed by using the popular social network analysis tool of Ucinet 6.0.

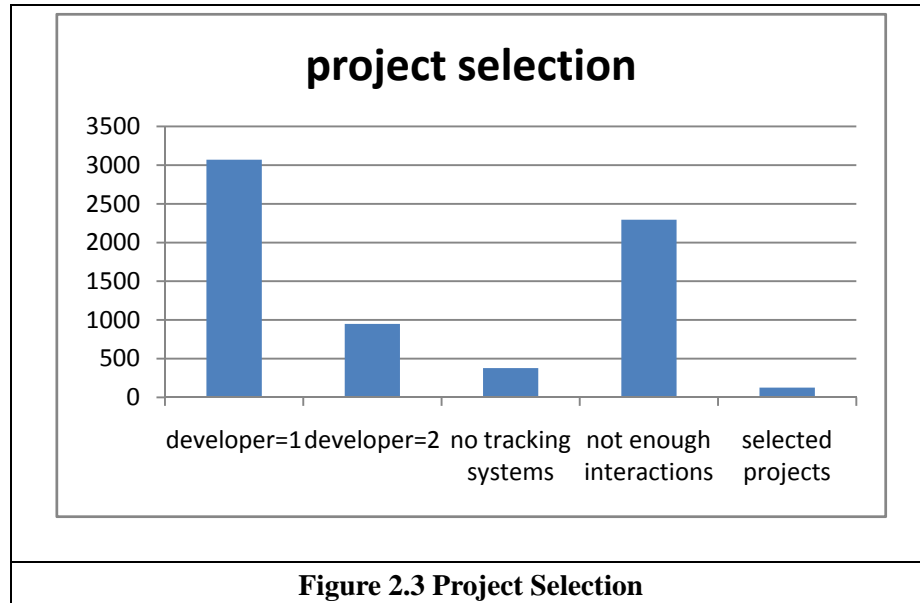


**Figure 2.2 Data Extraction**

Not all projects on SourceForge.net are suitable for our study. Three criteria are adopted to select useful projects: projects are selected from top 7000 ranked projects; projects have at least three developers; and there are enough interactions to ensure that each sociomatrix is equal to or larger than  $3 \times 3$  matrix. We choose projects with at least three developers because we are interested in team communications, instead of individual basis or dyadic interactions, which are not suitable for analyzing communication patterns. Although many projects have more than three developers, yet some projects do not make use of the SourceForge's BPSF tracking system, which leaves us impossible to track the communications; some projects do not have enough

bugs, or patches, or support requests, or feature requests to establish a non-null socio-matrix. We select only the top-ranked 7000 projects because in top 1000 projects, ninety projects are useful; from top 1000 to top 2000, twenty-one projects are useful; from top 2000 to top 4000, ten projects are useful; from top 4000 to top 7000, only five projects are useful. Therefore, we believe that there are few useful projects after the top-ranked 7000 projects.

In the top ranked 7000 projects, 6815 projects were examined because the remaining 185 projects were not ranked by the website. As shown in Figure 2.3, there were 3069 projects with only one developer; there were 948 projects with two developers; 378 projects did not have (bug, patch, support or feature request) tracking systems on SourceForge.net; 2294 projects did not have enough interactions. By excluding the unsuitable projects, the final sample was 126 projects. (The number of developers of these 126 projects ranges from 3 to 149.) In order to analyze the dynamic impacts of communication patterns on project success, we monitor each project over an extended period. Starting from 01 November 2006, we captured communication records (BPSF reports) of each project every month until 30 November 2007. We thus obtained data on 126 open source projects in 13 months.



Projects in our data samples are right-truncated, because all the projects are still active by the end of data collection date (31 Nov 2007). Since the projects are selected from top 7000 ranked projects at the beginning of our data collection process, most selected projects were in the stable, relatively productive phases of project team evolution. Hence, it is not surprising that all the selected projects are still active by the end of the data collection date. In addition, there are two projects whose registration dates (26 Jan 2007 and 04 Aug 2007) were later than the initial data collection date (01 Nov 2006). The other 124 projects' registration dates were earlier than the initial data collection date. Hence, except for the two projects mentioned earlier, there were 124 projects in our data sample that were left-truncated. For most projects (124 out of 126 projects) in our data set, with left and right truncation, the window of our observational period of 13 months is about a one-year snapshot of the projects' entire life cycle. However, for our

observation period beyond the initial starting phases of the project teams' formational beginnings, the performances of these projects are likely to be stable, relatively productive and can at least maintain a certain level of development activity and level of demand from end users. Through investigating a project's communication patterns during the stable periods beyond the formational stages, we can more confidently generalize our research results into helping us to examine how the project teams work in the stable periods, and further uncover factors impacting the long term sustainability of OSS projects. Therefore, studying the projects' communication patterns during the intermediate stable periods (between the starting and ending periods) helps researchers to understand and explain what factors influence the sustainability of these OSS projects.

#### **2.4.2 Measures**

Projects success is measured from both the supply side and demand side. For the supply side, development activity is calculated as the average number of total tracks and file releases. Number of total tracks includes sum of bugs, patches, support and feature requests. For the demand side, popularity is measured by the formula<sup>4</sup>: “number of web hits  $\times$  (1+ subscription)”, in which “subscription” is measured by level of accepted donations of each project, and “web hits” are measured by the number of web traffic

---

<sup>4</sup> We redefine the formula according to the measure of popularity given by the popular open source host: <http://freshmeat.net>.



visitations as recorded on SourceForge.net. Number of web hits indicates the flow of users. More users visiting the project's website indicate a high level of popularity for the project. In addition, from the perspective of economic theory, rational users will invest on promising products with good potential to flourish in the future. A project with a high level of donations indicates that many users may regard it as a promising up-and-coming project. Therefore, the level of donations can also indicate the popularity of a project. In order to capture popularity more comprehensively, we adopt a composite measure<sup>5</sup> of popularity as shown in the above formula.

We use project centrality, project density and leadership centrality to describe communication patterns. Project centrality can be measured by project degree centrality. Project degree centrality describes the inequality or variance of developers' contribution in the network (Freeman 1979). Project density is defined as the percentage of ties that exist in a network out of all possible ties. A density of 1 implies that every actor is connected to every other actor. A density of 0 implies that no actor knows any other actor. Leadership centrality indicates the stature and influence of the project leader, which can be measured by centrality of the project administrator or average centrality of the project administrators if there is more than one administrator. In order to moderate the effect of the leadership centrality, we include the moderating variable of the extent of leader's participations in other projects. It is defined as the

---

<sup>5</sup> We also apply single measure (web hits and subscription respectively) in the estimation. The regression results show that the composite measurement is more comprehensive.

average number of projects that project administrators (within a project) participate simultaneously. We import interaction data from SourceForge's BPSF tracking systems to create a sociomatrix for each project per month. By analyzing each socio-matrix through the popular social network software Ucinet 6.0, we can obtain the data of project centrality, density and leadership centrality.

There are several project-specific factors that may also influence the success of the project. We measure the types of licenses according to license categories classified by Lerner and Tirole (2002). A value of 1 indicates a restrictive license; 0 indicates a nonrestrictive license. Project complexity is measured by the number of software packages. Project age is defined as the months between the data collection date and the project registration date. In addition, a dummy variable is included to control for whether the project uses C/C++, which is one of the largest and most popular programming language categories on SourceForge.net. Finally, a dummy variable is employed to indicate whether the project is targeted at general end users.

## **2.5 Results and Analysis**

*Descriptive Statistics:* The development activity in terms of the numbers of tracks including bugs, patches, support and feature requests ranges from 0 to 358 and the average is 26.809. Project popularity in terms of the number of web hits ranges from 0 to  $1.480 \times 10^9$  and the average is  $1.688 \times 10^7$ . The average project centrality of our

sample projects is 49.8% and ranges from 0 to 100%. The average density is 22.8% and ranges from 0 to 100%. The average leadership centrality is 50.5% and ranges from 0 to 100%. The leaders simultaneously participate in 3.128 projects on average. The complexity of project ranges from 1 working package to 22 packages. The age of project ranges from -9.1 months<sup>6</sup> to 95.467 months. The descriptive statistics of all variables is shown in Table 2.1.

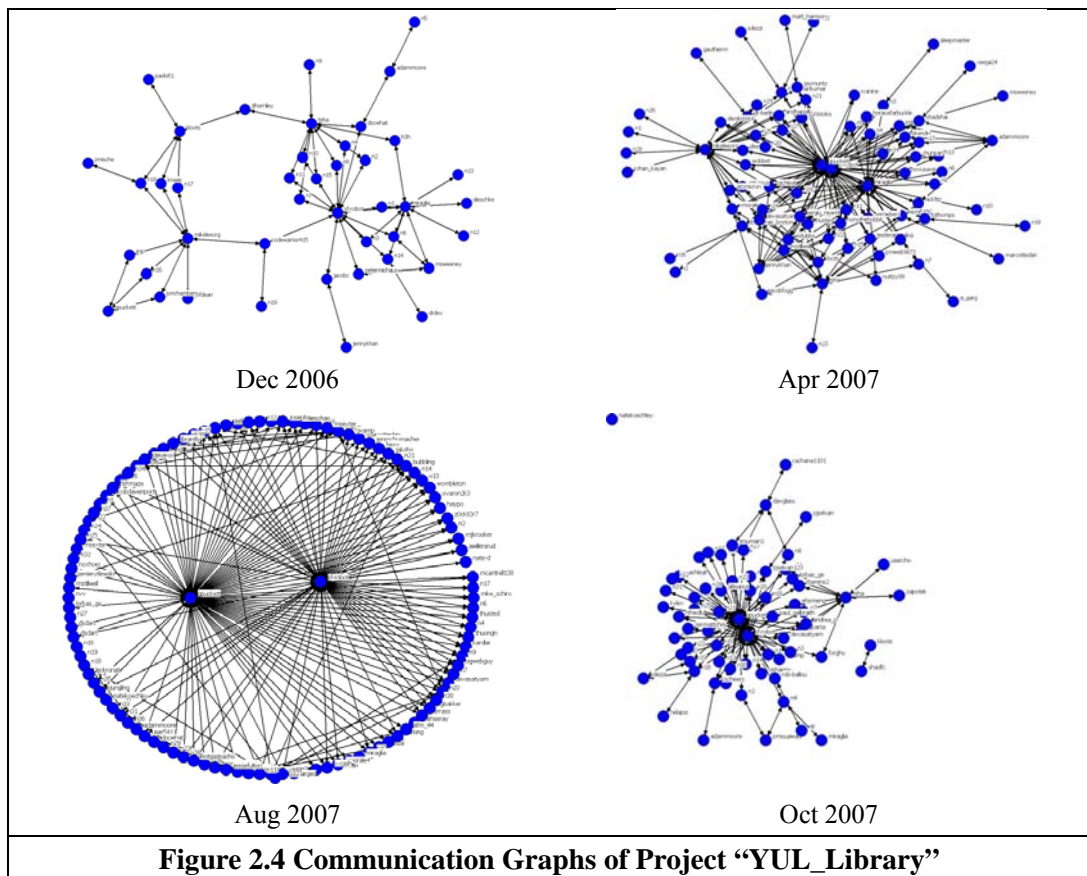
	Mean	Std. Dev.	Minimum	Maximum
Activity	26.809	34.443	0.000	358.000
Log Activity	1.152	0.526	-0.301	2.554
Popularity	$1.688 \times 10^7$	$1.170 \times 10^8$	0.000	$1.480 \times 10^9$
Log Popularity	5.004	1.615	0.000	9.171
Project Centrality	0.498	0.305	0.000	1.000
Project Density	0.228	0.180	0.000	1.000
Leadership Centrality	0.505	0.330	0.000	1.000
Participation	3.128	2.488	1.000	15.000
Complexity	4.095	3.712	1.000	22.000
Licence	0.750	0.433	0.000	1.000
Project Age	55.132	25.482	-9.100	95.467
Target Audience	0.615	0.487	0.000	1.000
Programming Language	0.614	0.487	0.000	1.000
Developers	23.213	24.319	3.000	149.000

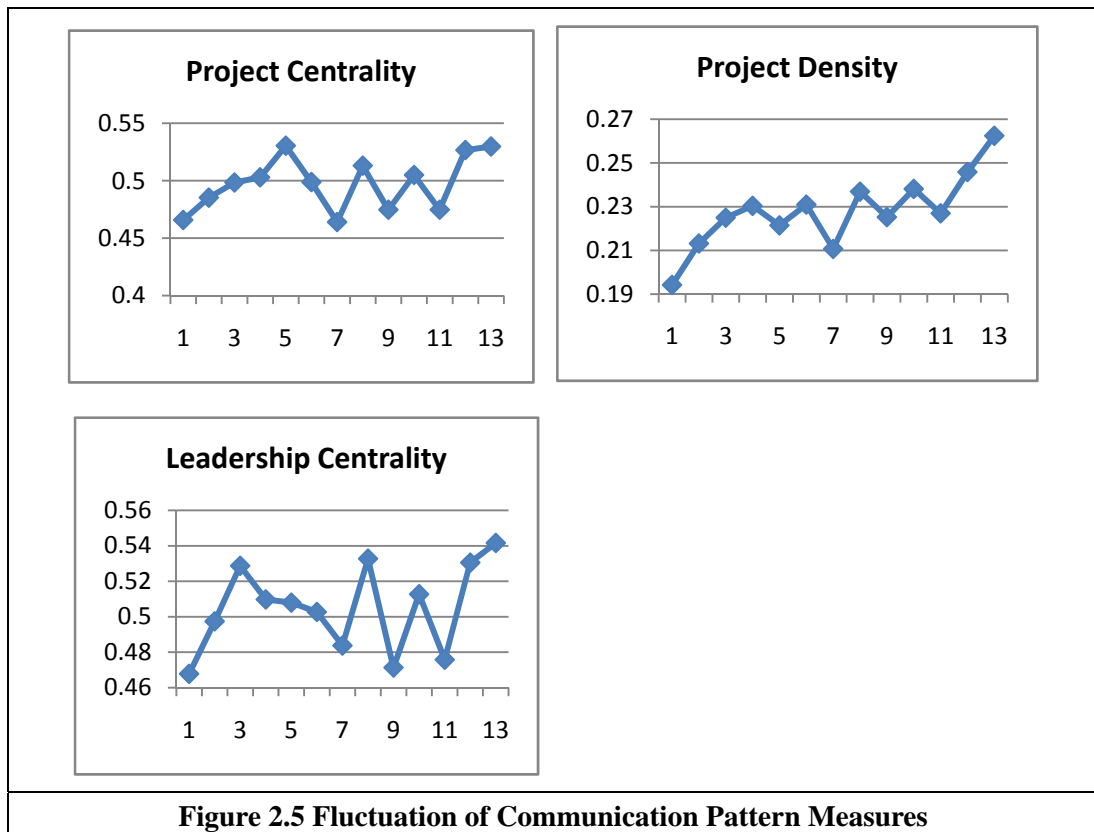
*Social Network Graphs:* In order to better understand the communication patterns, we compare examples of communication graphs of an open source project (project “YUI Library”) in four of the thirteen months<sup>7</sup> (See Figure 2.4). In Dec 2006, YUI Library’s project centrality is 33.7%, project density is 7.0%, and leadership centrality is 26.8%. In Apr 2007, project centrality is 59.6%, project density is 6.5%, and leadership

<sup>6</sup> Negative project age is because the registration date was later than the project collection date

<sup>7</sup> Because of space limitation, the whole communication graphs in thirteen months of project “YUL Library” are shown in Appendix.

centrality is 51.9%. In Aug 2007, project centrality is 73.6%, project density is 3.4%, and leadership centrality is 75.7%. In Oct 2007, project centrality is 68.8%, project density is 4.9%, and leadership centrality is 71.6%. In Figure 2.5, we draw the graphs of average project centrality, density and leadership centrality over the 13 months for all projects in our data sample. We find that distinct variations in measures of project centrality, density and leadership centrality over a 13-month period.





### 2.5.1 Econometric Models

The proposed empirical models for a project  $i$ 's development activity and popularity in time period  $t$  are:

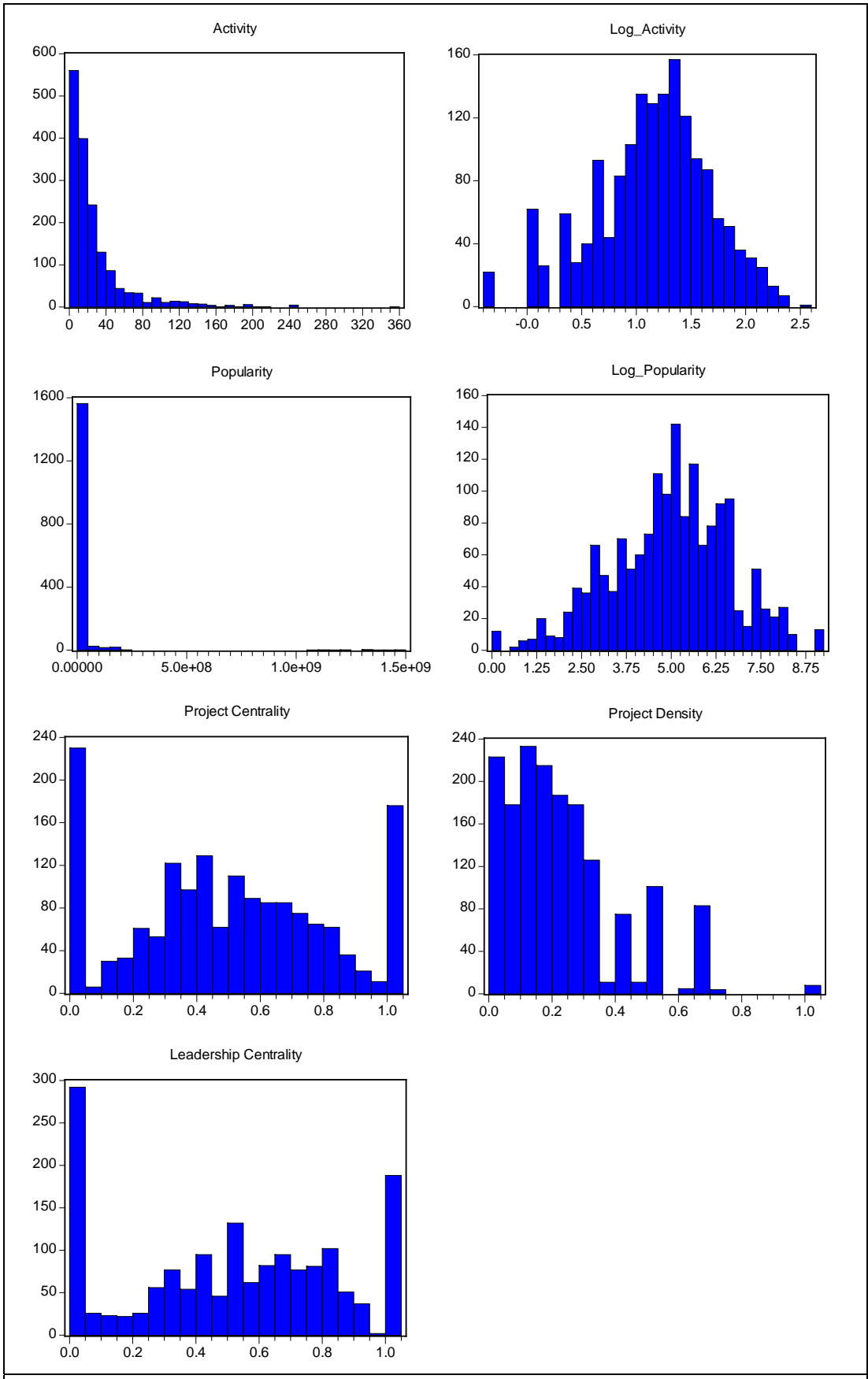
$$\begin{aligned}
 \text{Activity}_{it} = & \alpha_1 \text{Popularity}_{it} + \alpha_2 \text{Project\_Centrality}_{it} + \alpha_3 \text{Project\_Density}_{it} \\
 & + \alpha_4 \text{Leadership\_Centrality}_{it} + \alpha_5 \text{Participation}_i \\
 & + \alpha_6 \text{Leadership\_Centrality}_{it} \times \text{Participation}_i \\
 & + \alpha_7 \text{Complexity}_i + \alpha_8 \text{Licence}_i + \nu_{it}
 \end{aligned}$$

$$\begin{aligned}
 \text{Popularity}_{it} = & \beta_1 \text{Activity}_{it} + \beta_2 \text{Project\_Centrality}_{it} + \beta_3 \text{Project\_Density}_{it} \\
 & + \beta_4 \text{Leadership\_Centrality}_{it} + \beta_5 \text{Participation}_i \\
 & + \beta_6 \text{Leadership\_Centrality}_{it} \times \text{Participation}_i \\
 & + \beta_7 \text{Age}_{it} + \beta_8 \text{Language}_i + \beta_9 \text{Target\_Audience}_i + \nu_{it}
 \end{aligned}$$

Where  $v_{it}$  is the model residual error term.  $v_{it} = \mu_i + \lambda_t + \varepsilon_{it}$  in which  $\mu_i$  refers to project-specific dummy,  $\lambda_t$  refers to time-specific dummy, and  $\varepsilon_{it}$  refers to random error term.

By looking at the histograms of dependent variables (development activity and popularity) and some independent variables (project centrality, density, and leadership centrality) (shown in Figure 2.6), it appears that the dependent variables are skewed to the left and not normally distributed. Accordingly, we specify the dependent variables in logarithmic forms in the regression analysis.

We measure the success of OSS projects from both the supply side and demand side. It is widely acknowledged that the supply and demand of goods are endogenous and would influence each other simultaneously over time. To account for the simultaneity issue, we estimate the parameters of our research model by using the simultaneous equation modeling and estimation approach. We consider Three-Stage Least Squares (3SLS) estimation method (Wooldridge 2003, Greene 2003). 3SLS is the two-stage least squares version of the seemingly unrelated regression (SUR) method. It is an appropriate technique when right-hand side variables are correlated with the error terms, and there is both heteroskedasticity, and contemporaneous correlation in the residuals. Such an estimation technique accounts for the simultaneity of the demand (popularity) and supply (development activity) factors and thus minimizes the bias in the estimated parameters of the proposed model.



**Figure 2.6 Histograms of Selected Variables**

In our model, we considered some project-specific characteristics as control variables. However, what we accounted for in the model are the most salient project-specific attributes. Some other project-specific factors, such as data base environment, development status, operating systems, project topic, and user interface may also have impacts on the success of OSS projects. Those project-specific factors that do not change over time and which are unobserved to the researcher, are generally referred to as unobserved attributes or heterogeneity. In order to account for the impact of such unobserved attributes of the projects on the model's dependent variables, we utilize the fixed effects model estimation approach, that is, we include project dummies to estimate the model using the 3SLS method. In addition, the data was collected over time. There may be unobserved time-specific effects across different time periods. Thus, we not only include project dummies in our estimation model but also time period dummies.

The social network measures such as project centrality, project density and leadership centrality have potential endogeneity issues. To solve the potential endogeneity bias of the estimated model parameters, we apply lagged network measures (network variables measured in some earlier periods) as the instruments for the network measures. For example, we adopt project centrality measured in some earlier periods as the instrumental variables for current period's project centrality measure:

Project\_Centrality<sub>i,t+1</sub> , Project\_Centrality<sub>i,t+2</sub> , Project\_Centrality<sub>i,t+3</sub> , Project\_Centrality<sub>i,t+4</sub> and



$Project\_Centrality_{i,t-5}$  ; project density measured in some earlier periods as the instruments for current period's project density:  $Project\_Density_{i,t-1}$  ,  $Project\_Density_{i,t-2}$  ,  $Project\_Density_{i,t-3}$  ,  $Project\_Density_{i,t-4}$  and  $Project\_Density_{i,t-5}$  ; leadership centrality measured in some earlier periods as the instruments for current period's leadership centrality:  $Leadership\_Centrality_{i,t-1}$  ,  $Leadership\_Centrality_{i,t-2}$  ,  $Leadership\_Centrality_{i,t-3}$  ,  $Leadership\_Centrality_{i,t-4}$  and  $Leadership\_Centrality_{i,t-5}$  ; the instruments for interaction effect of leadership centrality and participation are:  $Leadership\_Centrality_{i,t-1} \times Participation_i$  ,  $Leadership\_Centrality_{i,t-2} \times Participation_i$  ,  $Leadership\_Centrality_{i,t-3} \times Participation_i$  ,  $Leadership\_Centrality_{i,t-4} \times Participation_i$  ,  $Leadership\_Centrality_{i,t-5} \times Participation_i$  .

Table 2.2 shows the main estimation results, in which we take into account project and time period fixed effects, simultaneity issues of popularity and development activity, and endogeneity issues of network communication measures. Before discussing the results of our hypotheses test using model estimation results in Table 2.2, we provide an elaboration of various robustness checks on the results in Table 2.2.

**Table 2.2 Main Estimation Results**

Observations: 1004				
	Coef.	S.E.	z	p> z
<b>Log Activity</b>				
Log Popularity	1.506***	0.268	5.630	0.000
Project Centrality	0.026	0.387	0.070	0.947
Project Density	-2.176***	0.475	-4.580	0.000
Leadership Centrality	1.402***	0.389	3.600	0.000
Participation	0.469***	0.092	5.080	0.000
Leadership Centrality * Participation	0.199***	0.052	3.860	0.000
Project Complexity	-0.166***	0.034	-4.900	0.000
Licence (Restrictive)	10.792***	1.871	5.770	0.000
$R^2 = 0.312$				
<b>Log Popularity</b>				
Log Activity	0.631***	0.073	8.690	0.000
Project Centrality	-0.366*	0.187	-1.950	0.051
Project Density	1.824***	0.217	8.420	0.000
Leadership Centrality	-0.845***	0.195	-4.340	0.000
Participation	0.009	0.013	0.710	0.479
Leadership Centrality * Participation	-0.101***	0.025	-3.970	0.000
Project Age	0.015***	0.001	15.620	0.000
Programming Language (C/C++)	0.262***	0.064	4.100	0.000
Target Audience (end user)	0.386***	0.095	4.060	0.000
$R^2 = 0.968$				
<b>Project Centrality</b>				
Project Centrality (-1)	0.288***	0.023	12.550	0.000
Project Centrality (-2)	0.187***	0.024	7.920	0.000
Project Centrality (-3)	0.093***	0.027	3.450	0.001
Project Centrality (-4)	0.095***	0.028	3.420	0.001
Project Centrality (-5)	0.042	0.027	1.550	0.122
$R^2 = 0.145$				
<b>Project Density</b>				
Project Density (-1)	0.221***	0.024	9.110	0.000
Project Density (-2)	0.170***	0.024	7.040	0.000
Project Density (-3)	0.116***	0.025	4.650	0.000
Project Density (-4)	0.113***	0.025	4.550	0.000
Project Density (-5)	0.081***	0.025	3.180	0.001
$R^2 = 0.108$				
<b>Leadership Centrality</b>				
Leadership Centrality (-1)	0.299***	0.020	15.180	0.000
Leadership Centrality (-2)	0.226***	0.020	11.080	0.000
Leadership Centrality (-3)	0.143***	0.022	6.380	0.000
Leadership Centrality (-4)	0.117***	0.023	5.160	0.000
Leadership Centrality (-5)	0.105***	0.022	4.820	0.000
$R^2 = 0.345$				
<b>Leadership*Participation</b>				
Leadership Centrality (-1) * Participation	0.248***	0.023	10.990	0.000
Leadership Centrality (-2) * Participation	0.227***	0.024	9.580	0.000
Leadership Centrality (-3) * Participation	0.190***	0.024	7.990	0.000
Leadership Centrality (-4) * Participation	0.243***	0.023	10.410	0.000
Leadership Centrality (-5) * Participation	0.051**	0.024	2.150	0.031
$R^2 = 0.590$				

Note: 1. Full table including the coefficients and statistics of month dummies and project dummies are shown in Appendix. 2. Significance: 10% \*; 5% \*\*; 0.01% \*\*\*

## 2.5.2 Robustness Checks

To strengthen the rigor of our work, we perform multiple robustness checks for simultaneity bias of popularity and development activity, endogeneity bias of network measures, and different measurements of popularity and development activity.

*Robustness Check for Simultaneity Bias:* Table 2.3 shows the statistics of robustness check for simultaneity bias of popularity and development activity. We compare the results from three estimations: (I) SB1: we first estimate a basic model using OLS without considering simultaneity of popularity and development activity and by

	SB1		SB2		SB3	
	1627		1627		1627	
Observations	Coef.	S.E.	Coef.	S.E.	Coef.	S.E.
<b>Log Activity</b>						
Log Popularity	0.028***	0.007	0.015	0.018	0.100***	0.015
Project Centrality	0.644***	0.055	0.659***	0.058	0.306***	0.044
Project Density	-0.971***	0.074	-0.993***	0.080	-0.462***	0.061
Leadership Centrality	0.215***	0.065	0.206***	0.066	0.567***	0.061
Participation	-0.002	0.008	-0.002	0.007	-0.036***	0.009
Leadership * Participation	-0.024*	0.013	-0.023*	0.013	-0.036***	0.012
Project Complexity	0.032***	0.003	0.032***	0.003	0.014***	0.005
Licence	-0.035	0.027	-0.031	0.027	0.118	0.089
	$R^2 = 0.250$		$R^2 = 0.249$		$R^2 = 0.699$	
<b>Log Popularity</b>						
Log Activity	0.294***	0.078	-0.127	0.313	1.444***	0.088
Project Centrality	0.886***	0.180	1.169***	0.273	-0.402***	0.078
Project Density	-1.265***	0.247	-1.729***	0.418	0.530***	0.099
Leadership Centrality	-0.495**	0.207	-0.403*	0.219	-0.694***	0.092
Participation	-0.036	0.024	-0.035	0.024	0.056***	0.012
Leadership*participation	0.026	0.043	0.019	0.043	0.045***	0.018
Project Age	0.022***	0.001	0.022***	0.002	0.009***	0.008
Programming Language	0.082	0.077	0.053	0.095	-0.087	0.111
Target Audience	0.474***	0.077	0.498***	0.090	0.455***	0.102
	$R^2 = 0.199$		$R^2 = 0.184$		$R^2 = 0.928$	

Note: 1. The coefficients and statistics of month dummies and project dummies are omitted because of space limitation; 2. Significance: 10% \*; 5% \*\*; 0.01% \*\*\*

excluding project and month fixed effects; (II) SB2: we adopt 3SLS to account for potential simultaneity bias without including project and month fixed effects; (III) SB3: we estimate a model using 3SLS accounting for simultaneity bias and by including project and month fixed effects.

Compared results across SB1, SB2 and SB3, the model coefficients' magnitudes vary considerably if simultaneity of popularity and development activity are accounted for and if project and time fixed effects are included in both the activity and popularity models. Some signs of other variables (project centrality, project density, leadership centrality, and leadership centrality  $\times$  participation) are changed in the popularity model. For example, in the activity model, the coefficient magnitude of popularity ( $\alpha_1$ ) in SB3 is five times larger than that of in SB1 or SB2 ( $\alpha_1=0.100$  in SB3,  $\alpha_1=0.028$  in SB1 and  $\alpha_1=0.015$  in SB2); the coefficient magnitudes of project centrality ( $\alpha_2$ ) and project density ( $\alpha_3$ ) in SB3 are around half less than those of in SB1 or SB2 ( $\alpha_2=0.306$  in SB3,  $\alpha_2=0.644$  in SB1 and  $\alpha_2=0.659$  in SB2;  $\alpha_3=-0.462$  in SB3,  $\alpha_3=-0.971$  in SB1 and  $\alpha_3=-0.993$  in SB2). In another example, for the popularity model, the signs of project centrality ( $\beta_2$ ) and project density ( $\beta_3$ ) in SB3 are different from those in SB1 or SB2 ( $\beta_2=-0.402$  in SB3,  $\beta_2=0.886$  in SB1 and  $\beta_2=1.169$  in SB2;  $\beta_3=0.530$  in SB3,  $\beta_3=-1.265$  in SB1 and  $\beta_3=-1.729$  in SB2). In addition, R-Squares in SB3 are much larger than those in SB1 or SB2 (in the activity model,  $R^2=0.699$  in SB3,  $R^2=0.250$  in SB1, and  $R^2=0.249$  in SB2; in the popularity model,  $R^2=0.928$  in SB3,  $R^2=0.199$  in SB1, and  $R^2=0.184$  in SB2). These large differences are due to

the unobserved effects of projects and time periods. There are some time-consistent factors, such as data base environment, operating systems, project topic, user interface and so on, which we do not consider in the regression. They may have the impact on success of OSS projects. Without considering these factors, the model coefficients would be inaccurately estimated. For example, if we do not consider project and time fixed effects, the results from 3SLS (SB2) show that there is no significant effects between popularity and development activity ( $\alpha_1=0.015$  and  $p=0.407$ ;  $\beta_1=-0.127$  and  $p=0.684$ ). However, from OLS (SB1), results show that popularity and development activity are highly related (both  $\alpha_1$  and  $\beta_1$  are significant at 1% confidence interval in SB1). Therefore, the estimated model parameters from both SB1 and SB2 are biased. Although we consider both simultaneity and fixed effects in SB3, the estimation may still be biased because there are possible endogeneity issues in the models.

*Robustness Check for Endogeneity Bias:* Table 2.4 shows the results of robustness check for endogeneity bias of network measures. All the estimations in Table 2.4 have considered simultaneity issue and project and month fixed effects. (I) EB1: we list the estimation results based on the basic 3SLS model (which is the same as SB3); (II) EB2: we use lagged network measures (network measures in the immediate past one period) as the independent variables in the regression. Thus, we use  $Project\_Centrality_{i,t-1}$ ,  $Project\_Density_{i,t-1}$ ,  $Leadership\_Centrality_{i,t-1}$ ,  $Leadership\_Centrality_{i,t-1} \times Participation_i$  together with other project control variables as the independent variables; (III) EB3: we apply lagged network measures as the instrumental variables for the network measures. The

**Table 2.4 Robustness Check: Endogeneity Bias**

	EB1		EB2		EB3	
Observations	1627		1503		879	
	Coef.	S.E.	Coef.	S.E.	Coef.	S.E.
<b>Log Activity</b>						
Log Popularity	0.100***	0.015	0.104***	0.016	1.839***	0.282
Project Centrality	0.306***	0.044	0.148***	0.049	-0.129	0.325
Project Density	-0.462***	0.061	-0.193***	0.068	-0.673**	0.328
Leadership Centrality	0.567***	0.061	0.267***	0.067	0.642*	0.362
Participation	-0.036***	0.009	-0.053***	0.010	-0.182***	0.016
Leadership * Participation	-0.036***	0.012	-0.025*	0.013	0.105***	0.036
Project Complexity	0.014***	0.005	0.017***	0.005	-0.199***	0.033
Licence	0.118	0.089	0.064	0.098	4.149***	0.833
	$R^2 = 0.699$		$R^2 = 0.654$		$R^2 = 0.577$	
<b>Log Popularity</b>						
Log Activity	1.444***	0.088	1.498***	0.103	0.458***	0.066
Project Centrality	-0.402***	0.078	-0.195**	0.080	-0.112	0.144
Project Density	0.530***	0.099	0.188*	0.105	0.682***	0.140
Leadership Centrality	-0.694***	0.092	-0.269***	0.098	-0.468***	0.159
Participation	0.056***	0.012	0.075***	0.013	-0.022*	0.013
Leadership*Participation	0.045***	0.018	0.027	0.019	-0.032**	0.016
Project Age	0.009***	0.008	0.010***	0.002	0.073***	0.003
Programming Language	-0.087	0.111	-0.036	0.125	-4.309***	0.156
Target Audience	0.455***	0.102	0.454***	0.112	1.797***	0.075
	$R^2 = 0.928$		$R^2 = 0.917$		$R^2 = 0.988$	
<b>Project Centrality</b>						
Project Centrality (-1)	---		---		0.280***	0.025
Project Centrality (-2)	---		---		0.184***	0.025
Project Centrality (-3)	---		---		0.073**	0.030
Project Centrality (-4)	---		---		0.146***	0.032
Project Centrality (-5)	---		---		0.033	0.032
Project Centrality (-6)	---		---		-0.018	0.030
<b>Project Density</b>						
Project Density (-1)	---		---		0.217***	0.027
Project Density (-2)	---		---		0.184***	0.027
Project Density (-3)	---		---		0.103***	0.028
Project Density (-4)	---		---		0.141***	0.028
Project Density (-5)	---		---		0.039	0.028
Project Density (-6)	---		---		0.023	0.028
<b>Leadership Centrality</b>						
Leadership Centrality (-1)	---		---		0.299***	0.021
Leadership Centrality (-2)	---		---		0.199***	0.022
Leadership Centrality (-3)	---		---		0.114***	0.024
Leadership Centrality (-4)	---		---		0.159***	0.025
Leadership Centrality (-5)	---		---		0.095***	0.025
Leadership Centrality (-6)	---		---		0.042*	0.024
<b>Leadership*participation</b>						
Leader_Cent(-1)*Part	---		---		0.249***	0.025
Leader_Cent(-2)*Part	---		---		0.184***	0.026
Leader_Cent(-3)*Part	---		---		0.148***	0.027
Leader_Cent(-4)*Part	---		---		0.325***	0.027
Leader_Cent(-5)*Part	---		---		0.042	0.027
Leader_Cent(-6)*Part	---		---		0.017	0.025

Note: 1. The coefficients and statistics of month dummies and project dummies are omitted because of space limitation; 2. Significance: 10% \*; 5% \*\*; 0.01% \*\*\*

instruments for project centrality are  $Project\_Centrality_{i,t-1}$  ,  $Project\_Centrality_{i,t-2}$  ,  $Project\_Centrality_{i,t-3}$  ,  $Project\_Centrality_{i,t-4}$  ,  $Project\_Centrality_{i,t-5}$  ,  $Project\_Centrality_{i,t-6}$  ; the instruments for project density are  $Project\_Density_{i,t-1}$  ,  $Project\_Density_{i,t-2}$  ,  $Project\_Density_{i,t-3}$  ,  $Project\_Density_{i,t-4}$  ,  $Project\_Density_{i,t-5}$  ,  $Project\_Density_{i,t-6}$  ; the instruments for leadership centrality are  $Leadership\_Centrality_{i,t-1}$  ,  $Leadership\_Centrality_{i,t-2}$  ,  $Leadership\_Centrality_{i,t-3}$  ,  $Leadership\_Centrality_{i,t-4}$  ,  $Leadership\_Centrality_{i,t-5}$  ,  $Leadership\_Centrality_{i,t-6}$  ; the instruments for interaction effect of leadership centrality and participation are:  $Leadership\_Centrality_{i,t-1} \times Participation_t$  ,  $Leadership\_Centrality_{i,t-2} \times Participation_t$  ,  $Leadership\_Centrality_{i,t-3} \times Participation_t$  ,  $Leadership\_Centrality_{i,t-4} \times Participation_t$  ,  $Leadership\_Centrality_{i,t-5} \times Participation_t$  ,  $Leadership\_Centrality_{i,t-6} \times Participation_t$  .

In EB1 and EB2, development activity and popularity are treated as endogenous variables. The only difference is that lagged network measures are used as exogenous variables in EB2. We find that the signs and significance of the variables do not change too much in EB1 and EB2, except that the coefficient magnitudes of lagged network measures (in EB2) are much lower than those of original network measures (in EB1). For example, in the activity model, the coefficient of project centrality  $\alpha_2 = 0.306$  in EB1 and the coefficient of lagged project centrality ( $Project\_Centrality_{i,t-1}$ )  $\alpha_2 = 0.148$  in EB2; the coefficient of project density  $\alpha_3 = -0.462$  in EB1 and lagged project density ( $Project\_Density_{i,t-1}$ )  $\alpha_3 = -0.193$  in EB2; the coefficient of leadership centrality  $\alpha_4 = 0.567$  in EB1 and lagged leadership centrality ( $Leadership\_Centrality_{i,t-1}$ )  $\alpha_4 = 0.267$  in EB2. Similarly, in the popularity model, the coefficient of project centrality  $\beta_2 = -0.402$  in EB1 and lagged project centrality  $\beta_2 = -0.195$  in EB2; the coefficient of

project density  $\beta_3=0.530$  in EB1 and lagged project density  $\beta_3=0.188$  in EB2; the coefficient of leadership centrality  $\beta_4=-0.694$  in EB1 and lagged leadership centrality  $\beta_4=-0.269$  in EB2.

EB3 is fairly different from EB1 and EB2. Besides development activity and popularity, network measures such as project centrality, project density, leadership centrality and interaction effect of leadership centrality and participation are also set as endogenous variables in EB3. The coefficient magnitude, significance and signs in EB3 differ greatly when comparing with EB1 and EB2. For example, the effect of popularity on development activity ( $\alpha_1$ ) in EB3 is much larger than that in EB1 or EB2 ( $\alpha_1=1.839$  in EB3,  $\alpha_1=0.100$  in EB1, and  $\alpha_1=0.104$  in EB2); on the contrary, the effect of development activity on popularity ( $\beta_1$ ) in EB3 is much lower than that in EB1 or EB2 ( $\beta_1=0.458$  in EB3,  $\beta_1=1.444$  in EB1, and  $\beta_1=1.498$  in EB2). For another example, the signs of project complexity ( $\alpha_7$ ) and interaction effect of leadership centrality and participation ( $\alpha_6$  or  $\beta_6$ ) in EB3 are different from the signs of those in EB1 or EB2 ( $\alpha_7=-0.199$  in EB3,  $\alpha_7=0.014$  in EB1 and  $\alpha_7=0.017$  in EB2;  $\alpha_6=0.105$  in EB3,  $\alpha_6=-0.036$  in EB1 and  $\alpha_6=-0.025$  in EB2;  $\beta_6=-0.032$  in EB3,  $\beta_6=0.045$  in EB1 and  $\beta_6=0.027$  in EB2). In addition, the effect of project centrality on development activity ( $\alpha_2$ ) is insignificant in EB3 ( $\alpha_2=-0.129$ ,  $p=0.691$ ), but significant at 1% confidence interval in EB1 or EB2; the effect of project centrality on popularity ( $\beta_2$ ) is also insignificant in EB3 ( $\beta_2=-0.112$ ,  $p=0.437$ ), but significant in EB1 or EB2.



These large differences between EB1, EB2 and EB3 can be attributed to the endogeneity of network measures (i.e. project centrality, project density and leadership centrality). Estimated model coefficients in EB1 are biased because the endogeneity of network measures is not considered. Results in EB2 are also biased. We use lagged network measures (  $Project\_Centrality_{i,t-1}$  ,  $Project\_Density_{i,t-1}$  ,  $Leadership\_Centrality_{i,t-1}$  ) to substitute original network measures (  $Project\_Centrality_{i,t}$  ,  $Project\_Density_{i,t}$  ,  $Leadership\_Centrality_{i,t}$  ) as the independent variables. However, these lagged network measures underestimate the impact of communication patterns on the project success, because they neglect the effect of communication patterns over multiple successive past periods that plays an importance role in determining current project success. In EB3, the original network measures are endogenous. We adopt lagged network measures as the instruments for the original network measures. We stop at using lagged measures up to lag=-6. There are two reasons that we choose the lag from -1 to -6. First, the more lags we adopt, the more observations are dropped from the estimation sample. When we do not use lagged variables, we have 1627 observations in the estimation (in EB1); when we use lagged variables (lag=-1, -2, ..., -6), we have only 879 observations left (in EB3). Second, when we choose the lag from -1 to -6 as the instruments of network measures, we find that the instruments with lag=-6 (e.g.  $Project\_Centrality_{i,t-6}$  ,  $Project\_Density_{i,t-6}$  ,  $Leadership\_Centrality_{i,t-6}$  ,  $Leadership\_Centrality_{i,t-6} \times Participation_i$  ) are not statistically correlated with the original network measures (shown in Table 2.4). Thus, EB3 is still biased because of using

more lags as the instruments. Hence, we choose instruments with the lag from -1 to -5 in our main results (as shown Table 2.2).

Compared with EB3, the coefficient magnitude, significance and signs of the variables in the main results (“MR” in short) differ somewhat. For example, the coefficient magnitudes of project density ( $\alpha_3$  or  $\beta_3$ ), leadership centrality ( $\alpha_4$  or  $\beta_4$ ), and interaction effect of leadership centrality and participation ( $\alpha_6$  or  $\beta_6$ ) in MR are much larger than those in EB3 (In the activity model,  $\alpha_3=-2.176$  in MR and  $\alpha_3=-0.673$  in EB3;  $\alpha_4=1.402$  in MR and  $\alpha_4=0.642$  in EB3;  $\alpha_6=0.199$  in MR and  $\alpha_6=0.105$  in EB3. In the popularity model:  $\beta_3=1.824$  in MR and  $\beta_3=0.682$  in EB3;  $\beta_4=-0.845$  in MR and  $\beta_4=-0.468$  in EB3;  $\beta_6=-0.101$  in MR and  $\beta_6=-0.032$  in EB3). The sign of project centrality ( $\alpha_2$ ) in the activity model in MR is different from that in EB3 ( $\alpha_2=0.026$  in MR and  $\alpha_2=-0.129$  in EB3) although the coefficients are statistically insignificant. In addition, the effect of project centrality on the popularity is significant at 10% confidence interval in MR ( $\beta_2=-0.366$ ,  $p=0.051$ ), but insignificant in EB3 ( $\beta_2=-0.112$ ,  $p=0.437$ ).

*Robustness Check for Measures of Dependent Variables:* At last, we show the robustness checks for the different measures of dependent variables in Table 2.5 and Table 2.6. We adopt a composite measure of each dependent variable in the main estimation. Development activity is measured by the average number of total tracks and file releases. Popularity is measured by the formula: “number of web hits  $\times$  (1+

**Table 2.5 Robustness Check: Development Activity**

Observations	DA1		DA2	
	999		352	
	Coef.	S.E.	Coef.	S.E.
<b>Log Activity</b>				
Log Popularity	3.163***	0.599	-3.090**	1.265
Project Centrality	-0.173	0.866	0.066	0.567
Project Density	-5.767***	1.091	-1.877	1.659
Leadership Centrality	4.370***	0.910	-0.787	1.372
Participation	1.065***	0.208	0.387	0.375
Leadership Centrality * Participation	0.460***	0.118	0.241	0.239
Project Complexity	-0.341***	0.076	0.158***	0.058
Licence	23.255***	4.193	-9.478**	3.752
	$R^2 = 0.163$		$R^2 = 0.622$	
<b>Log Popularity</b>				
Log Activity	0.285***	0.033	-0.079**	0.037
Project Centrality	-0.286	0.188	0.058	0.128
Project Density	2.044***	0.224	1.469***	0.323
Leadership Centrality	-1.156***	0.208	-1.183***	0.250
Participation	0.016	0.017	0.101**	0.041
Leadership Centrality * Participation	-0.106***	0.026	-0.078	0.055
Project Age	0.025***	0.002	0.031***	0.003
Programming Language	-0.781***	0.086	0.360**	0.184
Target Audience	-0.592***	0.089	0.998***	0.221
	$R^2 = 0.960$		$R^2 = 0.980$	
<b>Project Centrality</b>				
Project Centrality (-1)	0.284***	0.023	0.377***	0.039
Project Centrality (-2)	0.190***	0.024	0.140***	0.039
Project Centrality (-3)	0.094***	0.027	0.156***	0.052
Project Centrality (-4)	0.106***	0.028	-0.009	0.052
Project Centrality (-5)	0.031	0.027	0.049	0.052
<b>Project Density</b>				
Project Density (-1)	0.220***	0.024	0.258***	0.044
Project Density (-2)	0.177***	0.024	0.180***	0.045
Project Density (-3)	0.116***	0.025	0.202***	0.049
Project Density (-4)	0.117***	0.025	0.018	0.047
Project Density (-5)	0.077***	0.025	0.049	0.047
<b>Leadership Centrality</b>				
Leadership Centrality (-1)	0.297***	0.020	0.364***	0.033
Leadership Centrality (-2)	0.228***	0.020	0.239***	0.033
Leadership Centrality (-3)	0.145***	0.022	0.204***	0.041
Leadership Centrality (-4)	0.126***	0.022	0.002	0.040
Leadership Centrality (-5)	0.092***	0.022	0.118***	0.039
<b>Leadership*Participation</b>				
Leadership Centrality (-1) * Participation	0.240***	0.022	0.285***	0.037
Leadership Centrality (-2) * Participation	0.237***	0.024	0.308***	0.035
Leadership Centrality (-3) * Participation	0.194***	0.024	0.278***	0.038
Leadership Centrality (-4) * Participation	0.253***	0.023	-0.015	0.033
Leadership Centrality (-5) * Participation	0.038	0.024	0.154***	0.031

Note: 1. The coefficients and statistics of month dummies and project dummies are omitted because of space limitation; 2. Significance: 10% \*, 5% \*\*, 0.01% \*\*\*

**Table 2.6 Robustness Check: Project Popularity**

Observations	PP1		PP2	
	1004		1004	
	Coef.	S.E.	Coef.	S.E.
<b>Log Activity</b>				
Log Popularity	0.654***	0.116	-0.005	0.009
Project Centrality	0.026	0.387	-0.583	0.374
Project Density	-2.176***	0.475	-0.491	0.430
Leadership Centrality	1.403***	0.389	0.992***	0.373
Participation	0.260***	0.056	-0.079***	0.017
Leadership Centrality * Participation	0.199***	0.052	0.172***	0.050
Project Complexity	0.304***	0.063	0.024**	0.010
Licence	0.344	0.419	-0.258	0.271
	$R^2 = 0.312$		$R^2 = 0.589$	
<b>Log Popularity</b>				
Log Activity	1.453***	0.167	0.000***	0.000
Project Centrality	-0.843*	0.431	0.000	0.000
Project Density	4.199***	0.499	0.000***	0.000
Leadership Centrality	-1.946***	0.448	0.000***	0.000
Participation	0.022	0.030	-0.616***	0.000
Leadership Centrality * Participation	-0.232***	0.058	0.000***	0.000
Project Age	0.034***	0.002	0.011***	0.000
Programming Language	0.603***	0.147	7.695***	0.000
Target Audience	0.888***	0.219	-0.964***	0.000
	$R^2 = 0.961$		$R^2 = 1.000$	
<b>Project Centrality</b>				
Project Centrality (-1)	0.288***	0.023	0.283***	0.023
Project Centrality (-2)	0.187***	0.024	0.184***	0.024
Project Centrality (-3)	0.093***	0.027	0.089***	0.027
Project Centrality (-4)	0.095***	0.028	0.101***	0.028
Project Centrality (-5)	0.042	0.027	0.045	0.027
<b>Project Density</b>				
Project Density (-1)	0.221***	0.024	0.209***	0.025
Project Density (-2)	0.170***	0.024	0.171***	0.025
Project Density (-3)	0.116***	0.025	0.135***	0.026
Project Density (-4)	0.113***	0.025	0.137***	0.026
Project Density (-5)	0.081***	0.025	0.063**	0.026
<b>Leadership Centrality</b>				
Leadership Centrality (-1)	0.299***	0.020	0.301***	0.020
Leadership Centrality (-2)	0.226***	0.020	0.218***	0.021
Leadership Centrality (-3)	0.143***	0.022	0.141***	0.023
Leadership Centrality (-4)	0.117***	0.023	0.123***	0.023
Leadership Centrality (-5)	0.105***	0.022	0.109***	0.022
<b>Leadership*Participation</b>				
Leadership Centrality (-1) * Participation	0.248***	0.023	0.250***	0.022
Leadership Centrality (-2) * Participation	0.227***	0.024	0.218***	0.024
Leadership Centrality (-3) * Participation	0.190***	0.024	0.190***	0.024
Leadership Centrality (-4) * Participation	0.243***	0.023	0.241***	0.023
Leadership Centrality (-5) * Participation	0.051**	0.024	0.060**	0.024

Note: 1. The coefficients and statistics of month dummies and project dummies are omitted because of space limitation; 2. Significance: 10% \*; 5% \*\*; 0.01% \*\*\*

subscription)”. In order to show that these composite measures are more appropriate than the single measurements, we compare the estimation results when dependent variables are measured by a single element under the same settings of our main results (shown in Table 2.2). In Table 2.5, DA1 shows the results when development activity is measured by the number of total tracks; DA2 shows the results when development activity is measured by the number of file releases. In Table 2.6, PP1 shows the results when project popularity is measured by the number of web hits; PP2 shows the results when project popularity is measured by the level of subscriptions.<sup>8</sup>

In DA1, the coefficient magnitudes of some variables (project density  $\alpha_3$ , leadership centrality  $\alpha_4$  and interaction effect of leadership centrality and participation  $\alpha_6$ ) in the activity model are larger than those in MR ( $\alpha_3=-2.176$  in MR and  $\alpha_3=-5.767$  in DA1;  $\alpha_4=1.402$  in MR and  $\alpha_4=4.370$  in DA1;  $\alpha_6=0.199$  in MR and  $\alpha_6=0.460$  in DA1). However, the effect of project centrality on the popularity is insignificant in DA1 ( $\beta_2=-0.286$ ,  $p=0.130$ ), but significant at 10% confidence interval in MR ( $\beta_2=-0.366$ ,  $p=0.051$ ). In addition, the signs of some variables in DA1 are unreasonable. For example, the effect of programming language ( $\beta_8$ ) or target audience ( $\beta_9$ ) on popularity is significantly negative ( $\beta_8=-0.781$ ,  $\beta_9=-0.592$ ). From the end users’ perspective, software written in more popular programming languages may be more popular among end users, since more people can modify (or customize) the open source

---

<sup>8</sup> In Table 5, project popularity is still captured by the multiple measures of web hits and subscription; in Table 6, development activity is still measured by the average number of total tracks and file releases

software conveniently. Apparently, software targeted at the general end users may appeal to more users, and thus leading to increased project popularity. Therefore, compared with DA1 and MR, estimation in MR is more appropriate.

In DA2, the effects of network measures such as project centrality ( $\alpha_2$ ), project density ( $\alpha_3$ ), leadership centrality ( $\alpha_4$ ) and interaction effect of leadership centrality and participation ( $\alpha_6$ ) on the development activity are statistically insignificant ( $\alpha_2 = 0.066, p=0.970$ ;  $\alpha_3 = -1.877, p=0.258$ ;  $\alpha_4 = -0.787, p=0.566$ ;  $\alpha_6 = 0.241, p=0.313$ ). Moreover, the relationship between development activity and popularity is improper. The effect of popularity on development activity ( $\alpha_1 = -3.090$ ) is significantly negative at 5% confidence interval; the effect of development activity on popularity ( $\beta_1 = -0.079$ ) is also significantly negative at 5% confidence interval. Therefore, compared with DA2 and MR, estimated model coefficients in MR are more logical from a face validity point of view.

In PP1, the effect of type of licence on the activity is statistically insignificant ( $\alpha_8 = 0.344, p=0.411$ ); the sign of project complexity is significantly positive at 1% confidence interval ( $\alpha_7 = 0.304$ ). However, we expect that a more complex project is likely to achieve a less level of development activities because if the project is too complex, the users or developers need to spend much time and resources on the project per output delivered. In PP2, coefficients of many network variables are statistically insignificant and many signs of estimated parameters are unreasonable.

Apparently, the effects of network measures on the level of subscriptions are not significant. Therefore, estimated model parameters in MR are still more appropriate and valid than those given in PP1 or PP2.

In summary, through the above analysis of robustness checks, we find that it is necessary to estimate our empirical models by accounting for project and time period fixed effects, simultaneity issues of popularity and development activity, endogeneity issues of network measures, and adopting composite measures of development activity and popularity, which are shown in Table 2.2. We discuss our hypotheses test results with regard to model estimation results (listed in Table 2.2) in the next section.

### **2.5.3 Hypothesis Test**

*Project Centrality and Success:* Based on the estimation results, we find significant effects of communication patterns on our project success measures. H1 proposes that project centrality will positively affect the level of project development activities. H2 proposes that project centrality will negatively affect the level of project popularity. Our results provide some support for these hypotheses. In the activity model, we find a positive but statistically insignificant coefficient of project centrality ( $\alpha_2=0.026$ ,  $p=0.947$ ), but in popularity model, we find a negative and statistically significant coefficient of project centrality ( $\beta_2=-0.366$ ,  $p=0.051$ ), although significant at only the 10% confidence interval. Thus, H1 is not supported, but H2 is marginally supported.

After we apply lagged network instruments, the observations are largely reduced, from 1627 observations without considering lagged network instruments to 1004 observations after applying lagged instruments. More than one third of observations are missing, which may result in the insignificant effect of project centrality on development activity. For example, in EB1, there are no missing observations (1627 observations in EB1). The effects of project centrality on both development activity and popularity are significant at 1% confidence interval. In EB2, there are 124 missing observations because of one-lag variables (1503 observations in EB2). The effect of project centrality on development activity is significant at 1% confidence interval; and the effect of project centrality on popularity is significant at 5% confidence interval.

The hypotheses test results from H1 and H2 suggest that the centrality of OSS project teams play an important role in the evolving success of the projects. Centralized projects are good for increasing development activity, but the cost is that these projects are susceptible to losing popularity among end users. The managers of the OSS projects need to target either on development activities or for popularity among end users. Balancing of communication centrality within OSS projects is of crucial importance for the success of these projects.

*Project Density and Success:* H3 proposes that project density will negatively affect the level of project development activities. The model shows the negative and



significant effect of project density on development activity at 1% confidence interval ( $\alpha_3 = -2.176$ , S.E.=0.475). Thus, H3 is supported. H4 suggests that project density will positively affect the level of project popularity. The regression results shows the positive and significant effect of density on popularity at 1% confidence interval ( $\beta_3 = 1.824$ , S.E.=0.217). Thus, H4 is supported.

The hypotheses test results from H3 and H4 enable us to draw a tentative conclusion that the effect of project density is significant to the project success which is measured by both the level of development activities and project popularity. A high density project is likely to be responsive to changes in developmental requirements, but the communication efficiency among developers is likely to be reduced. Therefore, to project managers, there also exist tradeoffs in communication density. Thus, balancing project centrality and density is an important task for project managers to enable OSS projects to achieve long-term success in the competitive environment.

*Leadership Centrality and Success:* H5 proposes that project leadership centrality will positively affect the level of project development activities. H6 proposes that leader's participations in other projects will yield a positive effect on project development activities for the projects with higher leadership centrality. The estimation results show some support for both H5 and H6. In the activity model, the effect of leadership centrality is positive and significant at 1% confidence interval ( $\alpha_4 = 1.402$ , S.E.=0.389); the effect of interaction effect of leadership centrality and participation is also

positive and significant at 1% confidence interval ( $\alpha_6=0.199$ , S.E.=0.052). Thus, H5 and H6 are supported. H7 proposes that project leadership centrality will negatively affect the level of project popularity. H8 proposes that leader's participations in other projects will yield a negative effect on project popularity for the projects with higher leadership centrality. In the popularity model, we find a negative and significant coefficient of leadership centrality ( $\beta_4=-0.845$ , S.E.=0.195); and also a negative and significant coefficient of the interaction term ( $\beta_6=-0.101$ , S.E.=0.025). Thus, H7 and H8 are supported.

The hypotheses test results from H5 to H8 suggest that leadership centrality is important for the OSS projects. Leaders in the project play a crucial role in improving the performance of the project teams. Leaders may take part in several projects simultaneously, which can help to improve the performance of the leader and enhance the reputation of the leader among the developers in different projects. However, heavily reliance on the single leader has the potential risks for the demand among end users.

*Project Specific Characteristics and Success:* Hypothesis 9 (A, B) proposes the relationships between project-specific characteristics and project development activities. H9A proposes that a project with a restrictive license will be likely to achieve a higher level of development activities. The results from the model estimation show the positive and significant effect of type of licence on the

development activity at 1% confidence interval ( $\alpha_8=10.792$ , S.E.=1.871). H9B proposes that a more complex project will be likely to achieve a lower level of development activities. We find a negative and significant coefficient of project complexity at 1% confidence interval ( $\alpha_7=-0.166$ , S.E.=0.034). Therefore, H9A and H9B are supported.

Hypothesis 10 (A, B, C) proposes the relationships between project-specific characteristics and project popularity. We find support for these hypotheses. H10A proposes that a project utilizing a popular programming language will be likely to achieve a higher level of development activities. The model shows a positive and significant coefficient of programming language at 1% confidence interval ( $\beta_8=0.262$ , S.E.=0.064). H10B proposes that a project with a longer time history will be likely to achieve a higher level of popularity. The model also shows a positive and significant coefficient of project age at 1% confidence interval ( $\beta_7=0.015$ , S.E.=0.001). H10C proposes that a project targeting at end users will be likely to achieve a higher level of popularity. We find the positive and significant effect of target audience on popularity at 1% confidence interval ( $\beta_9=0.386$ , S.E.=0.095). Therefore, H10A, H10B and H10C are supported. Table 2.7 below presents a summary of the results of our hypotheses tests.

**Table 2.7 Summary of Hypotheses Test**

H1: Project centrality will positively affect the level of project development activities.	Not Supported
H2: Project centrality will negatively affect the level of project popularity.	Supported
H3: Project density will negatively affect the level of project development activities.	Supported
H4: Project density will positively affect the level of project popularity.	Supported
H5: Project leadership centrality will positively affect the level of project development activities.	Supported
H6: Leader's heightened participations in other projects will yield a positive effect on the level of project development activities for projects with high leadership centrality.	Supported
H7: Project leadership centrality will negatively affect the level of project popularity.	Supported
H8: Leader's heightened participations in other projects will yield a negative effect on the level of project popularity for projects with high leadership centrality.	Supported
H9A: A project with a restrictive license will be likely to achieve a higher level of development activities.	Supported
H9B: A more complex project will be likely to achieve a lower level of development activities.	Supported
H10A: A project utilizing a popular programming language will be likely to achieve a higher level of development activities.	Supported
H10B: A project with a longer time history will be likely to achieve a higher level of popularity.	Supported
H10C: A project targeting at end users will be likely to achieve a higher level of popularity.	Supported

## 2.6 Concluding Remarks

The main purpose of this study is to investigate the long term effects of communication pattern on the success of open source projects. We base our research on the theoretical study of social network theory. Results, shown in Table 2.2, are generally supportive of the hypotheses posited in this paper. By observing changes in communication patterns for an extended period, we find significant impacts of communication patterns on the

outcome of the project.

The findings of our research has implications for project managers and developers in open source environments, as well as for managers of commercial software firms, such as Microsoft and IBM, which are actively participating in open source projects. Furthermore, these findings and implications can also be generalized for virtual team communities. Open source software development is one of the prime manifestations of virtual teams collaborating over the Internet. Such virtual team creates value through developing and spreading new knowledge and capabilities, fostering innovations, and building and testing trust in working relations (Kidane and Gloor 2007), relying heavily on information and communication technologies to accomplish their tasks (Powell et al 2004). For virtual teams to achieve their objectives and successfully complete their tasks, information must be effectively exchanged. Thus, communication and coordination have been found to be two major aspects that significantly affect the performance of virtual teams (Johansson et al. 1999; Maznevski and Chudoba 2001). Thus, the findings and implications from examining the coordination and communication characteristics (i.e., the social network attributes: project centrality, project density, and leadership centrality) are also helpful for virtual team leaders to monitor the performance of the teams.

As one of the attributes of communication pattern, project centrality expresses the inequality or variance of contribution in communication among the contributors. The

effects of project centrality on project development activity and popularity are examined and uncovered by our research model. The significant effect indicates that substantial inequality of developers' contributions to the projects is important for project success. In a centralized network, some core members contribute more to the project. Such core-periphery structures can potentially enhance the speed and flexibility with which information diffuses within a group (Cummings and Cross 2003). Thus, centralized projects will be with higher communication efficiency and thereby related to better performance. However, heavily reliance on a small group of core single developers threatens the success of the open source projects. This may largely deter end users who seek stable continuous support and maintenance from the project teams. Centralized projects are good for developers to increase development activity, but the cost is that these projects are susceptible to losing end users and popularity. The project managers need to clarify their objectives: for development activity or for popularity. Balancing the communication centrality is important for the success of the projects.

Another attribute of communication pattern discussed in this paper is project density. Project density measures the readiness of the group to respond to changes, and how close a network is to realize its potential. In high-density projects, information dissemination is impeded, which negatively affects communication efficiency. However, this kind of structure allows for a speedy response to changes in the environment. Therefore, to project managers, there also exist tradeoffs of communication density. Thus, balancing project centrality and density is an important

task for project managers to ensure the success and survivability of OSS projects in a competitive environment.

Leadership centrality measures the stature and influence of the project managers on the projects. A project manager plays the key role of coordinating overall project development activity. In a project with higher manager centrality, the manager has higher stature and more influence in the project team. The project manager serves as a conduit for information communication among the project team. In such high manager centrality projects, the managers can attract and communicate with many developers, and those developers may actively exchange information with their managers because of manager's high stature and influence. However, heavy reliance on a single developer threatens the popularity the open source projects. There are the potential risks inside the project with high leadership centrality. There is the negative impact of leader's simultaneous participations in other projects on popularity in high leadership centrality projects.

In a nutshell, the project managers need to realize the importance of the communication pattern of project team. According to the objectives of projects, a proper and planned control for the communication among team members is crucial for the survivability of the open source projects.

In general, executives at open source communities should note that communication pattern of the project team is critical to the long-term sustainability of OSS movement.

Since the view of project success from developers and general users are different, the success of projects could not be captured by the single aspect. Based on our results, the impact of communication patterns on demand side and supply side success are different. It implies that the project managers can reap the benefits if they structure their project teams with care. The managers need to consider and balance not only communication centrality but also communication density among developers.

Since research on open systems environments is new, theoretical insights in this domain are just emerging (Hippel and Krogh 2003). From a theoretical standpoint, we apply social network theory into the information systems domain, in particular, into the study of success of OSS projects. Although previous works have been conducted on various IS phenomenon from a social network perspective, we are among the first to apply the social network theory with encouraging success to the OSS development realm. Our results suggest several directions for theory development on the effect of communication pattern of the project team on project success. First, it is important to recognize that the effect of communication pattern varies with the indicators of project success. Researchers in this domain could explore this difference more in depth. OSS development is different from the traditional software development. For the traditional software project, the eventual market share of the software product can indicate whether this project is a success or failure. However, for the OSS project, the success of the project needs to be viewed from both the developers and the end users' aspects. Thus, OSS success has different dimensions. A single measurement or



operationalization will not be sufficient to completely represent success. Therefore, development activity and popularity are both more useful indicators of the OSS success. The effects of communication pattern on development activity and popularity are different in that the communication structure of OSS development is viewed differently by developers and end users. Second, it is important to recognize that success is transient. The evolutionary changes in the success and survivability of software projects are related to dynamic communication patterns among stakeholders in the course of software development processes. Examining the status of projects over an extended period is a more rigorous method to assess the long term evolving success of OSS projects. Our research takes some important steps in this direction and we hope that further investigations of long term success from social network perspective are explored in the future research.

From a data manipulation standpoint, this work still has limitations which need to be further investigated. To strengthen the rigor of our work, we perform multiple robustness checks for simultaneity bias of popularity and development activity, endogeneity bias of network measures, and different measurements of popularity and development activity. However, the network measures are not something given from outside of the project. It is possible to still have the endogeneity problem. Therefore, future research will be conducted to further investigate the endogeneity issue of the measurements.

## References

Colazo, J. A., Fang, Y., Neufeld, D. “Development success in open source software projects: exploring the impact of copylefted licenses,” in *Proceedings of the 11th Americas Conference on Information Systems*, Omaha, NE, USA, 2005.

Comino, S., Manenti, F. M. and Parisi, M. L. “From planning to mature: on the determinants of open source take off,” Working Papers 0517, Department of Economics, University of Trento, Italia, 2005.

Crowston, K. “A coordination theory approach to organizational process design,” *Organization Science* (8:2), 1997, pp. 157–175.

Crowston, K., Annabi, H. and Howison, J. “Defining open source software project success,” in *Proceedings of the 11th International Conference on Information Systems*, Atlanta, GA, USA, 2003.

Crowston, K., Annabi, H., Howison, J., and Masano, C. “Towards a portfolio of FLOSS project success measures,” in *Proceedings of the Open Source Workshop of the International Conference on Software Engineering*, Edinburgh, Scotland, May 2004.

Crowston, K. and Howison, J. “The social structure of free and open source software development,” *First Monday* (10:2), 2005.

Cummings, J. and Cross, R. "Structural properties of work groups and their consequences for performance," *Social Networks* (25:3), 2003, pp. 197-210.

Feller, J. and Fitzgerald, B. "A framework for understanding the open source software development paradigm," in *Proceedings of the 8th International Conference on Information Systems*, Atlanta, GA, USA, 2000.

Fershtman, C. and Gandal, N. "The determinants of output per contributor in open source projects: an empirical examination," CEPR Discussion Paper, No. 4329, March 2004.

Freeman, L. C. "Centrality in social networks: conceptual clarification," *Social Networks* (1:3), 1979, pp. 215-239.

Freeman, L. C., Roeder D., and Mulholland, R. R. "Centrality in social networks II: experimental results," *Social Networks* (2:2), 1980, pp. 119-141.

Granovetter, M. "The strength of weak ties," *American Journal of Sociology* (78:6), 1973, pp. 1360-1380.

Greene, H. W. *Econometric Analysis*, Prentice Hall, Upper Saddle River, New Jersey, United State, 2003.

Grewal, R., Lilien, G. L., and Mallapragada, G. "Location, location, location: how network embeddedness affects project success in open source systems," *Management Science* (52:7), 2006, pp. 1043-1056.

Hanneman, R. A., and M. Riddle. *Introduction to social network methods*, University of California, Riverside, United State, 2005 (published in digital form at <http://faculty.ucr.edu/~hanneman/> ).

Hertel, G., Niedner, S., and Herrmann, S. "Motivation of software developers in open source projects: and internet-based survey of contributors to the Linux kernel," *Research Policy* (32:7), 2003, pp. 1159-1177.

Hinds D. and Lee M. R. "Social network structure as a critical success condition for virtual communities," in *Proceedings of the 41st Hawaii International Conference on System Sciences*, Hawaii, US, 2008.

Hippel, E. V., and Krogh, G. V. "Open source software and the "private-collective" innovation model: special issues for organization science," *Organization Science* (14:2), 2003, pp. 209-225.

Holmstrom, B. "Managerial Incentive Problems: A Dynamic Perspective," *Review of Economic Studies* (66), 1999, pp. 169-182.

Johansson, C, Dittrich, Y. and Juustila, A. “Software engineering across boundaries: student project in distributed collaboration,” *IEEE Transactions on Professional Communication* (42:4), 1999, pp. 286-296.

Kidane, Y. H., and Gloor, P. A. “Correlating temporal communication patterns of the Eclipse open source community with performance and creativity,” *Computational & Mathematical Organization Theory* (13:1), 2007, pp. 17-27.

Krishnamurthy, S. “Cave or community? An empirical examination of 100 mature open source projects,” *First Monday* (7:2), 2002.

Krogh, G. V., Spaeth, S., and Lakhani, K. R. “Community, joining, and specialization in open source software innovation: a case study,” *Research Policy* (32:7), 2003, pp. 1217–1241.

Lakhani, K. R. and von Hippel, E. “How open source software works: “free” user-to-user assistance,” *Research Policy* (32:6), 2003, pp. 923–943.

Lakhani, K. R. and Wolf, R. G. “Why hackers do what they do: Understanding motivation effort in free/open source software projects,” Working Paper 4425-03, MIT Sloan School of Management, Cambridge, MA, 2003.

Leavitt, H. J. “Some effects of communication patterns on group performance,” *Journal of Abnormal and Social Psychology*, 1951, (46), pp. 38-50.

Lerner, J. and Tirole, J. "Some simple economics of open source," *Journal of Industry Economics* (50:2), 2002, pp. 197-234.

Lerner, J. and Tirole, J. "The scope of open source licensing," *Journal of Law, Economics and Organization* (21:1), 2005, pp. 20-56.

Madey, G., Freeh, V., and Tynan, R. 2002. "The open source software development phenomenon: an analysis based on social network theory," in *Proceedings of the 8th Americas Conference on Information Systems*, Dallas, TX, USA, 2002.

Markus, M. L., Manville, B., and Agres, C. E. "What makes a virtual organization work?" *Sloan Management Review* (42:1), 2000, pp. 13–26.

Maznevski, M. and Chudoba, K. "Bridging space over time: global virtual team dynamics and effectiveness," *Organization Science* (11:5), 2001, pp. 473-492.

Mehra A., Dixon L. A., Brass J. D. and Robertson B. "The social network ties of group leaders: implications for group performance and leader reputation," *Organization Science* (17:1), 2006, pp. 64-79.

Mullen B., Johnson, C. and Salas, E. "Effects of communication network structure: Components of positional centrality," *Social Networks* (13:2), 1991, pp. 169-186.

O'Mahony, S. "Guarding the commons: how community managed software projects protect their work," *Research Policy* (32:7), 2003, pp. 1179–1198.

Powell, A., Piccoli, G. and Ives, B. "Virtual teams: a review of current literature and direction for future research," *The Data base for Advances in Information Systems* (35:1), 2004, pp. 6-36.

Raymond, E. S. "The cathedral and the bazaar," *First Monday* (3:3), 1998.

Sawyer, S. "Software development teams," *Communications of ACM* (47:12), 2004, pp. 95-99.

Scacchi, W. "Understanding the requirements for developing open source software systems," *IEE Proceedings Software* (149:1), 2002, pp. 24–39.

Scott John. (eds.). *Social Networks: Critical Concepts in Sociology*, London, New York, Routledge, 2002.

Sen. "Open source software development projects: determinants of project popularity," Working paper 0510003, Econometrics from EconWPA, 2005.

Singh, Param Vir, Tan, Yong and Mookerjee, Vijay "Network Effects: The Influence of Structural Social Capital on Open Source Project Success," working paper (April 2008), available at SSRN: <http://ssrn.com/abstract=1111868>.

Stewart, K. J., Ammeter, A. P., and Maruping, L. M. "Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects," *Information Systems Research* (17:2), 2006, pp. 126-144.

Suchan, J. and Hayzak, G. "The communication characteristics of virtual teams: a case study," *IEEE Transactions on Professional Communication* (44:3), 2001, pp. 174-186.

Thomas and Hunt. "Open source ecosystems," *IEEE Software* (21:4), 2004, pp. 89-91.

Wasserman, S. and Faust, K. (eds.). *Social Network Analysis*, Cambridge University Press, Cambridge, 1994.

Weber S. (eds.). *The Success of Open Source*, Harvard University Press, Cambridge, 2004

Wellman, B., and Berkowitz, S. D. (eds.). *Social Structures: A Network Approach*, Cambridge University Press, Cambridge, 1998.

Wheeler, D. A. "Why open source software/free software (OSS/FS)? Look at the number!" accessed from ([http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html)), December 31, 2003.

Wooldridge, J. M. *Introductory Econometrics: a Modern Approach*, South-Western College Publication, Australia; Cincinnati, Ohio, 2003.

Xu, J., Gao, Y., Christley, S., and Madey, G. "A topological analysis of the open source software development community," in *Proceedings of 38th Hawaii International Conference on System Sciences*, Hawaii, USA, 2005.



## Appendix

*Project Centrality and Leadership Centrality:* Centrality can be defined on an individual or overall level for a network. Degree centrality adopted in this chapter was defined by Freeman (1979). Degree centrality represents a position's potential for activity in communication, and signifies a group member who is "in the thick of things". Degree centrality refers to the number of other positions in the network in direct contact with a given position. Freeman defined it as

$$C_{Degree} = \frac{\sum n_x}{N-1}$$

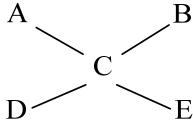
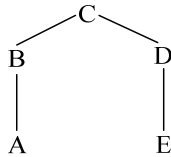
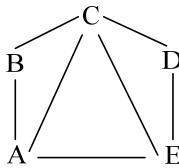
Where  $n_x$  = number of other positions in direct contact with position  $x$ ;  $N$  = number of positions in the network. In this study, leadership centrality refers to the degree centrality of the project manager if there is only one manager of the project; or refers to the average degree centrality of the project managers if there are more than one managers of the project.

We also examine the network centrality. In this study, project centrality refers to the network centrality on the project level. Freeman expressed the degree of inequality or variance in the network as a percentage of that of a perfect star network of the same size. Freeman defined network centrality as

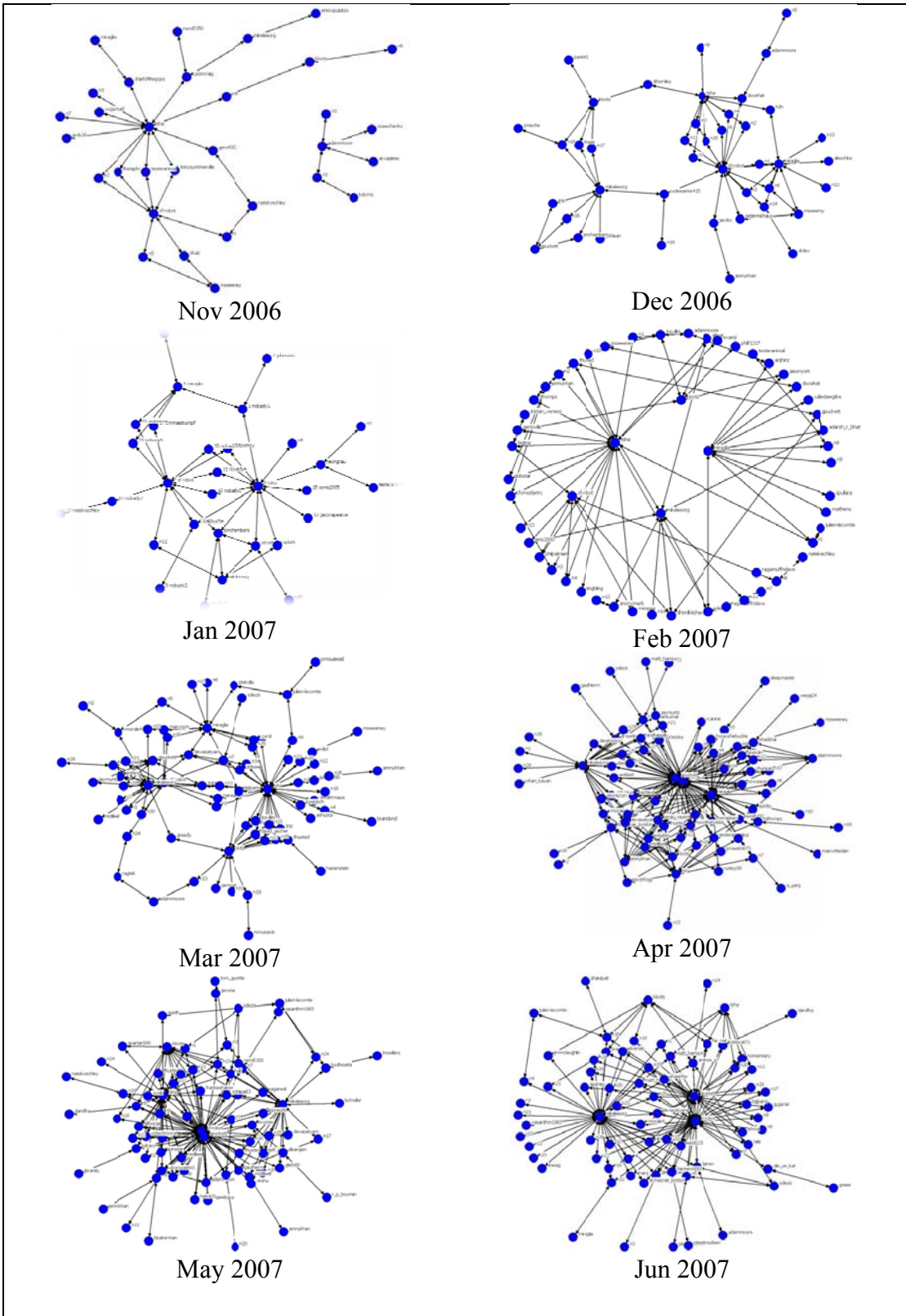
$$network\ centrality = \frac{\sum (c_{max} - c_i)_{current}}{\sum (c_{max} - c_i)_{star}}$$

Where  $c_{\max}$  = maximum degree in the network,  $c_i$  = degree centrality of each individual, “*current*” represents the target network, and “*star*” represents the perfect star network of the same size as target network.

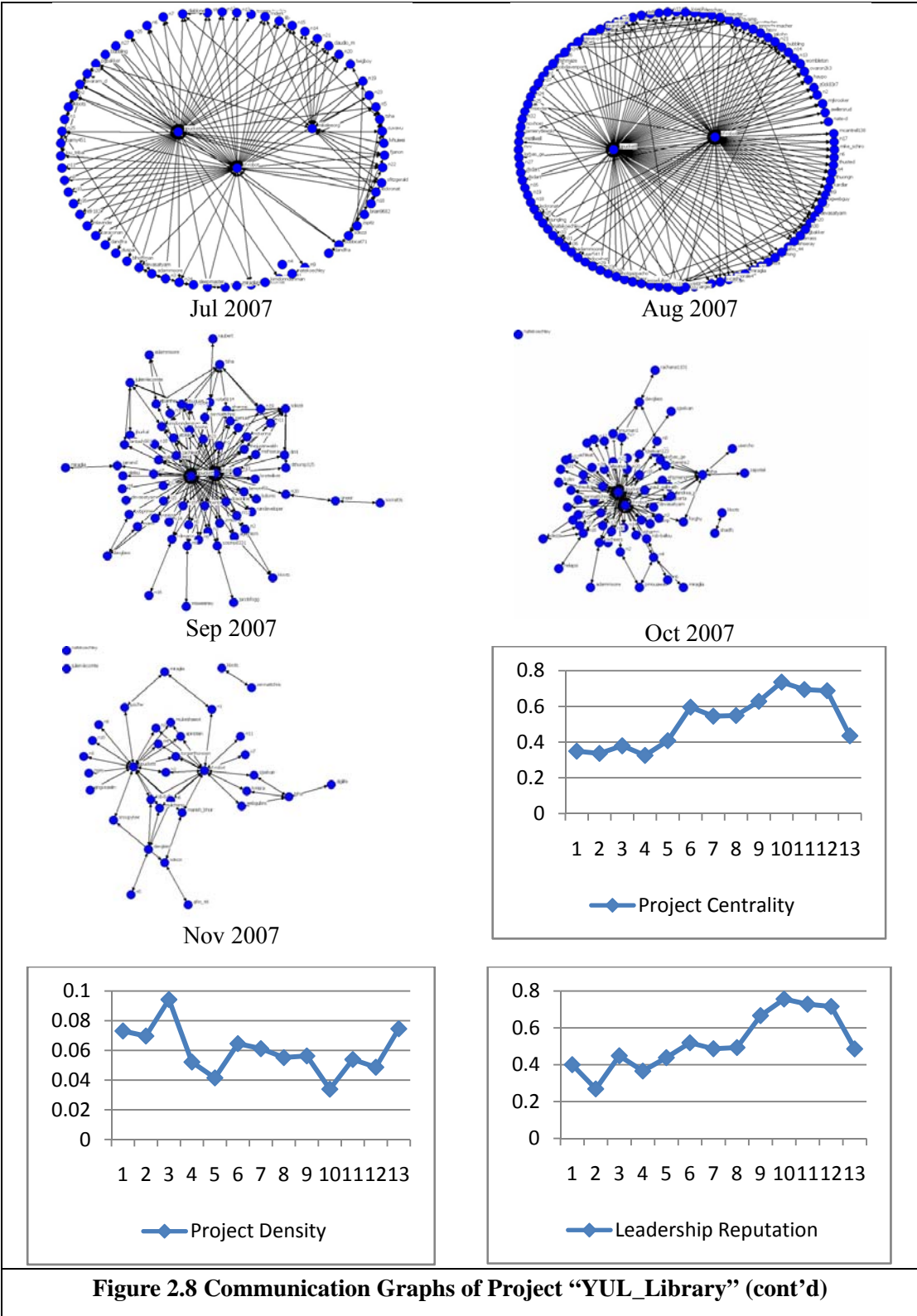
All the individual’s centrality and network centrality can be calculated by social network analysis software: Ucinet. Table 2.8 shows examples of degree centrality and network centrality in the five-person networks.

Table 2.8 Examples of Centrality		
Network	Position	Degree Centrality
	A	0.25
	B	0.25
	C	1.00
	D	0.25
	E	0.25
	Network	100%
	A	0.25
	B	0.50
	C	0.50
	D	0.50
	E	0.25
	Network	16.67%
	A	0.75
	B	0.50
	C	1.00
	D	0.50
	E	0.75
	Network	50%

*Social Network Graphs:* To better understand the communication patterns, we give the communication graphs of an example open source project (project “YUI Library”) of each month in Figure 2.7 and Figure 2.8. To let the readers clearly know how the communication patterns change, we also provide the data of communication patterns of each month in Table 2.9.



**Figure 2.7 Communication Graphs of Project "YUL\_Library"**



	Project Centrality (%)	Project Density (%)	Leadership Centrality (%)
Nov 2006	39.94	7.31	40.00
Dec 2006	33.66	6.97	26.83
Jan 2007	37.93	9.43	44.83
Feb 2007	32.54	5.22	36.54
Mar 2007	40.79	4.15	43.84
Apr 2007	59.59	6.46	51.90
May 2007	54.51	6.12	48.68
Jun 2007	54.87	5.52	49.32
Jul 2007	62.83	5.63	66.67
Aug 2007	73.60	3.39	75.68
Sep 2007	69.42	5.39	72.86
Oct 2007	68.79	4.87	71.64
Nov 2007	43.53	7.46	48.57

In the main contexts, we did not give the full table of the main results because of space limitation. In Table 2.10, we give the full results with the statistics of each project and month dummies (all the results are estimated by STATA).

	Obs	R-sq	Chi-sq	p
Log Activity	1004	0.312	2504.300	0.000
Log Popularity	1004	0.968	141573.900	0.000
Project Centrality	1004	0.145	523.240	0.000
Project Density	1004	0.108	361.620	0.000
Leadership Centrality	1004	0.345	1989.320	0.000
Leadership Centrality * Participation	1004	0.590	3534.050	0.000
	Coef.	S.E.	z	p> z
<b>Log Activity</b>				
Log Popularity	1.506	0.268	5.630	0.000
Project Centrality	0.026	0.387	0.070	0.947
Project Density	-2.176	0.475	-4.580	0.000
Leadership Centrality	1.403	0.389	3.600	0.000
Participation	0.469	0.092	5.080	0.000
Leadership Centrality * Participation	0.199	0.052	3.860	0.000
Project Complexity	-0.166	0.034	-4.900	0.000
Licence (Restrictive)	10.792	1.871	5.770	0.000
month7	0.028	0.036	0.790	0.432
month8	-0.031	0.036	-0.840	0.398
month9	0.058	0.036	1.600	0.110
month10	0.043	0.035	1.220	0.223

month11	0.069	0.036	1.920	0.055
month12	0.078	0.036	2.170	0.030
month13	0.127	0.036	3.500	0.000
project1	-1.630	0.530	-3.070	0.002
project2	-10.502	1.960	-5.360	0.000
project3	-4.286	0.794	-5.400	0.000
project4	-4.393	0.954	-4.600	0.000
project5	-4.047	1.027	-3.940	0.000
project6	-9.876	1.899	-5.200	0.000
project7	-9.264	1.719	-5.390	0.000
project8	9.595	1.492	6.430	0.000
project9	1.963	0.337	5.830	0.000
project10	8.547	1.426	5.990	0.000
project11	-2.582	0.626	-4.120	0.000
project12	-7.337	1.263	-5.810	0.000
project13	0.398	0.152	2.620	0.009
project14	3.534	0.515	6.860	0.000
project15	-3.884	0.790	-4.920	0.000
project16	-0.695	0.321	-2.170	0.030
project17	0.887	0.195	4.540	0.000
project18	-4.318	0.796	-5.430	0.000
project19	5.061	0.869	5.820	0.000
project20	-5.742	1.184	-4.850	0.000
project21	-0.004	0.201	-0.020	0.983
project22	-7.724	1.492	-5.180	0.000
project23	-6.591	1.149	-5.740	0.000
project24	-1.573	0.309	-5.100	0.000
project25	3.940	0.756	5.210	0.000
project26	-3.020	0.498	-6.070	0.000
project27	-7.539	1.355	-5.560	0.000
project28	4.463	0.721	6.190	0.000
project29	-0.973	0.319	-3.050	0.002
project30	3.888	0.714	5.450	0.000
project31	2.892	0.508	5.690	0.000
project32	-0.756	0.206	-3.670	0.000
project33	-6.238	1.120	-5.570	0.000
project34	8.361	1.369	6.110	0.000
project35	(dropped)			
project36	-3.051	0.542	-5.630	0.000
project37	-4.488	0.835	-5.380	0.000
project38	-8.332	1.574	-5.290	0.000
project39	-7.291	1.488	-4.900	0.000
project40	-3.760	0.853	-4.410	0.000
project41	-2.668	0.702	-3.800	0.000
project42	-7.385	1.401	-5.270	0.000
project43	-6.780	1.278	-5.300	0.000
project44	-6.744	1.233	-5.470	0.000
project45	-8.362	1.455	-5.750	0.000
project46	-5.167	0.943	-5.480	0.000
project47	-0.400	0.208	-1.920	0.055
project48	-5.888	1.155	-5.100	0.000

project49	4.180	0.755	5.540	0.000
project50	7.202	1.242	5.800	0.000
project51	-3.960	0.717	-5.520	0.000
project52	9.419	1.332	7.070	0.000
project53	-0.116	0.198	-0.590	0.557
project54	-6.980	1.208	-5.780	0.000
project55	-2.878	0.573	-5.020	0.000
project56	7.707	1.287	5.990	0.000
project57	-8.782	1.587	-5.530	0.000
project58	-1.702	0.311	-5.480	0.000
project59	-2.805	0.698	-4.020	0.000
project60	-2.261	0.410	-5.520	0.000
project61	-5.984	1.211	-4.940	0.000
project62	-7.293	1.340	-5.440	0.000
project63	-5.875	0.996	-5.900	0.000
project64	3.442	0.650	5.290	0.000
project65	-6.531	1.287	-5.070	0.000
project66	-6.161	1.360	-4.530	0.000
project67	-6.095	1.058	-5.760	0.000
project68	-8.788	1.542	-5.700	0.000
project69	3.761	0.616	6.110	0.000
project70	-3.334	0.621	-5.370	0.000
project71	-7.999	1.433	-5.580	0.000
project72	-5.700	1.089	-5.240	0.000
project73	-6.169	1.071	-5.760	0.000
project74	-7.559	1.390	-5.440	0.000
project75	-5.601	0.951	-5.890	0.000
project76	-7.056	1.126	-6.270	0.000
project77	-7.104	1.263	-5.620	0.000
project78	-2.504	0.538	-4.650	0.000
project79	10.329	1.716	6.020	0.000
project80	-3.000	0.522	-5.750	0.000
project81	-4.201	0.807	-5.210	0.000
project82	-11.167	1.772	-6.300	0.000
project83	-6.108	1.092	-5.590	0.000
project84	-5.881	0.998	-5.900	0.000
project85	-6.722	1.165	-5.770	0.000
project86	-3.529	0.614	-5.750	0.000
project87	-5.655	0.978	-5.780	0.000
project88	3.612	0.507	7.130	0.000
project89	-4.376	0.909	-4.820	0.000
project90	-7.207	1.129	-6.380	0.000
project91	-5.901	1.107	-5.330	0.000
project92	(dropped)			
project93	6.009	1.004	5.990	0.000
project94	-2.708	0.598	-4.530	0.000
project95	1.843	0.321	5.750	0.000
project96	6.105	1.086	5.620	0.000
project97	1.626	0.284	5.720	0.000
project98	6.309	1.090	5.790	0.000
project99	-9.100	1.640	-5.550	0.000

project100	-10.311	2.073	-4.970	0.000
project101	(dropped)			
project102	6.858	1.215	5.640	0.000
project103	-9.090	1.730	-5.250	0.000
project104	-9.265	1.745	-5.310	0.000
project105	8.122	1.285	6.320	0.000
project106	-1.537	0.369	-4.170	0.000
project107	6.760	1.198	5.640	0.000
project108	(dropped)			
project109	-6.378	1.364	-4.680	0.000
project110	-8.121	1.680	-4.840	0.000
project111	2.269	0.366	6.200	0.000
project112	-7.943	1.506	-5.280	0.000
project113	-8.434	1.503	-5.610	0.000
project114	-3.359	0.625	-5.370	0.000
project115	9.356	1.626	5.750	0.000
project116	3.736	0.562	6.650	0.000
project117	-4.709	0.834	-5.650	0.000
project118	-2.086	0.403	-5.170	0.000
project119	-4.841	1.076	-4.500	0.000
project120	-5.816	1.160	-5.010	0.000
project121	-4.425	0.848	-5.220	0.000
project122	8.973	1.501	5.980	0.000
project123	-7.397	1.228	-6.030	0.000
project124	-1.594	0.361	-4.420	0.000
project125	-6.360	1.147	-5.540	0.000
project126	-9.987	1.661	-6.010	0.000
constant	-13.247	2.406	-5.500	0.000
<b>Log Popularity</b>				
Log Activity	0.631	0.073	8.690	0.000
Project Centrality	-0.366	0.187	-1.950	0.051
Project Density	1.824	0.217	8.420	0.000
Leadership Centrality	-0.845	0.195	-4.340	0.000
Participation	0.009	0.013	0.710	0.479
Leadership Centrality * Participation	-0.101	0.025	-3.970	0.000
Project Age	0.015	0.001	15.620	0.000
Programming Language (C/C++)	0.262	0.064	4.100	0.000
Target Audience (end user)	0.386	0.095	4.060	0.000
month7	-0.034	0.018	-1.890	0.059
month8	-0.011	0.017	-0.660	0.508
month9	-0.083	0.019	-4.430	0.000
month10	-0.088	0.018	-4.900	0.000
month11	-0.122	0.019	-6.520	0.000
month12	-0.138	0.020	-6.930	0.000
month13	-0.190	0.020	-9.440	0.000
project1	-0.520	0.090	-5.760	0.000
project2	3.695	0.100	37.120	0.000
project3	0.995	0.129	7.700	0.000
project4	(dropped)			
project5	0.010	0.226	0.040	0.964
project6	2.561	0.068	37.440	0.000



project7	2.322	0.071	32.630	0.000
project8	-0.842	0.102	-8.260	0.000
project9	3.648	0.092	39.610	0.000
project10	0.000	0.122	0.000	0.997
project11	-0.687	0.082	-8.390	0.000
project12	1.208	0.104	11.630	0.000
project13	-2.085	0.085	-24.430	0.000
project14	-0.272	0.088	-3.090	0.002
project15	-1.641	0.126	-12.990	0.000
project16	-1.742	0.084	-20.700	0.000
project17	3.234	0.096	33.860	0.000
project18	-0.127	0.130	-0.980	0.326
project19	0.877	0.115	7.620	0.000
project20	1.348	0.171	7.900	0.000
project21	-2.421	0.099	-24.500	0.000
project22	1.739	0.103	16.820	0.000
project23	0.435	0.142	3.060	0.002
project24	-1.298	0.109	-11.870	0.000
project25	1.016	0.122	8.340	0.000
project26	-1.979	0.092	-21.510	0.000
project27	2.108	0.104	20.260	0.000
project28	1.206	0.109	11.110	0.000
project29	-0.611	0.126	-4.860	0.000
project30	1.200	0.105	11.390	0.000
project31	2.060	0.087	23.690	0.000
project32	-1.106	0.110	-10.010	0.000
project33	1.093	0.141	7.750	0.000
project34	(dropped)			
project35	(dropped)			
project36	-0.512	0.125	-4.110	0.000
project37	0.503	0.105	4.790	0.000
project38	2.692	0.074	36.200	0.000
project39	1.851	0.078	23.820	0.000
project40	-0.025	0.090	-0.270	0.786
project41	-0.847	0.072	-11.800	0.000
project42	1.521	0.083	18.360	0.000
project43	2.044	0.107	19.190	0.000
project44	1.969	0.092	21.410	0.000
project45	2.096	0.088	23.870	0.000
project46	0.421	0.084	4.980	0.000
project47	-2.880	0.082	-35.010	0.000
project48	0.320	0.081	3.930	0.000
project49	1.660	0.132	12.620	0.000
project50	-1.274	0.103	-12.420	0.000
project51	-0.717	0.121	-5.940	0.000
project52	-1.506	0.117	-12.840	0.000
project53	-1.589	0.116	-13.660	0.000
project54	1.505	0.097	15.460	0.000
project55	-1.310	0.102	-12.830	0.000
project56	-0.595	0.125	-4.750	0.000
project57	1.342	0.083	16.270	0.000

project58	-2.057	0.134	-15.320	0.000
project59	-1.299	0.131	-9.890	0.000
project60	-0.967	0.117	-8.240	0.000
project61	1.194	0.111	10.710	0.000
project62	1.593	0.080	19.820	0.000
project63	0.873	0.107	8.130	0.000
project64	1.438	0.120	11.970	0.000
project65	1.085	0.091	11.950	0.000
project66	0.878	0.132	6.630	0.000
project67	0.054	0.101	0.530	0.594
project68	0.790	0.106	7.430	0.000
project69	0.951	0.085	11.190	0.000
project70	-1.285	0.088	-14.550	0.000
project71	1.894	0.106	17.830	0.000
project72	0.612	0.118	5.200	0.000
project73	1.079	0.112	9.610	0.000
project74	0.836	0.099	8.410	0.000
project75	0.647	0.127	5.090	0.000
project76	0.271	0.134	2.030	0.042
project77	1.077	0.122	8.820	0.000
project78	-0.778	0.102	-7.660	0.000
project79	-2.575	0.074	-34.660	0.000
project80	0.317	0.097	3.260	0.001
project81	-0.236	0.121	-1.960	0.050
project82	1.280	0.192	6.650	0.000
project83	1.382	0.127	10.900	0.000
project84	1.204	0.111	10.800	0.000
project85	1.468	0.103	14.280	0.000
project86	-1.089	0.125	-8.710	0.000
project87	0.149	0.096	1.560	0.120
project88	-2.009	0.115	-17.510	0.000
project89	-0.374	0.175	-2.140	0.032
project90	-0.749	0.133	-5.640	0.000
project91	1.529	0.108	14.180	0.000
project92	-2.958	0.086	-34.270	0.000
project93	0.088	0.143	0.610	0.540
project94	-0.923	0.107	-8.640	0.000
project95	(dropped)			
project96	0.162	0.135	1.200	0.230
project97	-3.007	0.131	-22.880	0.000
project98	0.560	0.118	4.740	0.000
project99	3.503	0.096	36.590	0.000
project100	4.164	0.126	32.980	0.000
project101	-0.485	0.092	-5.290	0.000
project102	1.544	0.113	13.680	0.000
project103	1.984	0.084	23.600	0.000
project104	2.823	0.079	35.900	0.000
project105	-0.156	0.080	-1.950	0.051
project106	-2.330	0.091	-25.510	0.000
project107	0.252	0.130	1.940	0.052
project108	3.792	0.100	38.060	0.000

project109	(dropped)			
project110	0.777	0.107	7.280	0.000
project111	1.681	0.103	16.350	0.000
project112	1.706	0.082	20.800	0.000
project113	2.867	0.106	27.170	0.000
project114	0.124	0.109	1.130	0.257
project115	-1.916	0.102	-18.850	0.000
project116	1.199	0.115	10.430	0.000
project117	-0.710	0.118	-6.040	0.000
project118	-0.836	0.119	-7.010	0.000
project119	0.899	0.205	4.390	0.000
project120	0.065	0.082	0.800	0.426
project121	0.189	0.128	1.480	0.140
project122	-2.248	0.087	-25.710	0.000
project123	0.254	0.123	2.060	0.039
project124	-0.558	0.093	-5.990	0.000
project125	0.902	0.144	6.260	0.000
project126	1.327	0.103	12.910	0.000
constant	3.054	0.156	19.590	0.000
<b>Project Centrality</b>				
Project Centrality (-1)	0.288	0.023	12.550	0.000
Project Centrality (-2)	0.187	0.024	7.920	0.000
Project Centrality (-3)	0.093	0.027	3.450	0.001
Project Centrality (-4)	0.095	0.028	3.420	0.001
Project Centrality (-5)	0.042	0.027	1.550	0.122
constant	0.147	0.018	8.030	0.000
<b>Project Density</b>				
Project Density (-1)	0.221	0.024	9.110	0.000
Project Density (-2)	0.170	0.024	7.040	0.000
Project Density (-3)	0.116	0.025	4.650	0.000
Project Density (-4)	0.113	0.025	4.550	0.000
Project Density (-5)	0.081	0.025	3.180	0.001
constant	0.076	0.010	7.560	0.000
<b>Leadership Centrality</b>				
Leadership Centrality (-1)	0.299	0.020	15.180	0.000
Leadership Centrality (-2)	0.226	0.020	11.080	0.000
Leadership Centrality (-3)	0.143	0.022	6.380	0.000
Leadership Centrality (-4)	0.117	0.023	5.160	0.000
Leadership Centrality (-5)	0.105	0.022	4.820	0.000
constant	0.061	0.013	4.590	0.000
<b>Leadership*Participation</b>				
Leadership Centrality (-1) * Participation	0.248	0.023	10.990	0.000
Leadership Centrality (-2) * Participation	0.227	0.024	9.580	0.000
Leadership Centrality (-3) * Participation	0.190	0.024	7.990	0.000
Leadership Centrality (-4) * Participation	0.243	0.023	10.410	0.000
Leadership Centrality (-5) * Participation	0.051	0.024	2.150	0.031
constant	0.065	0.042	1.570	0.117

## **Chapter 3. Optimal Software Design and Pricing in the Presence of Open Source Software**

---

### **3.1 Introduction**

OSS refer to those programs “whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or the modified program without having to pay royalties to previous developers” (Wheeler, 03). Since the first open source software was developed by Richard Stallman (GNU) in the 70’s, there have been a myriad of open source applications, ranging from common office suites such as StarOffice, to database (mySQL) and thousands of specialized scientific applications<sup>9</sup>. Open source software challenges the traditional idea that software must be licensed and cannot be modified according to individual needs. It makes source code publicly available for free usage

---

<sup>9</sup> For more details, please refer to Scientific Applications on Linux (SAL) (<http://sal.linet.gr.jp/index.shtml>)

and modification, including bug fixing and customizing features. Ever since the burgeoning of open source software, it has attracted more and more attentions from individual users and organizations due to its “free of charge” and “freedom of distribution and modification.” A recent survey by Netcraft ([www.netcraft.com](http://www.netcraft.com)) suggests that the market share of Apache grows rapidly and has exceeded Microsoft’s IIS.

The growing popularity of open source software has stimulated wide-spread interest in academia. Numerous studies have been dedicated to decode the myth of open source participation and the relationship between open source and closed source software. Lerner and Tirole (2002) suggest that developers contribute to open source projects because of immediate and delayed economic benefits (signaling incentives) from open source development. Based on a case study of (L)A)TEX, Gaudel (2003) concludes that open source inspired commercial software more than the reverse. On the other hand, the software industry, which traditionally operates on basis of software licensing, is more concerned with how to respond strategically to the open source phenomenon. Microsoft, the once strongest opponent to open source movement, has recently decided to include open source code in its shipping of Windows Server 2003 Cluster Edition (Galli 2005). Sun Microsystems is actively involved in the development of open source based Java projects in a hope to promote Java-related products. Without a doubt, the profitability of a commercial firm is affected (if not threatened) when the consumers are offered with an alternative free option other than its proprietary software. Commercial firms are

now confronted with the critical decision on how to optimally compete with open source software.

The objective of this paper is to answer the key question about how a profit-seeking software firm should compete with open source software. Although competition has been the classic research topic in economic literature, the competition between open source and proprietary software has the following distinct features that deserve further analysis: (1) traditional duopoly competition model studies the equilibrium of two profit-making firms while open source software is free of charge and can't be made for profit by itself; (2) traditional competition models normally study the optimal pricing while in case of software competition, the software producers has two arms to fight with competition – pricing and product design. For instance, if the commercial product is quite similar to open source products, the commercial firm faces fierce competition, but if the proprietary software is highly differentiated, the product might appeal to a small niche market only; (3) software products exhibit positive network externalities, which further complicates the decision of optimal price and product design.

In this essay, we investigate two competition models: one-dimensional Hotelling model and two-dimensional vertical differentiation model. We first employ a stylized Hotelling model to study the optimal pricing and design of proprietary software in the presence of competitive open source software. We address the following research questions: (1) what is the impact of open source software's positioning (design) on the

optimal price, design and profit of the proprietary software; (2) how is social welfare affected by the positioning of open source software; (3) what are the firm's optimal strategy and profit when there's positive network effect. Our analysis suggests that the profit of the commercial firm is dependent on fit cost and the positioning (design) of open source software. In particular, the commercial firm is best off when the open source software targets more specialized users. Furthermore, if the open source software targets the more specialized users, the commercial firm should serve only a proportion of the whole population. The market share of the commercial firm is largest when the open source software caters to users with extreme requirements, and a predatory pricing strategy is adopted by the commercial firm. In addition, we find that the social welfare is maximized when open source software targets more specialized users. Finally, a simple analysis of a special case suggests that the commercial firm gains more profit when there's positive network externalities. However, since both open source and proprietary software benefit from network externalities, the commercial firm's profit is not monotonically increasing or decreasing in network intensity.

In the first model, we use one dimension to represent consumer taste. We did not give the details of the consumer taste. In the second model, we try to analyze the consumer taste in a specific way. Let's consider two fundamental features of a software product: functionality and usability. Functionality refers to how many functions the software can provide; while usability represents how friendly the interface is. When choosing a

software product, average users first consider whether it contains the necessary functions and then they want to know whether it is easy to use. Generally speaking, software firms have to provide a friendly user interface in order to allow users to easily learn and use the software. At the same time, the firms must provide the basic as well as advanced functions to cater to the needs of different users. Because consumers have different needs and knowledge, there's no single perfect design (usability and functionality) that fits everybody. In real business, commercial software companies (such as Microsoft) tend to target (naïve) customers with less computer skill and experience, while open source software (such as Linux) tends to target (sophisticated) customers with a certain level of computer and programming skill. In this model, we study the optimal design and pricing strategies for a monopoly commercial software firm to compete with open source software. The commercial software producer has to invest in a certain amount cost to achieve a certain level of usability and functionality for its product. We establish a two-dimensional vertical differentiation model to derive the optimal price and design of the commercial software product given the characteristics of the open source software.

### **3.2 Literature Review**

The open source phenomenon has captured attention by academic scholars in various areas. One stream of literature deals with the motivation of participating in open source projects. Lerner and Tirole (2002) explain the immediate and delayed benefit from



engaging in open source development. They find that the developers improve programming ability and get recognition from peer developers. In the long run, they gain reputation and attain more desirable jobs. Bonaccorsi and Rossi (2003) find similar results with that of Lerner and Tirole (2002). Schmidt and Schnitzer (2002) divide programmers into different types: those who are devoted to bug fixing (sophisticated users), those who are devoted to software development (signaling incentives), and those who are employed by the commercial software companies. Lakhani and Wolf (2003) use a web-based survey and find that joy-based intrinsic motivations, namely how a person feels from a creative project, is the strongest and most pervasive driver. Roberts et al (2004) evaluate the theoretical model using survey from Apache web server projects. Their results reveal that the different types of motivations lead to different contributions to open source projects. The study of motivations of open source development helps to understand the distinctive characteristics of open source software. For example, it explains why open source software usually has an inferior graphical interface and documentation compared to its proprietary counterparts.

A second stream of research compares the quality of open source and proprietary software, which has led to contradictory views. Kuan (2001) and Johnson (2004) believe that open source software provides higher quality than closed source software. At the same time, many industrial experts have raised concerns with the quality of open source software, criticizing its lack of customer support, security and reliability of open

source software (Shell 2005, Massel 2005). The different views imply that open source and proprietary software are adopted (preferred) by different groups of users. Bessen (2005) finds that the firms with strong software development capability or complex, specialized needs will most likely use open source software. Similarly, Gaudeul (2003) points out that because of lack of interface, open source software is left with users who cannot afford proprietary software (students), those who need its advanced capabilities (academics) and those who valued the flexibility to customize its own product (sophisticated users).

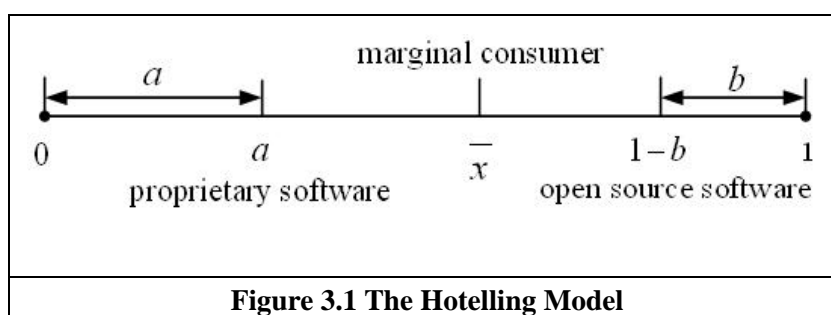
Our research is mostly related to a third stream of research that focuses on the competition between open source and commercial software. Casadesus-Masanell and Ghemawat (2003) examine the dynamic competition between commercial software and open source by analyzing the demand-side learning effect. They conclude that in equilibrium the open source software will either coexist with commercial software or be driven out of the market. Gaudeul (2004) explores the market equilibrium by considering the interaction between software users and developers. It is assumed that users have different valuations of software features and developers have heterogeneous programming costs. Gaudeul (2004) concludes that in equilibrium, open source software is either driven out of market by proprietary software or coexists with proprietary software. If open source software does survive, it is used either by users who like more advanced features and do not care about awkward user interfaces or those low-income consumers who cannot afford to buy the proprietary software. Our

research differs from the work by Gaudoul (2004) in that we use horizontal differentiation model instead of the vertical differentiation adopted by Gaudoul (2004). In general, open source or proprietary software is preferred by different types of users: expert users like open source software because of its more advanced features and the flexibility to modify source code; while ordinary users prefer proprietary software since it offers better interface and more reliable customer support (Nichols & Twidale, 2003). In addition, we study the impact of open source software's positioning on the optimal design and pricing of proprietary software, while Gaudoul (2004) concentrates on characterizing the equilibrium market share of open source and proprietary software.

### **3.3 Model 1**

Consider two differentiated but competitive software products in market. One is offered by a commercial firm at price  $P$  and the other is free as open source software (OSS). Although in many cases, a commercial firm can make profit by selling complementary applications or services to open source software, we focus on competition between a free open source and commercial software to study the optimal strategies of a commercial software firm in the presence of not-for-profit open source software. Software users, which differ in computing skills and knowledge, have different tastes of software products. Consumer tastes can be manifested in multiple ways. For example, expert users are more interested in advanced functions and the flexibility to customize source code while ordinary users are more interested in better interface for the ease of

use. We model consumer's taste as her preference between open source and proprietary software. We adopt a Hotelling model to represent consumer taste, denoted by  $x$ , which is distributed uniformly in the interval between 0 and 1. The proprietary software and open source software are situated at points  $a$  and  $1-b$ , respectively (see Figure 3.1). Without loss of generality, we assume that  $(1-b) > 1/2$ . The opposite scenario can be studied by simply reversing the analysis in this paper.



**Figure 3.1 The Hotelling Model**

We assume that the location of OSS ( $1-b$ ) is exogenously given. This is due to the idiosyncratic development process of open source projects. Open source software is developed by voluntary programmers. A new version of OSS is released whenever there's sufficient contribution by the development community. Unlike commercial firms which can set a long-term development objective, open source projects are less coordinated and organized. As a result, the design of open software is more random in nature compared to proprietary software. Therefore, we first explore commercial firm's optimal design and pricing strategies when the location (design) of OSS is given. Later, we compare the firm's profit and social welfare with respect to different settings of OSS location. In a one-shot model, we assume the location of the OSS remain fixed

during the period of study. In addition, we assume consumers have perfect information about the location (design) of OSS and proprietary software. With the pervasive usage of Internet, users nowadays can easily find out information about a software product at low cost, such as via online forums and professional reviews.

Consumer's position on the Hotelling line represents her ideal location of software product. Each consumer incurs a fit cost that is proportional to the distance between her ideal product location and the location of proprietary or open source software. Let  $t$  be the unit fit cost and  $v$  be consumer's reservation price for each type of software. Then the net utility for a consumer located at point  $x$  from using open source ( $U_{os}$ ) and proprietary software ( $U_{cs}$ ), are defined in Eq. (1).

$$\begin{aligned} U_{os}(x) &= v - t|1 - b - x| \\ U_{cs}(x) &= v - t|x - a| - P \end{aligned} \tag{1}$$

Consumers choose OSS or proprietary software by comparing her net utility from each type of software. The marginal consumer, denoted by  $\bar{x}$ , is indifferent between open source and proprietary software. Since the consumers are uniformly distributed on the Hotelling line and the OSS is located at the right half of the line, the proprietary software must be located to the left of the OSS. Otherwise, the commercial firm will suffer from a smaller market share with the same price. As a result, the marginal consumer must be situated between the proprietary and OSS, see Figure 3.1. Specifically, the marginal consumer is described in Eq. (2).

$$v - t(\bar{x} - a) - P = v - t(1 - b - \bar{x}) \quad (2)$$

The commercial firm should optimally set the price and location of its proprietary software to maximize its profit. In order to focus on the competition between OSS and proprietary software, we don't consider the case when the marginal consumer receives negative utility. In other words, we assume that no consumer between  $a$  and  $1$  is "stranded" by both types of software. Therefore, the commercial firm has two options – cover the entire market demand together with OSS; or cover a proportion of market demand together with OSS. In the following subsections, we analyze commercial firm's optimal profit for each option. The optimal strategy is found by selecting the strategy that maximizes the firm's profit. Due to limitation of space, we don't provide detailed derivations. All proofs are available upon request.

### 3.3.1 Market is fully covered

When the market is fully covered, all consumers derive non-negative utilities from either type of software. It is required that the net utilities for consumers with largest fit cost, i.e., the consumer who are situated at point 0 and the marginal consumer situated at point  $\bar{x}$ , are non-negative. When the market is fully covered, all consumers to the left of the marginal consumer will purchase the proprietary software and consumers to the right of the marginal consumer will use OSS. The market share for the commercial firm is  $\bar{x}$ , which is described in Eq. (2). In sum, the profit maximization problem for

the commercial firm is formulated as Eq. (3) below. We solve for the constrained optimization problem by using Kuhn-Tucker conditions. The optimal price, location and profit of the commercial firm are summarized in Lemma 1.

$$\begin{aligned}
 \underset{P,a}{Max} \quad & \pi = \bar{x}P = \frac{t(1-b+a)-P}{2t} P \\
 \text{s.t.} \quad & U_{cs}(0) = v - ta - P \geq 0 \\
 & U_{cs}(\bar{x}) = v - t\left(\frac{t(1-b+a)-P}{2t} - a\right) - P \geq 0
 \end{aligned} \tag{3}$$

*Lemma 1.* If  $\frac{1}{2} < (1-b) < 2\frac{v}{t}$ , the market is fully covered by the OSS and proprietary software, and

a) If  $\frac{1}{2} < 1-b < \frac{5v}{3t}$ , the optimal price is  $P_1^* = \frac{t(1-b)+v}{4}$ ; the optimal location is

$$a_1^* = \frac{3v-t(1-b)}{4t}; \text{ the commercial firm's profit is } \pi_1^* = \frac{[t(1-b)+v]^2}{16t};$$

b) If  $\frac{5v}{3t} < 1-b < 2\frac{v}{t}$ , the optimal price is  $P_2^* = \frac{3v-t(1-b)}{2}$ ; the optimal location

is  $a_2^* = \frac{1}{2}\left(1-b-\frac{v}{t}\right)$ ; the commercial firm's profit is

$$\pi_2^* = \left(1-b-\frac{v}{t}\right)\left(\frac{3v-t(1-b)}{2}\right).$$

### 3.3.2 Market is not fully covered

When the market is not fully covered, some consumers located in the far left section of the Hotelling line derive negative utility from both OSS and proprietary software and

are thus left out of market. In such case, the furthest consumer who can be served by the proprietary software is the consumer who derives zero utility. The left most customer who will buy proprietary software ( $x_0$ ) is defined by Eq. (4)

$$U_{cs}(x_0) = v - t(a - x_0) - P = 0 \quad (4)$$

In correspondence, the demand (market share) of the proprietary software is  $\bar{x} - x_0$ . We formulate the profit maximization decision of the commercial firm in Eq. (5). The constrained optimization problem can be solved by using Kuhn-Tucker conditions. The optimal price, location and profit of the proprietary software are given in Lemma 2.

$$\begin{aligned} \underset{P,a}{Max} \quad & \pi = (\bar{x} - x_0)P = \frac{t(1-b-a) - 3P}{2t} P \\ \text{s.t.} \quad & U_{cs}(0) = v - ta - P < 0 \\ & U_{cs}(\bar{x}) = v - t\left(\frac{t(1-b+a) - P}{2t} - a\right) - P \geq 0 \end{aligned} \quad (5)$$

*Lemma 2.* If  $2\frac{v}{t} < (1-b) < 1$ , the market is not fully covered by OSS and proprietary software. The optimal price is  $P_3^* = \frac{v}{2}$ ; the optimal location is  $a_3^* = 1 - b - \frac{3v}{2t}$ ; the profit is  $\pi_3^* = \frac{v^2}{2t}$ .

### 3.3.3 Analysis of results

The optimal strategies prescribed by Lemma 1 and Lemma 2 are dependent on the ratio of product valuation and fit cost ( $v/t$ ) as well as the location of the OSS ( $1-b$ ). For



example, when  $v/t > 1/2$ , the condition in Lemma 2 is never satisfied since  $1-b < 1$ .

Proposition 1 summarizes the optimal strategies of the commercial firm under different conditions. Proposition 2 shows the optimal strategies with respect to different design of the open source software.

*Proposition 1.* The optimal price, location and profit of the commercial firm given fit cost ( $t$ ), product valuation ( $v$ ) and the location of the OSS ( $1-b$ ) is as follows, where  $P_i^*$ ,  $a_i^*$  and  $\pi_i^*$  ( $i=1, 2, 3$ ) are defined in Lemma 1 and 2.

(1) If  $0 < v/t < 3/10$ , the optimal strategy is to cover entire market by setting the price at  $P_2^*$  and locate at  $a_2^*$  when  $\frac{5v}{3t} < 1-b < 2\frac{v}{t}$ , while the optimal strategy is to cover partial market by setting the price at  $P_3^*$  and locate at  $a_3^*$  when  $2\frac{v}{t} < 1-b < 1$ ;

(2) If  $3/10 \leq v/t < 1/2$ , the optimal strategy is to cover entire market by setting the price at  $P_1^*$  and located at  $a_1^*$  when  $\frac{1}{2} < 1-b < \frac{5v}{3t}$  and set the price the price at  $P_2^*$  and locate at  $a_2^*$  when  $\frac{5v}{3t} < 1-b < 2\frac{v}{t}$ , while the optimal strategy is to cover partial market by setting the price at  $P_3^*$  and locate at  $a_3^*$  when  $2\frac{v}{t} < 1-b < 1$ ;

(3) If  $1/2 \leq v/t < 3/5$ , the optimal strategy is to cover entire market by setting the price at  $P_1^*$  and located at  $a_1^*$  when  $\frac{1}{2} < 1-b < \frac{5v}{3t}$  and set the price the price at  $P_2^*$  and locate at  $a_2^*$  when  $\frac{5v}{3t} < 1-b < 2\frac{v}{t}$ ;

(4) If  $v/t \geq 3/5$ , the optimal strategy is to cover the entire market by setting the price at  $P_1^*$  and located at  $a_1^*$ .

*Proposition 2.* The commercial firm's profit is at maximum when OSS targets more specialized users ( $2v/t < 1 - b < 1$ ), i.e.  $\pi_3^* > \pi_2^* > \pi_1^*$ . In addition, the commercial firm's market share is maximized when  $2v/t < 1 - b < 1$ , i.e.,  $\bar{x}_3^* - x_0 > \bar{x}_2^* > \bar{x}_1^*$ . The price of the proprietary software is lower when the market is not fully covered, i.e.,  $P_1^* > P_3^*, P_2^* > P_3^*$ .

Next we examine the impact of OSS positioning of on social welfare. Since the price transferred from consumers to the commercial firms cancel out when we calculate social welfare, we only need to examine the total fit cost and gross utility by the consumers. The total fit cost ( $TF$ ) is defined in Eq (6), where the lower bound of integration of the first term ( $L$ ) is 0 if market is fully covered and  $x_0$  if market is not fully covered. Total social welfare is calculated as  $v(1 - L) - TF$ .

$$TF = \int_L^a t(a - x)dx + \int_a^{\bar{x}} t(x - a)dx + \int_x^{1-b} t(1 - b - x)dx + \int_{1-b}^1 t(x - (1 - b))dx \quad (6)$$

*Proposition 3.* The social welfare is maximized when OSS targets more specialized users. Specifically, social welfare is at maximum when  $(1 - b) = 2v/t$ .

Proposition 1 suggests that the commercial firm is best off when the OSS caters to users with extreme requirements. In addition, we find that commercial firm does not need to serve the entire market if the OSS is situated at extreme locations. Interestingly,

although the commercial firm only targets partial consumers, its market share is still larger than the case when the market is fully covered. When the market is not fully covered, the commercial firm will set a low predatory price to command a large market share and subsequently gain greater profit. Proposition 2 suggests that the profit maximization objective by the commercial firm is also aligned with the objective of a social planner. Currently, most open source products are adopted by the niche market of low-income or high-skill users. Advocators of open source software often argue that the open source community has been improving the interface and documentation to make it more like their user-friendly proprietary counterparts. However, our analysis suggests that the social welfare might decrease with improved usability of open source products, as it reduces the differentiation between OSS and proprietary software. At the same time, commercial firm suffers from profit loss by the presence of a “closer-look” open source product.

### **3.3.4 The impact of network effect**

In this section, we examine the impact of network effect on commercial firm’s optimal strategy and profit. The competition between OSS and proprietary software becomes rather complicated in the presence of positive network externalities. For example, the marginal consumer is found between OSS and proprietary software without network effect. However, in the presence of positive network effect, the marginal consumer could be situated to the left of the proprietary software if the intensity of network

externalities for OSS is sufficiently large. It's beyond the space limit to examine all possible scenarios. Rather, we present the simplest case. That is, we assume that the network intensity is same for both OSS and proprietary software. Further, we study the special case where the fit cost is low ( $t < v$ ) and the OSS is situated at the extreme location ( $b = 0$ ). Notice that optimal strategy without network effect when  $t < v$  is specified by case (4) in Proposition 1, i.e., the market is fully covered. In comparison, we analyze the commercial firm's profit in the presence of network externalities by considering the case when the market is fully covered.

Let  $\gamma$  be the intensity of network effect for both OSS and proprietary software. Further, it is assumed that there's no cross-product externality between OSS and proprietary software. This is because open source and proprietary software are normally adopted by separate consumer camps with different tastes. When there's positive network externality, each consumer enjoys an increase of utility, characterized by  $\gamma Q_i$ , where  $Q_i$  ( $i = c, o$ ) is the total installed base of the software adopted by the particular consumer. The net utilities from open source ( $U'_{os}$ ) and closed source ( $U'_{cs}$ ) software with network externality for a consumer of type  $x$  is defined by Eq. (7).

$$\begin{aligned} U'_{os}(x) &= v - t(1 - x) + \gamma Q_o \\ U'_{cs}(x) &= v - t(x - a) - p + \gamma Q_c \end{aligned} \tag{7}$$

Let  $\bar{x}$  be the marginal consumer indifferent between OSS and proprietary software. If the market is fully covered, the size of the installed base for OSS is  $Q_o = 1 - \bar{x}$  and the

size of the installed base for proprietary software is  $Q_c = \bar{x}'$ . We assume that  $\gamma < t$ , which is required by the decrease of demand in price. The profit maximization problem for the commercial firm is expressed as follows.

$$\underset{p, a}{Max} \quad \pi = Q_c P = \frac{p + \gamma - t - ta}{2(\gamma - t)} P \quad (8)$$

$$s.t. \quad \gamma + at - t \leq P \quad (8a)$$

$$P \leq t - \gamma + 2a\gamma - at \quad (8b)$$

$$v - t(\bar{x}' - a) - P + \gamma\bar{x}' \geq 0 \quad (8c)$$

$$v - ta - P + \gamma\bar{x}' \geq 0 \quad (8d)$$

Conditions (8a) and (8b) specify that the marginal consumer is situated between proprietary (located at  $a$ ) and open source software (located at  $l$ ). Conditions (8c) and (8d) are required to ensure that market is fully covered, i.e., the net utilities for consumer at location  $0$  and  $\bar{x}'$  are non-negative. We apply Kuhn-Tucker conditions to solve the above constrained optimization problem. The solution is presented in Proposition 4. The impact of network externality is summarized in Proposition 5.

*Proposition 4.* In the presence of network externality, the optimal price, location and profit of the proprietary software when  $t < v$ ,  $b = 0$  are specified as follows.

(1) If  $0 < \gamma < 2/3t$ , the optimal price is  $P_1^* = \frac{1}{2}(t - \gamma)$ , the optimal location is

$$a_1^* = \frac{1}{2} \text{ and the commercial firm's profit is } \pi_1^* = \frac{1}{8}(2t - \gamma);$$

(2) If  $\frac{2}{3}t < \gamma < t$ , the optimal price is  $P_2^* = \frac{\gamma(\gamma-t)}{\gamma-2t}$ , the optimal location is

$$a_2'^* = \frac{2(\gamma-t)}{\gamma-2t} \text{ and the commercial firm's profit is } \pi_2'^* = \frac{\gamma(\gamma-t)}{\gamma-2t}.$$

*Proposition 5.* In the presence of network externality, the price of commercial software decreases, the market share of the proprietary software increases, and the commercial firm's profit increases. However, the commercial firm's profit is not monotonically increasing or decreasing in network externality intensity. Specifically,  $P_1^* > P_1'^* > P_2'^*$ ,  $Q_1^* < Q_1'^* < Q_2'^*$  and  $\pi_1'^* > \pi_2'^* > \pi_1^*$ .

In summary, we find the commercial firm benefits from positive network externality. More interestingly, we find that the commercial firm's profit is not strictly increasing or decreasing with network intensity. This is because both open source and proprietary users benefit from the positive network effect. In other words, larger network intensity renders both types of software more attractive to users.

### 3.4 Model 2

In the first model, we use one dimension to represent consumer taste. We did not give the details of the consumer taste. In this model, we try to analyze the consumer taste in a specific way. Let's consider two fundamental features of a software product: functionality and usability. Functionality refers to how many functions the software can provide; while usability represents how friendly the interface is. When choosing a

software product, average users first consider whether it contains the necessary functions and then they want to know whether it is easy to use. Generally speaking, software firms have to provide a friendly user interface in order to allow users to easily learn and use the software. At the same time, the firms must provide the basic as well as advanced functions to cater to the needs of different users. Because consumers have different needs and knowledge, there's no single perfect design (usability and functionality) that fits everybody. In real business, commercial software companies (such as Microsoft) tend to target (naïve) customers with less computer skill and experience, while open source software (such as Linux) tends to target (sophisticated) customers with a certain level of computer and programming skill. In this model, we study the optimal design and pricing strategies for a monopoly commercial software firm to compete with open source software. The commercial software producer has to invest in a certain amount cost to achieve a certain level of usability and functionality for its product. We establish a two-dimensional vertical differentiation model to derive the optimal price and design of the commercial software product given the characteristics of the open source software.

### **3.4.1 Model Setting**

We employ a two-dimensional vertical differentiation model to analyze the optimal strategies (price and product design) of a commercial software firm in the presence of the competing open source software. We focus on two basic characteristics of an

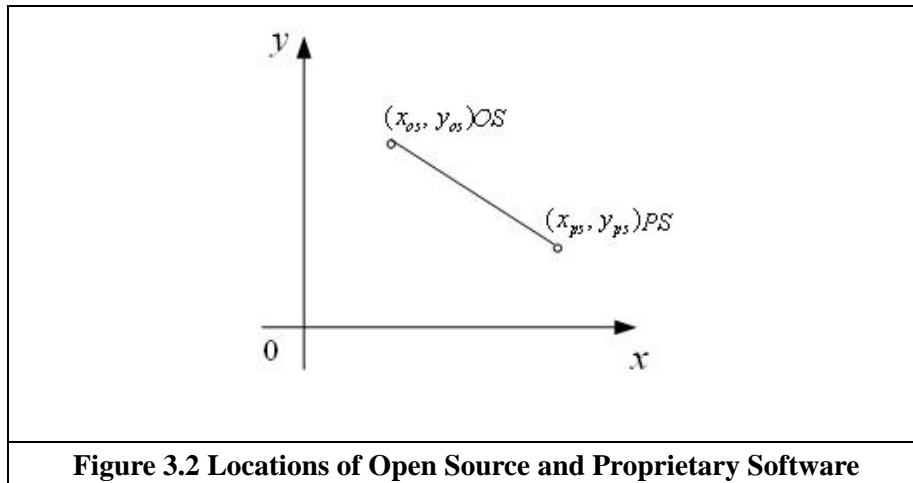
application: how friendly the interface is (usability) and how many functions the application provides (functionality). Due to different knowledge and experience with IT products, the consumers' taste (valuation) for usability and functionality are heterogeneous. We model the consumer's valuation of usability and functionality in an orthogonal coordinate, with each coordinate uniformly distributed in the interval  $(0, 1)$ . Specifically, the horizontal dimension ( $x$ -axis) represents the distribution of consumer's valuation of usability, denoted by  $\theta_1$ ; and the vertical dimension ( $y$ -axis) represents the distribution of consumer's valuation of functionality, denoted by  $\theta_2$ . In short, the consumer type is described by the pair  $(\theta_1, \theta_2)$ .

Each product (open source and proprietary software) is characterized by its usability and functionality, denoted by the pair  $(x, y)$ . We assume that there's no correlation between usability and functionality. In accordance with the consumer distribution, the product space is set in an orthogonal coordinate, with  $x$ -axis representing usability and  $y$ -axis representing functionality. The consumers with at most one unit demand can choose between a proprietary software and an open source software, which are located at point  $(x_{ps}, y_{ps})$  and  $(x_{os}, y_{os})$ , respectively (see Figure 3.2). We assume that the consumers' reservation prices are high enough such that they either choose the open source software or buy the commercial software. Let  $P$  be the price charged by the commercial firm. The consumer's net utility from the commercial software ( $U_{ps}$ ) and open source software ( $U_{os}$ ) can be described as follows:



$$\begin{aligned}
 U_{ps} &= \theta_1 x_{ps} + \theta_2 y_{ps} - P \\
 U_{os} &= \theta_1 x_{os} + \theta_2 y_{os}
 \end{aligned}
 \tag{9}$$

where  $(x_{ps}, y_{ps})$  and  $(x_{os}, y_{os})$  are the locations (characteristics) of commercial and open source software and  $(\theta_1, \theta_2)$  denotes the consumer type.

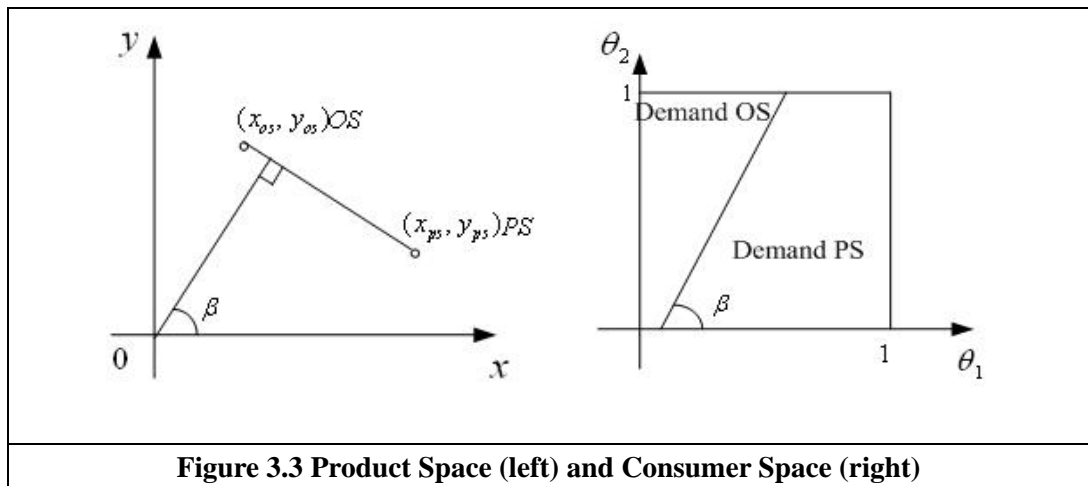


Previous studies (Lerner and Tirole, 2002; Bonaccorsi and Rossi, 2003; Schmidt and Schnitzer, 2002) of the motivation of open source projects suggests that developers like to work on open source projects which are interesting and challenging. From the computer experts' perspective, adding features is usually deemed as more interesting and challenging than user interface improvement since it is a better way to show off their talent. Therefore, developers of open source software are more interested in improving functionality than usability (Nichols and Twidale, 2003). On the other hand, commercial software firms (such as Microsoft) tend to target ordinary (naïve) customers who have less computer knowledge and skill. Therefore, commercial firms

have to provide a friendlier user interface in order to let users easily understand and use their software. To reflect business reality, we assume that open source software has a relative advantage on functionality, while proprietary software has a relative advantage on usability. That is,  $x_{ps} > x_{os}$  and  $y_{os} > y_{ps}$ . We don't consider the cases where either  $x_{ps} = x_{os}$  or  $y_{ps} = y_{os}$ , which reduce the problem to one dimension differentiation, which has been studied in the first model.

The characteristics (usability and functionality) of the open source software are assumed to be exogenous. This assumption is based on the nature of the open source and commercial software. First, since the developers voluntarily join open source projects, the developer camp is more fluid in an open source environment and the projects are less organized (Lerner and Tirole, 2002). Thus the open source projects usually do not have as clear direction as its commercial counterpart. Second, the quality (usability or functionality) of open source software is constricted by the capability and devotion of "freelance" developers, resulting in more uncertainty in usability and functionality. Third, the launch of a new version of commercial software usually involves costly and time-consuming preparation, such as market analysis and forecast, product design and price setting. In contrast, open source projects are not profit-driven and enjoy the flexibility of releasing new versions whenever there's sufficient improvement. Therefore, commercial software is less volatile compared to open source software.

To solve the two-dimensional vertical differentiation model, we need to examine the product space and the consumer space to determine demand for the commercial product (see Figure 3.3). As explained before, the open source software has an advantage over functionality and commercial software has an advantage over usability, i.e.,  $x_{ps} > x_{os}$  and  $y_{os} > y_{ps}$ . Let  $\beta$  be the slope of the line perpendicular to the line connecting two products, i.e.,  $\tan \beta = \frac{x_{ps} - x_{os}}{y_{os} - y_{ps}}$  (see Figure 3.3). That is,  $\beta$  represents the relative positioning of two products.



**Figure 3.3 Product Space (left) and Consumer Space (right)**

Consumers choose either open source software or proprietary software in order to maximize their utility as defined by Eq. (9). The marginal consumers who are indifferent between choosing either product can be defined by the indifference curve as below.

$$\theta_2 = \frac{x_{ps} - x_{os}}{y_{os} - y_{ps}} \theta_1 - \frac{P}{y_{os} - y_{ps}} \quad (10)$$

Note that the slope of the indifference curve is  $tg\beta$  defined in the product space. The indifference curve divides the consumer population into two parts: consumers above the indifference line choose open source software and consumers below the indifference line choose proprietary software. In other words, the demand of commercial software is determined by the area below the indifference line in the consumer space (unit square), denoted by  $D_{ps}$ .

The commercial software firm has to incur a cost to deliver a certain level of quality (usability and functionality). We assume the development cost is quadratic in software quality. Put in math, the total cost to reach the product location  $(x_{ps}, y_{ps})$  is  $c_1x_{ps}^2 + c_2y_{ps}^2$ , where  $c_1$  is the cost parameter of designing user interface and  $c_2$  is the cost parameter of developing functions. In summary, the profit function of a proprietary software firm is defined as:

$$\pi_{ps} = D_{ps} \cdot P - (c_1x_{ps}^2 + c_2y_{ps}^2) \quad (11)$$

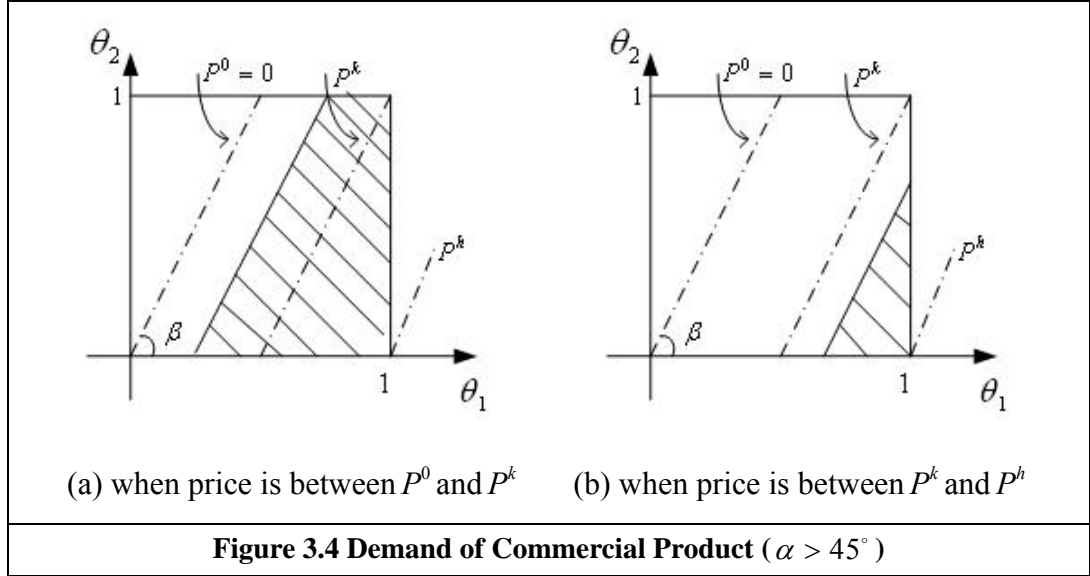
Due to the long development cycle and high uncertainties associated with software development, firms usually develop the software before setting the price. Therefore, we use backward induction to find the sub-game perfect equilibrium. That is, we first derive the optimal price given the location (design) of the product and then solve for the optimal location of the commercial software.

### 3.4.2 Optimal Pricing of Commercial Software

Due to different expressions of the demand function ( $D_{ps}$ ), we need to study the firm's optimal pricing strategy in the following two cases:  $\beta > 45^\circ$  and  $\beta \leq 45^\circ$ .

#### 3.4.2.1 Case 1. $\beta > 45^\circ$

In this case, the difference in usability is greater than that of functionality, i.e.,  $x_{ps} - x_{os} > y_{os} - y_{ps}$  (see Figure 3.4). We define  $P^0$  as the price when the indifference curve passes through point (0, 0) and  $P^h$  is the price when the indifference curve passes through point (1, 0). By definition,  $P^0 = 0$  and  $P^h = x_{ps} - x_{os}$ . Note that  $P^h$  is the highest price the commercial firm can charge. If the price is larger than  $P^h$ , all the consumers will choose open source software. Let  $P^k$  be the price when the indifference curve passes through point (1,0), i.e.,  $P^k = (x_{ps} - x_{os}) - (y_{os} - y_{ps})$ . If the indifference curve falls between the line defined by  $P^0$  and  $P^k$ , the demand of the commercial software has the shape of a trapezoid. If the indifference curve falls between the line defined by  $P^k$  and  $P^h$ , the demand of the commercial software has the shape of a triangle. These two scenarios are depicted in (a) and (b) of Figure 3.4 respectively.



### Scenario 1

When the price of the commercial software is between  $P^0$  and  $P^k$ , the demand for the proprietary software is the sum of area of a triangle and a parallelogram (see Figure 3.4

(a)). That is, the demand ( $D_{ps}$ ) is calculated as follows:

$$D_{ps} = 1 - \frac{1}{2} \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} - \frac{P}{x_{ps} - x_{os}} \quad (12)$$

The profit maximization of the commercial firm is defined as:

$$\begin{aligned} \text{Max}_P \quad \pi_{ps} &= \left(1 - \frac{1}{2} \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} - \frac{P}{x_{ps} - x_{os}}\right) \cdot P - (c_1 x_{ps}^2 + c_2 y_{ps}^2) \\ \text{s.t.} \quad &0 \leq P < P^k \end{aligned} \quad (13)$$

We derive the optimal price for the commercial software as required by first order condition, described in Eq. (14):

$$P^* = \frac{2(x_{ps} - x_{os}) - (y_{os} - y_{ps})}{4} \quad (14)$$

Note that the optimal price ( $P^*$ ) is in the interval  $(0, P^k)$  when  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ .

## Scenario 2

When price is in the interval  $[P^k, P^h]$ , the demand for proprietary software is the area of a triangle as follows:

$$D_{ps} = \frac{1}{2} \frac{(x_{ps} - x_{os} - P)^2}{(x_{ps} - x_{os})(y_{os} - y_{ps})} \quad (15)$$

We solve for the profit maximization problem for the commercial firm to get the optimal price in this scenario, described in Eq. (16).

$$P^* = \frac{x_{ps} - x_{os}}{3} \quad (16)$$

Note that the optimal price ( $P^*$ ) is in the interval  $[P^k, P^h]$  under the condition

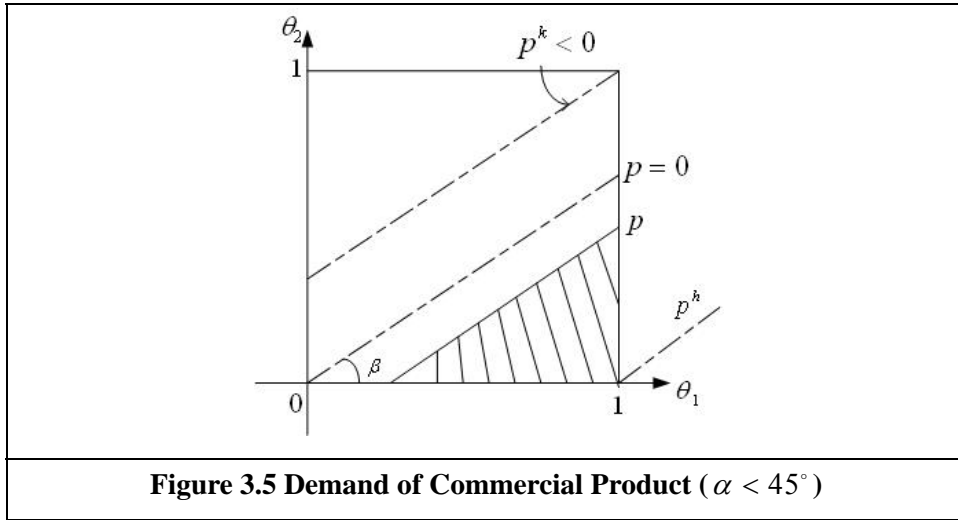
$$\frac{2}{3} \leq \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < 1.$$

### 3.4.2.2 Case 2. $\beta \leq 45^\circ$

In this case, the difference in functionality is greater than that of usability, i.e.,

$x_{ps} - x_{os} \leq y_{os} - y_{ps}$  (see Figure 3.5).  $P^h = x_{os} - x_{ps}$  is still the highest price that

commercial software firm can set. Since  $P^k < 0$  in this case, the equilibrium price of the commercial software must be set in the interval  $(0, P^h)$ . Therefore, the demand for the commercial software is the area of shaded triangle (see Figure 3.5). The demand is defined as:



$$D_{ps} = \frac{1}{2} \frac{(x_{ps} - x_{os} - P)^2}{(x_{ps} - x_{os})(y_{os} - y_{ps})} \quad (17)$$

We solve for the profit maximization problem for the commercial firm to get the optimal price in this case, described in Eq. (18).

$$P^* = \frac{x_{ps} - x_{os}}{3} \quad (18)$$

Note that the optimal price ( $P^*$ ) is in the interval  $(0, P^h)$  under the condition



$$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} \geq 1.$$

The optimal price, demand and profit of commercial software firm under the different conditions are summarized in Table 3.1. Note that the conditions are determined by the relative positioning of the two products (open source and proprietary software), or  $tg \beta$  defined in the product space.

<b>Table 3.1 The Optimal Price, Demand and Profit of Commercial Firm</b>		
	$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$	$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} \geq \frac{2}{3}$
Profit	$\pi_1^* = \frac{(x_{ps} - x_{os}) - (y_{os} - y_{ps})}{4} + \frac{1}{16} \frac{(y_{os} - y_{ps})^2}{x_{ps} - x_{os}} - (c_1 x_{ps}^2 + c_2 y_{ps}^2)$	$\pi_2^* = \frac{2}{27} \frac{(x_{ps} - x_{os})^2}{y_{os} - y_{ps}} - (c_1 x_1^2 + c_2 y_1^2)$
Price	$P_1^* = \frac{x_{ps} - x_{os}}{2} - \frac{y_{os} - y_{ps}}{4}$	$P_2^* = \frac{x_{ps} - x_{os}}{3}$
Demand	$D_{ps}^{1*} = \frac{1}{2} - \frac{1}{4} \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}}$	$D_{ps}^{2*} = \frac{2}{9} \frac{x_{ps} - x_{os}}{y_{os} - y_{ps}}$

### 3.4.3 Optimal Design of Commercial Software

Since the price equilibria are determined by the product positions, we can directly incorporate the price equilibria into product positions analysis. To find the optimal positions, we solve these constrained optimization problems by using Kuhn-Tucker conditions (see proof in Appendix). As a result, there are two possible solutions under the certain conditions: one is interior; the other is on the boundary.

There is an interior solution under the conditions:  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$  and

$3x_{ps} - x_{os} > \frac{4c_2 + c_1}{16c_1c_2}$ . It is too complex to express the solution in the specific equations.

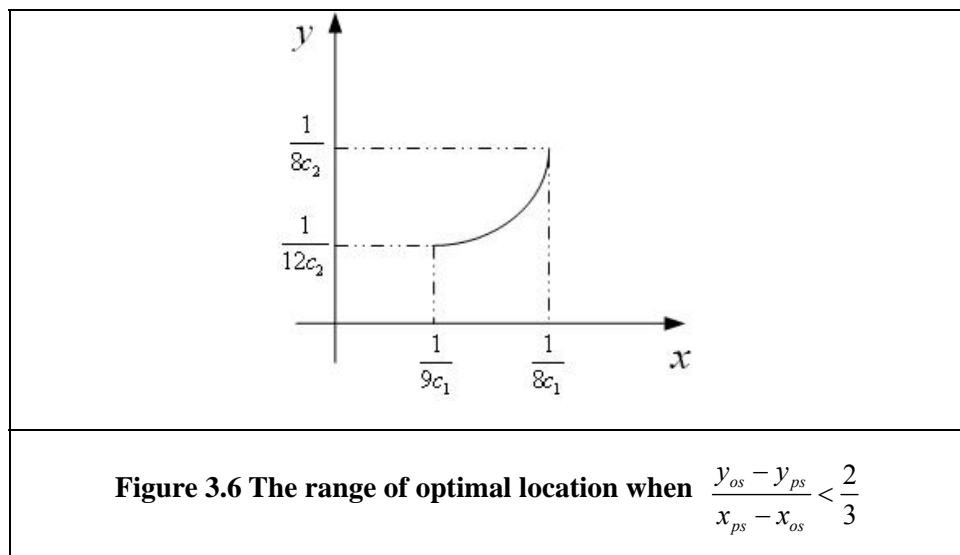
But we can confirm that if it is valid<sup>10</sup>, the optimal usability ( $x_1^*$ ) and functionality ( $y_1^*$ ) of commercial software are on the curve (shown in Figure 3.6):

$$4 - 32x_{ps}c_1 = (2 - 16y_{ps}c_2)^2$$

In which  $x_{ps}, y_{ps}$  are in the following ranges:

$$\frac{1}{9c_1} < x_{ps} < \frac{1}{8c_1}$$

$$\frac{1}{12c_2} < y_{ps} < \frac{1}{8c_2}$$

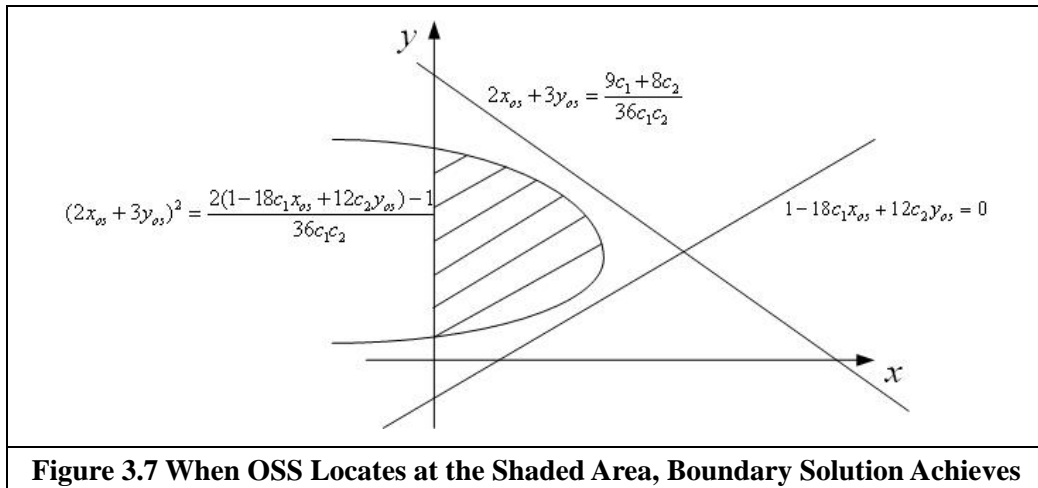


<sup>10</sup> There is the positive profit for the commercial firm.

The boundary solution is achieved under the conditions:

$$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} = \frac{2}{3} \text{ and } (2x_{os} + 3y_{os})^2 < \frac{2(1 - 18c_1x_{os} + 12c_2y_{os}) - 1}{36c_1c_2}.$$

It implies that only when usability and functionality of open source software satisfy the condition above (see Figure 3.7), the boundary solution exists. Table 3.2 shows the optimal solutions (price and product design) of the commercial software firm.



**Figure 3.7** When OSS Locates at the Shaded Area, Boundary Solution Achieves

<b>Table 3.2 Boundary Solution of the Software Firm</b>	
Profit	$\pi^* = \frac{1 + 24c_2y_{os} - 36c_1(x_{os} + 4c_2x_{os}^2 + 12c_2x_{os}y_{os} + 9c_2y_{os}^2)}{36(9c_1 + 4c_2)}$
Price	$P^* = \frac{1 - 18c_1x_{os} + 12c_2y_{os}}{54c_1 + 24c_2}$
Product Design	$x_{ps}^* = \frac{8c_2x_{os} + 12c_2y_{os} + 1}{18c_1 + 8c_2} \quad y_{ps}^* = \frac{18c_1x_{os} + 27c_1y_{os} - 1}{3(9c_1 + 4c_2)}$

When  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ , the interior solution achieves and the market share of commercial

software is larger than one third; when  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} = \frac{2}{3}$ , boundary solution reaches and the

market share of commercial software is just one third; when  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} > \frac{2}{3}$ , the market

share of commercial software is less than one third and there no valid optimal product design because of negative profit. It implies that if the open source has sufficient better advantage in functionality relative to its disadvantage in user interface, it will attract most customers in the market (2/3 of the market) and commercial software will be washed out of the market since commercial firm cannot make profit.

*Proposition 5.* If the open source has sufficient better advantage in functionality relative to its disadvantage in user interface, the proprietary software will be driven out of the market.

The open source has sufficient better advantage over functionality relative to its disadvantage in user interface, that is, the case  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} > \frac{2}{3}$  in Table 3.1. When

commercial software firm implements optimal strategy in this case, the firm could not make positive profit (detail proof in Appendix). We also find that in this case the market share of commercial software is less than one third and in the previous cases

$(\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} = \frac{2}{3}, \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3})$  the market share is equal or larger than one third. In the

other words, if market share of commercial software is very low, the commercial firm

will give up this market.

### **3.4.4 Comparative Static Analysis**

We summarize the comparative statistics in Table 3.3.

#### **3.4.4.1 Relationship between product design and development cost**

*(Boundary Solution)* For a commercial software firm, higher cost in designing usability will induce lower usability; higher cost in designing functionality will result in lower functionality. It is reasonable that high cost induces low “quality”. However, higher cost in designing usability will encourage better functionality and higher cost in designing functionality may also encourage better usability. It implies that if the cost on one character is very high, the effort on the other character will increase, that is, the “quality” of the other character increases.

*(Interior Solution)* For the proprietary software, higher cost in designing functions or usability will induce both lower usability and lower functionality, i.e., high cost induces low “quality”. It is an obvious result that the costs are the tradeoff of a “good” product.

<b>Table 3.3 Comparative Statistics of Optimal Solutions</b>		
	Interior Solution	Boundary Solution
Conditions	$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ $3x_{ps} - x_{os} > \frac{4c_2 + c_1}{16c_1c_2}$	$\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} = \frac{2}{3}$ $(2x_{os} + 3y_{os})^2 < \frac{2(1 - 18c_1x_{os} + 12c_2y_{os}) - 1}{36c_1c_2}$
Effects of cost on software design	$\frac{dx_{ps}}{dc_1} < 0, \frac{dy_{ps}}{dc_1} < 0$ $\frac{dx_{ps}}{dc_2} < 0, \frac{dy_{ps}}{dc_2} < 0$	$\frac{dx_{ps}}{dc_1} < 0, \frac{dy_{ps}}{dc_1} > 0, \frac{dy_{ps}}{dc_2} < 0$ $\frac{dx_{ps}}{dc_2} < 0 \text{ if } 2x_{os} + 3y_{os} < \frac{1}{9c_1}$ $\frac{dx_{ps}}{dc_2} > 0 \text{ if } 2x_{os} + 3y_{os} > \frac{1}{9c_1}$
Effects of OSS positions on software design	$\frac{dx_{ps}}{dx_{os}} < 0, \frac{dy_{ps}}{dx_{os}} < 0$ $\frac{dx_{ps}}{dy_{os}} < 0, \frac{dy_{ps}}{dy_{os}} < 0$	$\frac{dx_{ps}}{dx_{os}} > 0, \frac{dy_{ps}}{dx_{os}} > 0$ $\frac{dx_{ps}}{dy_{os}} > 0, \frac{dy_{ps}}{dy_{os}} > 0$
Effects of OSS positions on price setting	$\frac{dP}{dx_{os}} > 0, \frac{dP}{dy_{os}} > 0$	$\frac{dP}{dx_{os}} < 0, \frac{dP}{dy_{os}} > 0$
Effects of OSS positions on profit	$\frac{d\pi}{dx_{os}} < 0, \frac{d\pi}{dy_{os}} < 0$	$\frac{d\pi}{dx_{os}} < 0$ $\frac{d\pi}{dy_{os}} > 0 \text{ if } 2x_{os} + 3y_{os} < \frac{1}{9c_1}$ $\frac{d\pi}{dy_{os}} < 0 \text{ if } 2x_{os} + 3y_{os} > \frac{1}{9c_1}$

### 3.4.4.2 Relationship between product design and open source position

*(Boundary Solution)* The more usability and functions provided by open source software, the more usability and functions commercial software needs to design. In the other words, when two products compete with each other, if one product provides better “quality”, the other should provide better “quality” too. Since in this case, market share of commercial software is one third. In order to maintain this demand, if open source

provides better product, commercial should also provides better product to attract customers.

*(Interior Solution)* Opposite to boundary solution, if open source is with higher usability or functionality, commercial software will provide lower usability or functionality instead.

#### **3.4.4.3 Relationship between price and open source position**

*(Boundary Solution)* After designing the product, commercial software firm need to set its price. If open source software provides better interface, the price of commercial software should be lower; if open source software provides more functions, the price should be higher. Since proprietary software has the relative advantage in usability, if open source is with better usability, this advantage will be weakened. In order to compete with open source, lowering price is a reasonable way to catch the users. On the other side, open source has the relative dominance in functionality. Since open source has owned high functionality but low usability, improving functions will push some average users to proprietary software. At this time, increasing price can bring commercial software more profit. There are still some naive users who will buy commercial software with higher price because they have not enough ability to use open source software which is with not friendly user interface.

*(Interior Solution)* If open source provides better interface or better functionality,

commercial software firm should set higher price to compete with open source. This strategy is to give up some users but to earn more from the naive users. Although the demand is lower, but commercial firm still make profit.

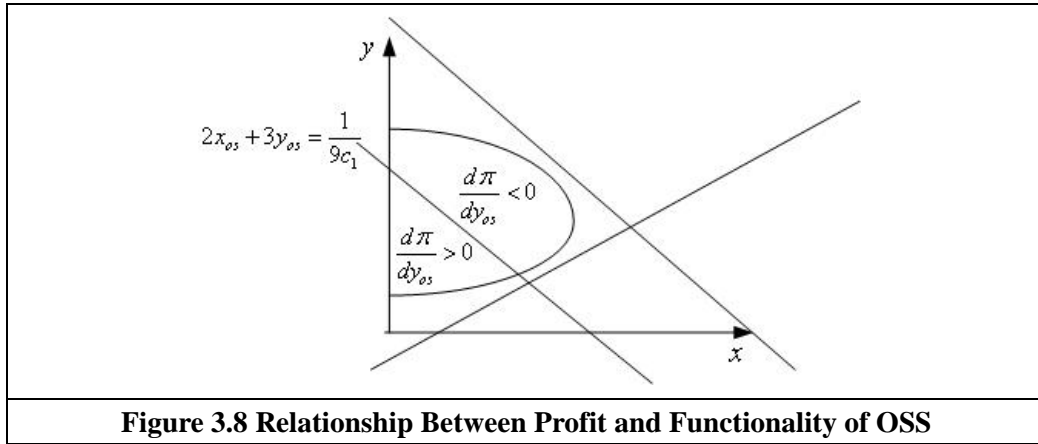
#### **3.4.4.4 Relationship between profit and open source position**

*(Boundary Solution)* Profit is the most concern for the commercial software. Whatever the product designing and price setting, all are for maximizing profit. The level of usability and functionality of open source software does affect the profit of commercial firm. From our analysis, if open source provides higher usability, the profit of commercial firm will be lower; if open source provides higher functionality, the profit has both possibilities of higher or lower which depends on characteristics of open source product (see Figure 3.8). Through this result, it is obvious that better usability of open source software brings a bigger threat to commercial software. It just identifies that the great fear of commercial software firms is that somebody will come behind them and make products that not only are more functions but are also easy to use, fun, and make people more productive. In fact, however, open source developers do not have more intention to improve the usability. Therefore, as mentioned by Lerner and Tirole (2002), open source project requires a credible leader who must provide a ‘vision’.

*(Interior Solution)* In this case, if open source provides better usability or better functionality, the profit of commercial firm will be lower. That is, the better “quality” of



open source software, the less profit a commercial firm can earn.



**Figure 3.8 Relationship Between Profit and Functionality of OSS**

### 3.4.5 Welfare Analysis

Since the specific product characteristics are difficult to drive when there are interior solutions for the software firm, welfare analysis only focuses on that of boundary solutions. We investigate the welfare properties of this equilibrium. When commercial firm obtains optimal profit, the welfare that all users get (consumer surplus) are captured as (proved in Appendix):

$$CS = \frac{2 + 207c_1x_{os} + 108c_2x_{os} + 243c_1y_{os} + 132c_2y_{os}}{486c_1 + 216c_2}$$

From the above equation, better usability or functionality of open source software will bring more consumer surplus. Since open source is free of usage, if it can provide better product and attract more people to use it, consumer surplus would increase. It implies

that consumers are always better off when the open source software provides better usability or functionality.

### **3.4.6 Overall Analysis**

*(Boundary Solution)* The appearance of open source does affect the commercial software firm. In the equilibrium, the price, profit and product characteristics of commercial software firm are determined by the costs of designing product and the characteristics of open source software. If open source provides better user interface, commercial software firm needs to provide better usability and functionality with lower price to maintain the demand. As a result, the profit is cut down. If open source provides better functionality, commercial software firm needs to provide better usability and functionality with higher price. Then, the profit will be either higher or lower which is dependent on the current characteristics of open source software. Furthermore, better usability or functionality by open source will bring higher surplus for consumers. It is a good situation for users that open source provides better usability and functionality.

*(Interior Solution)* For open source software, the advantage over functionality relative to its disadvantage in user interface is not obvious in this case. When open source provides higher usability or functionality, the advantage of open source is more obvious. In order to make profit, commercial firm have to provide less usability (but still higher

than open source usability) and functionality with higher price. Through giving up some consumers but increasing price, commercial firm still can make profit. When open source is strong enough, commercial firm could not compete with it using this interior strategy, and should change to boundary strategy.

### **3.5 Concluding Remarks**

Open source software has been gaining popularity among individuals and organizations as a "free" alternative to traditional proprietary software. No doubt, the commercial firm's profit is affected by the presence of open source software. Past study on the comparison between open source and proprietary software suggests that open source software offers better functionality while proprietary software has advantage in ease of use and reliable customer support. Consequently, open source and proprietary software are normally preferred by different groups of consumers with distinct requirement (tastes). In the first model, we employ a Hotelling model to study the optimal strategies of commercial firms when consumers have different tastes. In particular, we seek to solve for the optimal design and pricing of proprietary software. Further, we analyze the impact of network externality on the optimal strategy and profit of the commercial firm. In the second model, extend the tastes of consumers to two dimensions: usability and functionality. We employ a two-dimensional vertical differentiation model in which consumers choose open source or commercial software on base of usability and functionality of the product. Given the characteristics of the open source software, a

monopoly commercial software provider seeks the optimal design and price to maximize its profit.

From the first model, the analysis suggests that the profit of the commercial firm is dependent on the fit cost and the positioning of open source software. The profit of the commercial firm is maximized when the open source software targets more specialized users. Furthermore, the market share of the commercial firm is largest when the open source software caters to users with extreme requirements, and the commercial firm should set a low price to capture a proportion of the market. In addition, we find that the social welfare is maximized when open source software targets more specialized users. Finally, our analysis suggests that the commercial firm gains more profit when there's positive network externalities.

From the second model, we find that the optimal strategies are not unique for a commercial firm. Different characteristics of open source product will lead to different strategies. In some circumstance, if open source provides better user interface, commercial software firm needs to provide better usability and functionality with lower price to maintain the demand. As a result, the profit is cut down. If open source provides better functionality, the profit of commercial firm will be either higher or lower which is dependent on the current characteristics of open source software. Furthermore, consumers are always better off when the open source software provides better usability or functionality. It is a good situation for users that open source provides

better usability and functionality. In the other circumstance, when open source provides higher usability or functionality, commercial firm have to provide less usability (but still higher than open source usability) and functionality with higher price. Through giving up some consumers but increasing price, commercial firm still can make profit. When open source is more and more strong, commercial firm could not compete with it using this interior strategy, and should change to boundary strategy. At the end, when the open source has sufficient better advantage over functionality relative to its disadvantage in user interface, the commercial firm will be driven out of the market.

Our research has limitations that can be extended in the following directions. For model 1, first, we shall analyze whether the commercial firm should reach out to attract open source users to compete. That is, whether all customers located between open source and proprietary software should be served. Secondly, we shall explore the possibility of different network intensity for open source and proprietary software. Finally, we shall provide answers to optimal strategies for the more complicate situation when the fit cost is high ( $t > v$ ). For model 2, we will investigate a two-period model with network externality. In considering network externality, we will study the impact of new release (upgrade) of open source software. Upgrading (or new release) is common in the software industry. In general, the upgrading of software involves adding more features, improving the user interface, and fixing bugs. Bug fixing is a necessary task for each upgrade to sustain customer satisfaction. Thus, it's of no strategic interest to study whether and how firms should fix their product. On the other hand, the improvement of

usability and functionality might have a strategic impact on firm's profitability in the presence of the competing open source software. Provided by a new release of the open source software (which has better user interface and/or functionality), the commercial software users may choose to continue using the commercial software or switch to the open source software if they get more utility from the open source software; new consumers may enter the market and select either the open source software or the commercial software. Therefore, the commercial firm has to optimally upgrade its software and set up a new price in order to maximize profit.

## References

Bessen J. E. "Open source software: free provision of complex public goods," Working paper, 2005, Available at SSRN: <http://ssrn.com/abstract=588763>.

Bonaccorsi A. and Rossi C. "Why open source software can succeed," *Research Policy*, 32, 2003, pp. 1243-1258.

Casadesus-Masanell and Ghemawat "Dynamic mixed duopoly: a model motivated by Linux vs. Windows," Working Paper, 2003, Graduate School of Business Administration, Harvard University.

Galli P. "Open source code finds way into Microsoft product," eWeek.com article, September, 2005.

Gaudeul A. "The (LA)TEX project: a case study of open source software," Research paper, 2004, Universities of Toulouse and Southampton.

Gaudeul A. "Competition between open-source and proprietary Software: the (LA)TEX case study," Unpublished working paper, 2004, Universities of Toulouse and Southampton.

Roberts J., Hann I., and Slaughter S. "Understanding the motivations, participation and performance of open source software developers: a longitudinal study of the Apache projects," Unpublished working paper, 2004, Carnegie-Mellon University.

Johnson J. P. "Collaboration, peer review and open source software," Unpublished working paper, 2004, Cornell University.

Kuan J. "Open source software as consumer integration into production", Unpublished working paper, 2001, Stanford University.

Lakhani R. K. and Wolf G. R. "Why hackers do what they do: understanding motivation and effort in free/open source software projects," Working Paper 4425-03, 2003, MIT Sloan School of Management.

Lerner J. and Tirole J. "Some simple economics of open source," *Journal of Industrial Economics*, 52(2), 2002, pp. 197-234.

Massel D. "The changing face of open source," LinuxInsider.com article, October, 2005.

Nichols M. D. and Twidale M. B. "Usability and open source software," *First Monday*, 8(1), 2003.

Schmidt K. and Schnitzer M. "Public subsidies for open source? Some economic policy issues of the software market," CEPR Discussion Paper, 2002.

Shell S. "Open source vs. commercial software: why proprietary software is here to stay," Informit.com article, October, 2005.



Wheeler D. A. "Why open source software/free software (OSS/FS)? Look at the number!" Online resource: [http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html), December, 2003.

## Appendix

We analyze a sequential game in which commercial software firm first choose the product characteristics and then choose the price when given the characteristics of open source software. To find the sub-game perfect equilibrium, we use backwards induction method. The price equilibrium has been analyzed. Then, the task is to find optimal product design. We solve these constrained optimization problems by using Kuhn-Tucker Theory.

When  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ , the problem is to solve

$$\begin{aligned} \mathit{Max}_{x_{ps}, y_{ps}} \pi_{ps} &= \left(1 - \frac{1}{2} \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} - \frac{P}{x_{ps} - x_{os}}\right) \cdot P - (c_1 x_{ps}^2 + c_2 y_{ps}^2) \\ \text{s.t.} \quad \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} &< \frac{2}{3} \end{aligned}$$

Define Lagrangian Function as:

$$L_1 = \frac{(x_{ps} - x_{os}) - (y_{os} - y_{ps})}{4} + \frac{1}{16} \frac{(y_{os} - y_{ps})^2}{x_{ps} - x_{os}} - (c_1 x_{ps}^2 + c_2 y_{ps}^2) - \lambda \left( \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} - \frac{2}{3} \right) \quad (1)$$

When  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} \geq \frac{2}{3}$ , the problem is to solve

$$\begin{aligned} \mathit{max}_{x_{ps}, y_{ps}} \pi_{ps} &= \frac{2}{27} \frac{(x_{ps} - x_{os})^2}{y_{os} - y_{ps}} - (c_1 x_{ps}^2 + c_2 y_{ps}^2) \\ \text{s.t.} \quad \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} &\geq \frac{2}{3} \end{aligned}$$

Define Lagrangian Function as:

$$L_2 = \frac{2}{27} \frac{(x_{ps} - x_{os})^2}{y_{os} - y_{ps}} - (c_1 x_{ps}^2 + c_2 y_{ps}^2) - \mu \left( \frac{2}{3} \frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} \right) \quad (2)$$

Solve the following equations:

$$\frac{\partial L_1}{\partial x_{ps}} = 0, \quad \frac{\partial L_1}{\partial y_{ps}} = 0, \quad \lambda \frac{\partial L_1}{\partial \lambda} = 0 \quad \lambda \geq 0, \quad \frac{\partial L_1}{\partial \lambda} \geq 0$$

$$\frac{\partial L_2}{\partial x_{ps}} = 0, \quad \frac{\partial L_2}{\partial y_{ps}} = 0, \quad \mu \frac{\partial L_2}{\partial \mu} = 0 \quad \mu \geq 0, \quad \frac{\partial L_2}{\partial \mu} \geq 0$$

After simplifying,

$$\lambda = (1 - 18c_1x_{os} + 12c_2y_{os})[36c_1c_2(2x_{os} + 3y_{os}) - 9c_1 - 8c_2]$$

$$\mu = -(1 - 18c_1x_{os} + 12c_2y_{os})[36c_1c_2(2x_{os} + 3y_{os}) - 9c_1 - 8c_2]$$

$y_{ps}^* > 0, y_{os} - y_{ps}^* > 0, x_{ps}^* - x_{os} > 0, P^* > 0$  are assured by  $1 - 18c_1x_{os} + 12c_2y_{os} > 0$

To ensure  $\pi^* > 0$ ,  $(2x_{os} + 3y_{os})^2 < \frac{2(1 - 18c_1x_{os} + 12c_2y_{os}) - 1}{36c_1c_2}$

If  $36c_1c_2(2x_{os} + 3y_{os}) - 9c_1 - 8c_2 > 0$ , then  $\lambda > 0$

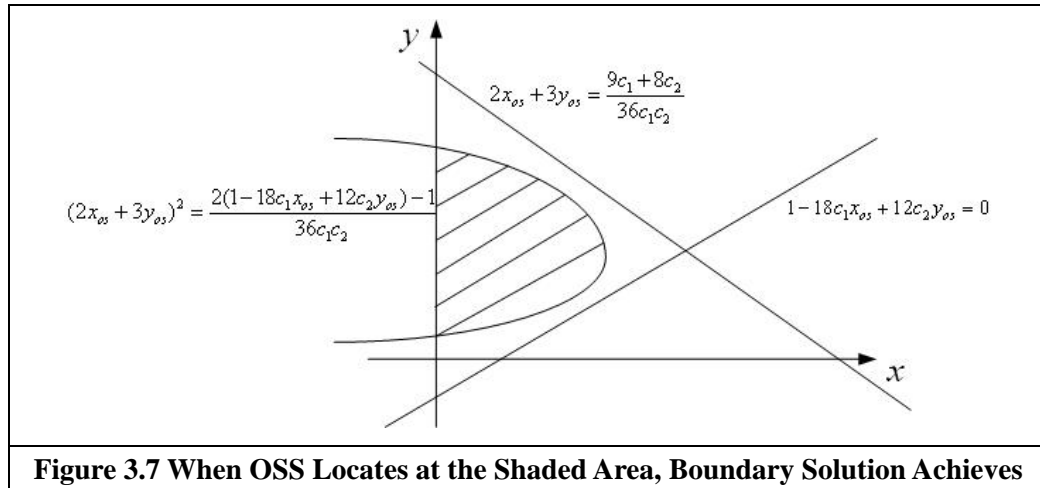
If  $36c_1c_2(2x_{os} + 3y_{os}) - 9c_1 - 8c_2 < 0$ , then  $\mu > 0$

For Case 1, the boundary solution is invalid because of negative profit. For Case 2, the

boundary solution is under the following conditions:

$$\begin{cases} 1 - 18c_1x_{os} + 12c_2y_{os} > 0 \\ 2x_{os} + 3y_{os} < \frac{9c_1 + 8c_2}{36c_1c_2} \\ (2x_{os} + 3y_{os})^2 < \frac{2(1 - 18c_1x_{os} + 12c_2y_{os}) - 1}{36c_1c_2} \end{cases}$$

We express the above conditions in the shaded area of Figure 3.7.



**Figure 3.7 When OSS Locates at the Shaded Area, Boundary Solution Achieves**

Therefore, under the condition that  $(2x_{os} + 3y_{os})^2 < \frac{2(1 - 18c_1x_{os} + 12c_2y_{os}) - 1}{36c_1c_2}$ , the

optimal solutions are:

$$x_{ps}^* = \frac{8c_2x_{os} + 12c_2y_{os} + 1}{18c_1 + 8c_2} \quad (3)$$

$$y_{ps}^* = \frac{18c_1x_{os} + 27c_1y_{os} - 1}{3(9c_1 + 4c_2)} \quad (4)$$

$$P^* = \frac{1 - 18c_1x_{os} + 12c_2y_{os}}{54c_1 + 24c_2} \quad (5)$$

$$\pi^* = \frac{1 + 24c_2y_{os} - 36c_1(x_{os} + 4c_2x_{os}^2 + 12c_2x_{os}y_{os} + 9c_2y_{os}^2)}{36(9c_1 + 4c_2)} \quad (6)$$

In the equilibrium, the consumer surplus can be solved as the following:

$$CS = W_{ps} + W_{os}$$

$$W_{ps} = \int_{\frac{1}{3}}^1 \int_0^{\frac{3}{2}\theta_1 - \frac{1}{2}} (\theta_1 x_{ps} + \theta_2 y_{ps} - P) d\theta_2 d\theta_1 \quad (6)$$

$$W_{os} = \int_0^1 \int_0^{\frac{2}{3}\theta_2 + \frac{1}{3}} (\theta_1 x_{os} + \theta_2 y_{os}) d\theta_1 d\theta_2 \quad (7)$$

Thus,

$$CS = \frac{2 + 207c_1x_{os} + 108c_2x_{os} + 243c_1y_{os} + 132c_2y_{os}}{486c_1 + 216c_2}$$

For both Case 1&2, there is an interior solution for each. Since they are too complex to express here, we just give some conditions to assure their ranges. Through maximization conditions:

$$\frac{\partial \pi}{\partial x_{ps}} = 0, \quad \frac{\partial \pi}{\partial y_{ps}} = 0, \quad \frac{\partial^2 \pi}{\partial x_{ps}^2} < 0, \quad \frac{\partial^2 \pi}{\partial y_{ps}^2} < 0, \quad \frac{\partial^2 \pi}{\partial x_{ps}^2} \cdot \frac{\partial^2 \pi}{\partial y_{ps}^2} > \left( \frac{\partial^2 \pi}{\partial x_{ps} \partial y_{ps}} \right)^2$$

and the conditions:

$$y_{ps}^* > 0, \quad y_{os} - y_{ps}^* > 0, \quad x_{ps}^* - x_{os} > 0, \quad P^* > 0,$$

we can find that:

1. When  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} < \frac{2}{3}$ , the condition  $3x_{ps} - x_{os} > \frac{4c_2 + c_1}{16c_1c_2}$  is needed and the optimal

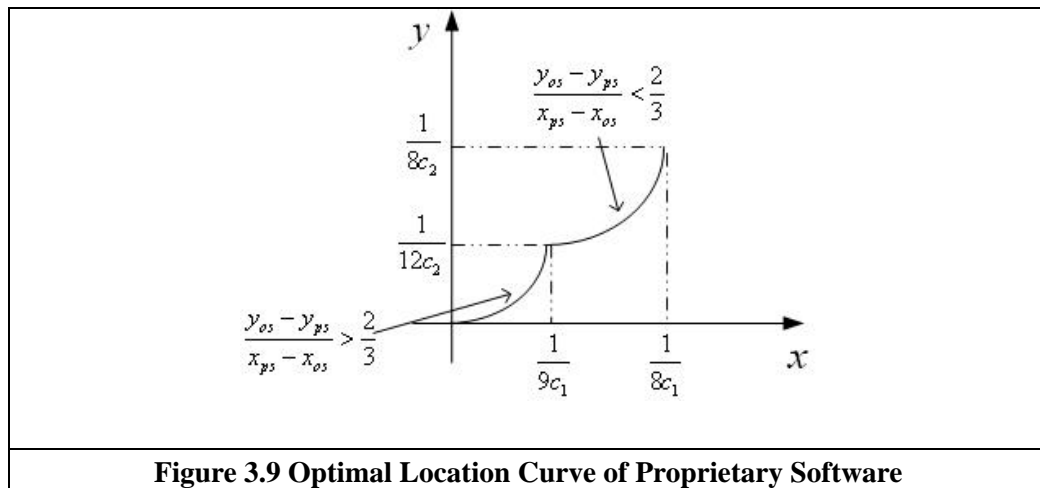
location of proprietary software should be on the part of the curve (see Figure 3.9):

$$4 - 32x_{ps}c_1 = (2 - 16y_{ps}c_2)^2$$

In which  $x_{ps}, y_{ps}$  are in the following ranges:

$$\frac{1}{9c_1} < x_{ps} < \frac{1}{8c_1}$$

$$\frac{1}{12c_2} < y_{ps} < \frac{1}{8c_2}$$



2. When  $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} > \frac{2}{3}$ , the condition  $y_{os} - 3y_{ps} > \frac{2}{27c_1}$  is needed and the optimal

location of commercial software should be on the part of the curve (see the above figure):

$$y_{ps} = \frac{27 c_1^2}{4 c_2} x_{ps}^2$$

In which  $x_{ps}, y_{ps}$  are in the following ranges:

$$0 < x_{ps} < \frac{1}{9c_1}$$

$$0 < y_{ps} < \frac{1}{12c_2}$$

However, for the Case 2,

$$\pi_{ps} = \frac{2}{27} \frac{(x_{ps} - x_{os})^2}{y_{os} - y_{ps}} - (c_1 x_{ps}^2 + c_2 y_{ps}^2) \quad (8)$$

$$\frac{\partial \pi}{\partial x_{ps}} = 0 \Rightarrow \frac{x_{ps} - x_{os}}{y_{os} - y_{ps}} = \frac{27}{2} c_1 x_{ps} \quad (9)$$

Put Eq. (9) into Eq. (8), we find

$$\pi_{ps} = -c_1 x_{ps} x_{os} - c_2 y_{ps}^2 < 0$$

Thus, if the open source has sufficient better advantage over functionality relative to its disadvantage in user interface ( $\frac{y_{os} - y_{ps}}{x_{ps} - x_{os}} > \frac{2}{3}$ ), the commercial firm will be driven out of the market.

## **Chapter 4. Partially Opening Source Code: A New Competitive Tool For Software Firms**

---

### **4.1 Introduction**

Microsoft, the once strongest opponent to open source movement, has recently decided to include open source code in its shipping of Windows Server 2003 Cluster Edition (Galli 2005). Sun Microsystems is actively involved in the development of open source based Java projects in a hope to promote Java-related products. Without a doubt, the profitability of a commercial firm is affected (if not threatened) when the consumers are offered with an alternative free option other than its proprietary software. Commercial software firms are now confronted with the critical decision on how to optimally compete with open source software. Some previous works try to seek answers of how the commercial software firms compete with open source community. However, for the software firms, it is far not enough to only consider threatens from open source projects. The competitions among the software firms are still aggressive. Although competition



between commercial firms is the traditional topic, yet one phenomenon makes this traditional problem shedding a new light.

Netscape opened source of their web browser to public for free, which was released as the Mozilla open source project. IBM offered the source code of Cloudscape, a Java-based relational database software, to the Apache Software Foundation. Zend Technologies Ltd., who offered proprietary Web applications and development tools, participated in the PHP open source community. Such actions were followed by a bunch of big firms like Sun who also opened part of their commercial code. More and more commercial firms come to realize that combining open source strategy can make more value for them in the aggressively competitive environment.

The world was shocked again. People cannot understand why these firms would give up their profit to support the open source movement. Some industry practitioners come to realize that mixing open source with commercial software is the great prospect for success with open source (Taft 2005). Adam Fitzgerald, director for developer solutions at BEA Systems Inc., of San Jose, California, said at the panel at the BEAWorld conference: “You need to start thinking about what an open-source solution can do for you and identify best practices and best-of-breed open-source technology. This notion of blending open source solutions is what we see customers already using.” “Combining the best open source software and the best commercial software will give you the best solution,” said by Zhongyuan Zheng, vice president for R&D at

Beijing-based Red Flag Software Co. Ltd., China's premier Linux vendor and maker of Red Flag Linux. In addition, among those papers discussing the competition between open source and proprietary software or competition among the for-profit software firms, few studies examined the open source as a software company's competing for-profit strategy. It turns out that there are very few papers discussing the situation when some software firms open part of their code for profit reasons to actively compete with other software firms.

Thus, in this study, instead of focusing on the competition between open source and proprietary software, we will study the competition between two profit-oriented firms, and analyze the model that when open source is as a software company's competing for-profit strategy. We seek to find insights from answers of these questions: Why does a for-profit firm open up its commercial product? How much should the firm open to achieve most profit? What is the best competition structure of the market when both firms choose their best competitive strategies? In addition, we consider the situations whether there is a competing open source product available in the software market or not. We try to see how the presence of open source affects the strategies of commercial firms.

## **4.2 Literature Review**

The open source phenomenon has captured attention by academic scholars in various

areas. As mentioned in the previous chapter (section 3.2), there are three streams of research in examining open source phenomenon. One stream of literature deals with the motivation of participating in open source projects (Lerner and Tirole 2002, Bonaccorsi and Rossi 2003, Schmidt and Schnitzer 2003, Lakhani and Wolf 2005, Roberts et al 2006, etc.). A second stream of research compares the quality of open source and proprietary software (Kuan 2001, Johnson 2004, Shell 2005, Massel 2005, Bessen 2005, Gaudeul 2004, etc.). A third stream of research focuses on the competition between open source and commercial software (Casadesus-Masanell and Ghemawat 2006, Gaudeul 2004, etc.).

Our research is related but different to the third stream of research. We study the competition between two commercial software firms, instead of competition between open source and commercial software product. Our model is different from other traditional competition model in that we investigate the hybrid business model of open source and proprietary software by using economic modeling approach. Based on survey of software firms, Bonaccorsi et al (2006) found that few firms choose a pure business model by offering open source solutions alone, and vast majority receive revenues from hybrid business model which attempts to combine the advantages of open source with proprietary software. We adopt vertical differentiation model to examine the competition between two profit-oriented firms, considering whether to adopt open source strategy.

### 4.3 The Model

There are two competing systems in the software market provided by two software firms. In order to differentiate with each other, both of them will consider opening part of their source codes. There are two cases to be analyzed in this section. (1) There is no competing pure open source system available in the market. Two firms will compete with each other in duopoly market. (2) There is the competing pure open source system in the market. In this case, two software firms not only consider the competition between each other, but also consider the competition with the open source software.

Consider two profit-seeking software firms A and B that provide competing systems. Firm A and B decide to compete with each other by opening the source codes of some parts of the systems. The proportion of open source is specified by  $\alpha_A$  and  $\alpha_B$  respectively, where  $\alpha_A$  or  $\alpha_B$  is between 0 and 1. They provide open source components ( $\alpha_A$  or  $\alpha_B$ ) publicly available for free; and sell the proprietary portion ( $1 - \alpha_A$ , or  $1 - \alpha_B$ ) at price  $P_A$  and  $P_B$  respectively. The end users need both the proprietary portion and open source component to deploy the entire system.

Let  $\theta$  denote consumer's naïveté, i.e.,  $\theta$  is uniformly distributed between 0 and 1, with 0 referring to the most naïve consumer, and 1 referring to the most skillful consumer.

Consumers choose one of the following options to maximize their own utility: (1) buy the partial product from firm A at price  $P_A$  and integrate it with open source component

provided by firm A; (2) buy the partial product from firm B at price  $P_B$  and integrate it with open source component provided by firm B; (3a) develop the part of the product by themselves and integrate it with open source component provided by the firm which opens more proportion, if there is no competing open source product available in the software market; or (3b) adopt the pure open source product if there is a competing open source product in the market.

Before defining user's utility function, we need to clarify the cost for users. In case when users choose to make use of open source component provided by the firms or the pure open source product, users incur the customization cost. The customization cost is determined by consumer's skillfulness ( $\theta$ ) and the degree of openness of the system ( $\alpha$ ). Users with higher computer skills incur lower cost of customization. Furthermore, more proportion of the open source component requires more effort to customize, since users have to go through more (complex) source codes to carry out configuration and customization at various points. Thus, we define customization cost as:

$$C(\alpha, \theta) = c \cdot \alpha \cdot (1 - \theta)$$

In case when the users choose to develop the rest of the software by themselves, they incur the development cost proportional to their skill and the amount of workload. Users with higher computer skills incur lower development cost. In addition, the more proportion of the open source component, the less development cost required, since the less proportions need to self-develop. Hence, we define development cost as:

$$D(\alpha, \theta) = d \cdot (1 - \alpha) \cdot (1 - \theta)$$

Although users endure customization cost and development cost for adopting or developing open source system, they also obtain additional benefit from open source. For example, end users have more flexibility of software usage. They can customize the system, which would be more suitable for their own requirements. If the system is open source, the user can put it on the open source host such as SourceForge.net. Other open source users or developers, if interested in the program, will improve the performance of the system by modifying the functions, fixing bugs, or providing support and so on. Therefore, we assume there is the additional benefit for users who adopt open source. Users with higher computer skills obtain more benefit from open source. At meanwhile, more proportion of the open source component brings more flexibility to customize and thereafter brings more benefit for users. Thus, we define the benefit from adopting open source as:

$$S(\alpha, \theta) = s \cdot \alpha \cdot \theta$$

#### **4.3.1 Case 1: Duopoly Market Dominated by Firm A and Firm B.**

Let's first discuss the case when there is no competing open source product available in the software market. Firm A and B dominate the market, but they need to consider there are some highly skillful users who prefer to open source products, and may self-develop the systems. According to the options for customers: (1) buy the partial

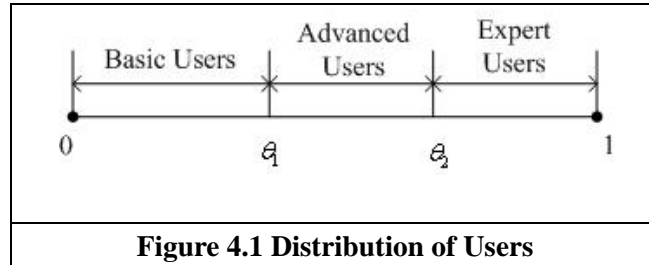
product from firm A at price  $P_A$  and integrate it with open source component provided by firm A; (2) buy the partial product from firm B at price  $P_B$  and integrate it with open source component provided by firm B; (3a) develop the part of the product by themselves and integrate it with open source component provided by the firm which opens more proportion, we define utility functions when customers choose different options as:

$$\begin{aligned}
 U_A &= v - P_A - C(\alpha_A, \theta) + S(\alpha_A, \theta) \\
 U_B &= v - P_B - C(\alpha_B, \theta) + S(\alpha_B, \theta) \\
 U_F &= v - C(\max[\alpha_A, \alpha_B], \theta) - D(\max[\alpha_A, \alpha_B], \theta) + S(1, \theta)
 \end{aligned}$$

where  $v$  is consumer's valuation;  $U_A$  is the consumer's utility function by adopting option (1);  $U_B$  is the utility function by adopting option (2);  $U_F$  is the utility function by adopting option (3a). We assume the customization cost ( $c$ ) is less than the development cost ( $d$ ), so that when there is open source code available, the users will not develop the same codes by themselves. The basic requirements for  $c$ ,  $d$ , and  $s$  are  $0 < c < d$  and  $s > 0$ . In the following discussions, in order to simplify the computation, we assume  $\alpha_A > \alpha_B$ , which is symmetric with the opposite side that  $\alpha_A < \alpha_B$ . In this chapter, we will not discuss the case  $\alpha_A = \alpha_B$ . When  $\alpha_A = \alpha_B$ , the research objective comes to the competition between open source and proprietary software, instead of competition between two for-profit software firms, which is the focus of this chapter. (We will investigate competition between open source and proprietary software in Chapter 3.)

According to the level of expertise  $\theta$  ( $0 \leq \theta \leq 1$ , small  $\theta$  for basic users) in using the open

source software, the consumers consist of three groups of persons: basic users, advanced users and expert users. The distribution of the users is shown in Figure 4.1.



By letting  $U_A=U_B$ , we get the marginal user ( $\theta_1$ ) who is indifferent with option (1) and option (2). Similarly, by letting  $U_A=U_F$ , we get the marginal user ( $\theta_2$ ) who is indifferent with option (2) and option (3a). Users located at  $(0, \theta_1)$  are basic users who will choose option (2), since they have much higher cost but less benefit for using open source software. These basic users tend to use the software system with less open source part (firm B's product). Users located at  $(\theta_1, \theta_2)$  are advanced users who will choose option (1). Users located at  $(\theta_2, 1)$  are expert users who will choose option (3a), since they can obtain more benefit from adopting open source but less cost for customization and development. We can solve the marginal user  $\theta_1$  and  $\theta_2$  as:

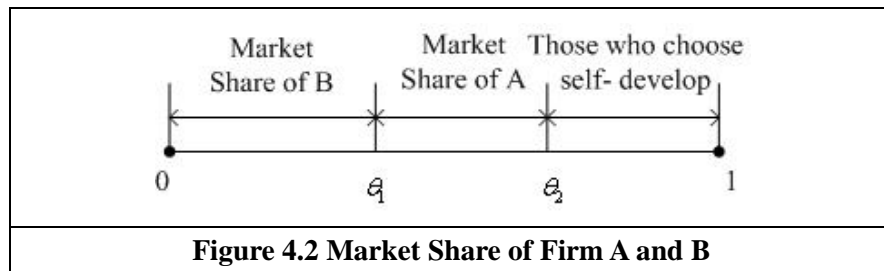
$$\theta_1 = \frac{(\alpha_A - \alpha_B)c + P_A - P_B}{(\alpha_A - \alpha_B)(s + c)}$$

$$\theta_2 = \frac{(1 - \alpha_A)d - P_A}{(1 - \alpha_A)(s + d)}$$

Firm A and B should optimally set the software price and degree of openness to maximize their profits. In order to focus on the competition between firm A and B, we



assume that no consumer between 0 and 1 is “stranded” by three options discussed above. In the following part we will analyze firm A and B’s optimal strategies to maximize their profits. The market size is normalized to 1, and thereby the market share of firm A and B are  $\theta_2 - \theta_1$  and  $\theta_1$ , respectively, which are shown in Figure 4.2. The



expert users who lie in  $(\theta_2, 1)$  will choose to self-develop part of the software and integrate it with open source component provided by firm A.

The profit functions for firm A and B turn out to be:

$$\begin{aligned}\pi_A &= (\theta_2 - \theta_1)P_A \\ \pi_B &= \theta_1 \cdot P_B\end{aligned}$$

In order to solve the profit maximization problem for firm A and B, we use the method of backward induction (Schotter 2001). As defined by Schotter (2001), backward induction is “the process of solving a game by going to its end and working backward, to figure out what each player will do along the way”. The firms first decide how much to open the source code, and then set the price of the produce. Thus, first, we solve maximization functions under the known degree of openness  $(\alpha_A, \alpha_B)$ ,

and get the pre-optimal price ( $P'_A, P'_B$ ) of each firm; second, we substitute price ( $P_A, P_B$ ) in the profit functions with pre-optimal price ( $P'_A, P'_B$ ), and solve optimal the degree of openness ( $\alpha^*_A, \alpha^*_B$ ) and then get ( $P^*_A, P^*_B$ ). The detailed procedures are shown in Appendix.

*Lemma 1.* In the duopoly market, which is dominated by firm A and B, given the degree of openness  $\alpha_A$  and  $\alpha_B$ , the pre-optimal price, demand and profit for firm A is  $P'_A, D'_A$ , and  $\pi'_A$ , respectively; the pre-optimal price, demand and profit for firm B is  $P'_B, D'_B$ , and  $\pi'_B$ , respectively. All these pre-optimal solutions are shown in Table 4.1.

<b>Table 4.1 Pre-optimal Strategy in Duopoly Market</b>		
Firm A	Profit	$\pi'_A = \frac{(1-\alpha_A)(\alpha_A-\alpha_B)(c(d-s)+2ds)^2[\alpha_A(c-d)+d+s-\alpha_B(c+s)]}{(s+c)(s+d)[-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(s+d)]^2}$
	Price	$P'_A = \frac{(1-\alpha_A)(\alpha_A-\alpha_B)[c(d-s)+2ds]}{-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(d+s)}$
	Demand	$D'_A = \frac{[c(d-s)+2ds][\alpha_A(c-d)+d+s-\alpha_B(c+s)]}{(c+s)[-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(d+s)]}$
Firm B	Profit	$\pi'_B = \frac{(\alpha_A-\alpha_B)[2cd+cs+ds-2\alpha_Bc(c+s)+\alpha_A(c-d)(2c+s)]^2}{(s+c)[-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(s+d)]^2}$
	Price	$P'_B = \frac{(\alpha_A-\alpha_B)[2cd+cs+ds-2\alpha_Bc(c+s)+\alpha_A(c-d)(2c+s)]}{-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(d+s)}$
	Demand	$D'_B = \frac{2cd+cs+ds-2\alpha_Bc(c+s)+\alpha_A(c-d)(2c+s)}{(c+s)[-4\alpha_B(c+s)+\alpha_A(4c-3d+s)+3(d+s)]}$

Firm A and B needs to optimally design the degree of openness ( $\alpha_A, \alpha_B$ ) to maximize their profits. Since the derivations are very laborious, the details are shown in Appendix. Through careful analysis and derivation, we find that the best strategy for

firm B is not to open any part of the source code ( $\alpha_B^* = 0$ ).

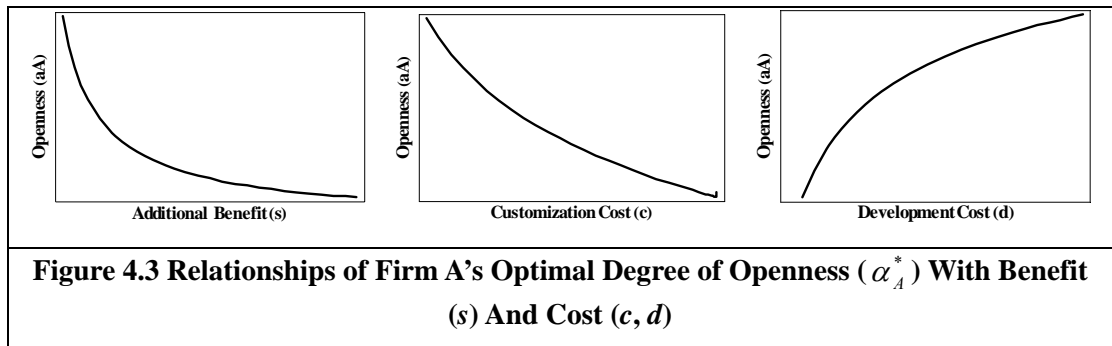
*Proposition 1.* In the duopoly market, one of the firms will choose not to open any part of the source code.

The firm, which chooses not to open any part of the source code, targets at the most basic users in the software market. This kind of users does not have any advanced computer skills, so that they do not have enough ability to customize the systems according to their own needs by themselves. Therefore, they do not have intention to use the products with the open source part. The firm knows this point! The firm will set a “good” price to get most profit and at the same time ensure these users to accept this price to buy its product. However, there are many advanced users, who care much more about flexibility. The other firm in the duopoly market will decide how much it will open the source code to attract more advanced users, and at the same time, consider the price of the product to ensure its own profit.

*Lemma 2.* In the duopoly market, which is dominated by firm A and B, under firm B's best choice:  $\alpha_B^* = 0$ , the pre-optimal price, demand and profit for firm A is  $P_A^*$ ,  $D_A^*$ , and  $\pi_A^*$ , respectively; the pre-optimal price, demand and profit for firm B is  $P_B^*$ ,  $D_B^*$ , and  $\pi_B^*$ , respectively. All the pre-optimal solutions are shown in Table 4.2.

Table 4.2 Pre-optimal Strategy in Duopoly Market		
Firm A	Profit	$\pi_A^* = \frac{(1-\alpha_A)\alpha_A[\alpha_A(c-d)+d+s](c(d-s)+2ds)^2}{(s+c)(s+d)[\alpha_A(4c-3d+s)+3(s+d)]^2}$
	Price	$P_A^* = \frac{(1-\alpha_A)\alpha_A[c(d-s)+2ds]}{\alpha_A(4c-3d+s)+3(d+s)}$
	Demand	$D_A^* = \frac{[c(d-s)+2ds][\alpha_A(c-d)+d+s]}{(c+s)(d+s)[\alpha_A(4c-3d+s)+3(d+s)]}$
Firm B	Profit	$\pi_B^* = \frac{\alpha_A[2cd+cs+ds+\alpha_A(c-d)(2c+s)]^2}{(s+c)[\alpha_A(4c-3d+s)+3(s+d)]^2}$
	Price	$P_B^* = \frac{\alpha_A[2cd+cs+ds+\alpha_A(c-d)(2c+s)]}{\alpha_A(4c-3d+s)+3(d+s)}$
	Demand	$D_B^* = \frac{2cd+cs+ds+\alpha_A(c-d)(2c+s)}{(c+s)[\alpha_A(4c-3d+s)+3(d+s)]}$

Although we have mathematically solved the closed forms of optimal degree of openness ( $\alpha_A^*$ ), which are shown in Appendix, yet it is a too complex formula expressed by  $s$ ,  $c$ , and  $d$ . Fortunately, we can clearly show the relationships of  $\alpha_A^*$  with  $s$ ,  $c$  and  $d$  (in Figure 4.3): (1) The more the additional benefit users can obtain, the fewer proportions firm A will open; (2) The more the customization cost, the fewer proportions firm A will open; (3) The more the development cost, the more proportions firm A will open.

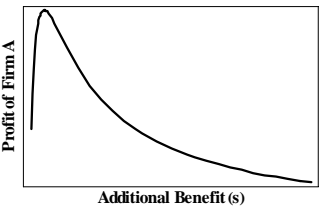
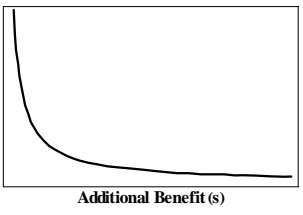
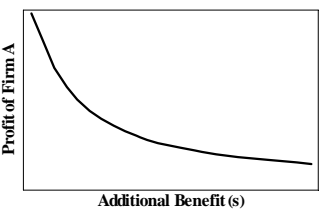
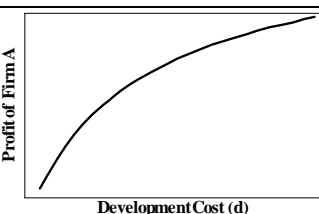
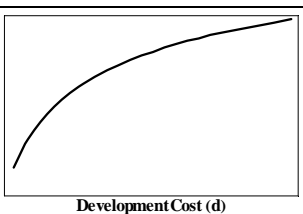
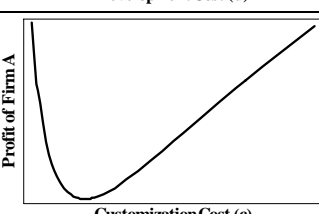
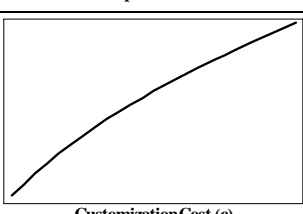
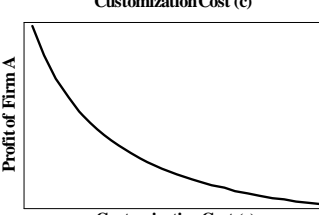


*Corollary 1.* The optimal degree of openness of firm A is positively affected by development cost, but negatively affected by additional benefit or customization cost.

In order to attract more advanced users, firm A provides a hybrid product of proprietary software product and open source product. Those advanced users do not have ability to develop the software system by themselves, but have strong requirements for flexibility. Thus, they choose the hybrid product provided by firm A. The optimal degree of openness of firm A is impacted by customization cost, development cost and additional benefit. First, the higher the development cost, the more difficulties the users meet in developing software by themselves. Hence, some of the advanced users will not develop the software by themselves. Thus, firm A would open more part of source code to attract more advanced users to let them give up self-developing the software. Second, the higher the customization cost, the more users would have more difficulties to customize the product. Although some of the users like flexibility, yet higher customization cost keep them away from the product with more open source part. Thus, firm A would open less part of source code to bring less trouble to customize product and at the same time to give the users some flexibility. Third, the higher the additional benefit from using open source, the more advanced users would like to develop part of the product by themselves. But for some basic users, this benefit would not affect them too much. Firm A would open less part of source code to attract more basic users but give up some advanced users.

Same as the optimal degree of openness of firm A, the optimal results including profit, price and demand for both firms are too complex formulas expressed by  $s$ ,  $c$ , and  $d$ . It shows that the nature or characteristics of product would affect the firms' decision making. For example, for a complex software system, if users want to customize it, they will incur high customization cost; obviously, if users want to develop a complex system, they must bear the high development cost. From graphic analysis, we find how these cost and benefit affect the optimal strategies of software firms. Corollary 2(a, b, c) present the relationships of firms' optimal profits with additional benefit and costs, which are graphically shown in Table 4.3. Corollary 3(a, b, c) present the relationships of firms' optimal price with additional benefit and costs, which are graphically shown in Table 4.4. Corollary 4(a, b, c) present the relationships of firms' optimal demand with additional benefit and costs, which are graphically shown in Table 4.5.

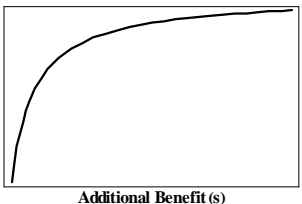
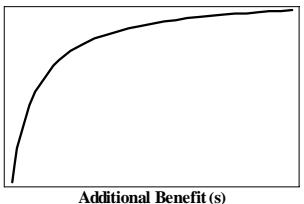
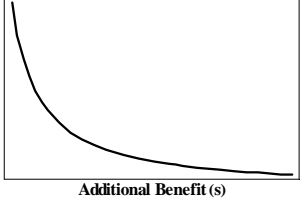
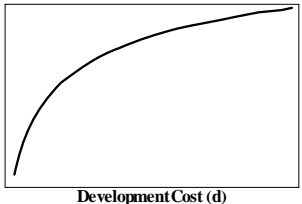
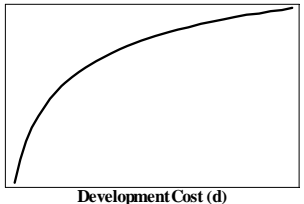
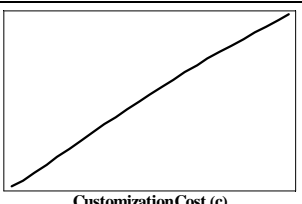
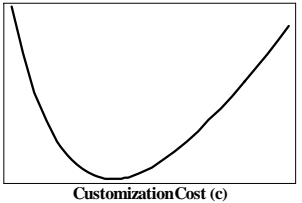
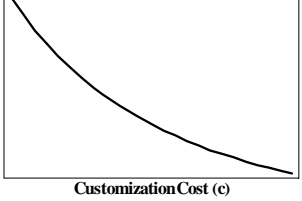
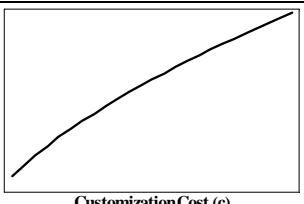
*Corollary 2a. (Profit and  $s$ )* When  $0 < c < \frac{3}{4}d$ , firm A's profit is increasing first and decreasing later, with the increase of  $s$ ; when  $\frac{3}{4}d \leq c < d$ , firm A's profit is monotonically decreasing with the increase of  $s$ . Firm B's profit is monotonically decreasing with the increase of  $s$ .

Table 4.3 Relationships of Optimal Profit With Benefit ( $s$ ) And Cost ( $c, d$ )		
	Profit of Firm A	Profit of Firm B
$s$	 $0 < c < \frac{3}{4}d$	
	 $\frac{3}{4}d \leq c < d$	
$d$		
$c$	 $0 < s < \frac{d}{5}$	
	 $s \geq \frac{d}{5}$	

*Corollary 2b. (Profit and  $d$ )* Firm A's profit is monotonically increasing with the increase of  $d$ . Firm B's profit is monotonically increasing with the increase of  $d$ .

*Corollary 2c. (Profit and  $c$ )* When  $0 < s < \frac{d}{5}$ , firm A's profit is decreasing first and increasing later, with the increase of  $c$ ; when  $s \geq \frac{d}{5}$ , firm A's profit is monotonically decreasing with the increase of  $c$ . Firm B's profit is monotonically increasing with the increase of  $c$ .

Corollary 3a. (Price and  $s$ ) Firm A's price is monotonically increasing with the increase of  $s$ . When  $0 < c \leq \frac{3}{4}d$ , firm B's price is monotonically increasing with the

<b>Table 4.4 Relationships of Optimal Price With Benefit (<math>s</math>) And Cost (<math>c, d</math>)</b>		
	Price of Firm A	Price of Firm B
$s$	 <p style="text-align: center;">Additional Benefit (<math>s</math>)</p>	 <p style="text-align: center;">Additional Benefit (<math>s</math>)</p> <p style="text-align: right;"><math>0 &lt; c \leq \frac{3}{4}d</math></p>  <p style="text-align: center;">Additional Benefit (<math>s</math>)</p> <p style="text-align: right;"><math>c &gt; \frac{3}{4}d</math></p>
$d$	 <p style="text-align: center;">Development Cost (<math>d</math>)</p>	 <p style="text-align: center;">Development Cost (<math>d</math>)</p>
$c$	 <p style="text-align: center;">Customization Cost (<math>c</math>)</p> <p style="text-align: right;"><math>0 &lt; s &lt; \frac{d}{3}</math></p>  <p style="text-align: center;">Customization Cost (<math>c</math>)</p> <p style="text-align: right;"><math>\frac{d}{3} \leq s \leq \frac{4}{9}d</math></p>  <p style="text-align: center;">Customization Cost (<math>c</math>)</p> <p style="text-align: right;"><math>s &gt; \frac{4}{9}d</math></p>	 <p style="text-align: center;">Customization Cost (<math>c</math>)</p>



increase of  $s$ ; when  $c > \frac{3}{4}d$ , firm B's price is monotonically decreasing with the increase of  $s$ .

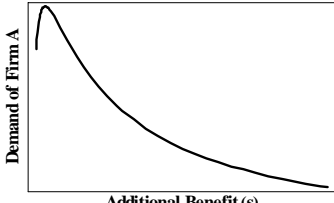
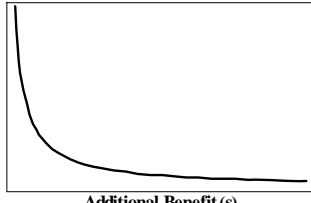
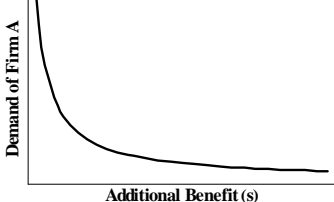
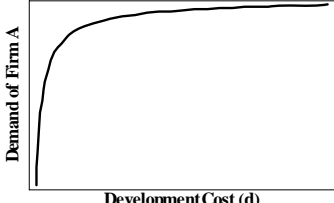
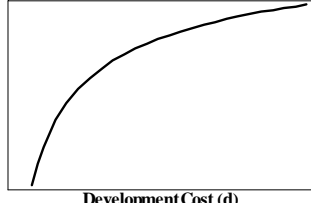
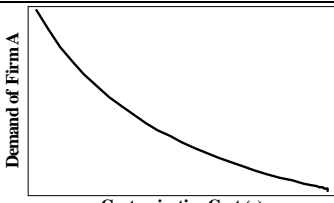
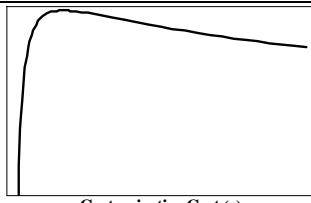
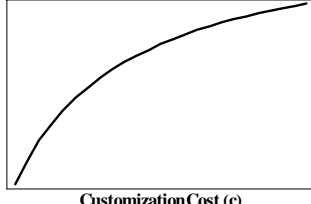
*Corollary 3b. (Price and  $d$ )* Firm A's price is monotonically increasing with the increase of  $d$ . Firm B's price is monotonically increasing with the increase of  $d$ .

*Corollary 3c. (Price and  $c$ )* When  $0 < s < \frac{d}{3}$ , firm A's price is monotonically increasing with the increase of  $c$ ; when  $\frac{d}{3} \leq s \leq \frac{4}{9}d$ , firm A's price is decreasing first and increasing later, with the increase of  $c$ ; When  $s > \frac{4}{9}d$ , firm A's price is monotonically decreasing with the increase of  $c$ . Firm B's price is monotone, and increasing with the increase of  $c$ .

*Corollary 4a. (Demand and  $s$ )* When  $0 < c \leq \frac{1}{2}d$ , firm A's demand is increasing first and decreasing later, with the increase of  $s$ ; when  $\frac{1}{2}d < c < d$ , firm A's demand is monotonically decreasing with the increase of  $s$ . Firm B's demand is monotonically decreasing with the increase of  $s$ .

*Corollary 4b. (Demand and  $d$ )* Firm A's demand is monotonically increasing with the increase of  $d$ . Firm B's demand is monotonically increasing with the increase of  $d$ .

*Corollary 4c. (Demand and  $c$ )* Firm A's demand is monotonically decreasing with the increase of  $c$ . When  $0 < s \leq \frac{d}{25}$ , firm B's demand is increasing first and decreasing later, with the increase of  $c$ ; when  $s > \frac{d}{25}$ , firm B's demand is monotonically increasing with the increase of  $c$ .

Table 4.5 Relationships of Optimal Demand With Benefit ( $s$ ) And Cost ( $c, d$ )		
	Demand of Firm A	Demand of Firm B
$s$	 <p>Demand of Firm A</p> <p>Additional Benefit (<math>s</math>)</p> <p><math>0 &lt; c \leq \frac{1}{2}d</math></p>	 <p>Demand of Firm B</p> <p>Additional Benefit (<math>s</math>)</p>
	 <p>Demand of Firm A</p> <p>Additional Benefit (<math>s</math>)</p> <p><math>\frac{1}{2}d &lt; c &lt; d</math></p>	
$d$	 <p>Demand of Firm A</p> <p>Development Cost (<math>d</math>)</p>	 <p>Demand of Firm B</p> <p>Development Cost (<math>d</math>)</p>
$c$	 <p>Demand of Firm A</p> <p>Customization Cost (<math>c</math>)</p>	 <p>Demand of Firm B</p> <p>Customization Cost (<math>c</math>)</p> <p><math>0 &lt; s \leq \frac{d}{25}</math></p>
		 <p>Demand of Firm B</p> <p>Customization Cost (<math>c</math>)</p> <p><math>s &gt; \frac{d}{25}</math></p>

From corollary 2, 3, 4, we get the following insights:

First, we find that when the development cost is higher, both firm A and B will obtain more profits and higher demands and set higher prices. It is because high development cost makes few users choose to self-develop part of the product. Firm A would open more part to attract more advanced users, although it would lose some basic users because of opening more part. Thus, both firms may have high market share when

development cost is high.

Second, for firm B, the increase of additional benefit makes its profit decrease; the increase of customization cost makes its profit increase. It is good for users who would like to use product with open source part when the additional benefit is high. Some basic users, who decide to choose firm B's pure proprietary product initially, may use firm A's hybrid product because of high additional benefit. However, if customization cost is high, there would be many users who want to use pure proprietary product because they do not have enough computer skills to customize open source product.

Third, we find that the impact of additional benefit ( $s$ ) or customization cost ( $c$ ) on firms' profit, price and demand are different according to different relationships between  $s$  or  $c$  and development cost ( $d$ ). For example, with the increase of  $s$ , when customization cost is relatively small ( $0 < c \leq \frac{3}{4}d$ ), firm A's profit is increasing first and decreasing later; when customization cost is relatively large ( $\frac{3}{4}d < c < d$ ), firm A's profit is monotonically decreasing. For another example, with the increase of  $c$ , when additional benefit is relatively small ( $0 < s < \frac{d}{5}$ ), firm A's profit is decreasing first and increasing later; when additional benefit is relatively large ( $s \geq \frac{d}{5}$ ), firm A's profit is monotonically decreasing.

### 4.3.2 Case 2: There is a Competing Pure Open Source Product

In the duopoly market, which is dominated by firm A and B, some users may choose to self-develop the software in order to achieve more flexibility. However, if there is a pure open source product available, they do not need to self-develop the software.

What strategies will the firms adopt, open or not open?

There are three options for customers: (1) buy the partial product from firm A at price  $P_A$  and integrate it with open source component provided by firm A; (2) buy the partial product from firm B at price  $P_B$  and integrate it with open source component provided by firm B; (3b) adopt the pure open source product. Same as previous case, we define utility functions when customers choose different options as:

$$U_A = v - P_A - C(\alpha_A, \theta) + S(\alpha_A, \theta)$$

$$U_B = v - P_B - C(\alpha_B, \theta) + S(\alpha_B, \theta)$$

$$U_F = v - C(1, \theta) + S(1, \theta)$$

where  $v$  is consumer's valuation;  $U_A$  is the consumer's utility function by adopting option (1);  $U_B$  is the utility function by adopting option (2);  $U_F$  is the utility function by adopting option (3b). The first two utility functions are the same as those in Case 1. The only difference is that customers, who are going to use pure open source product, will not incur development cost any more. Therefore, the utility function for the third option is not related to development cost. We still assume the customization cost ( $c$ ) is less than the development cost ( $d$ ), so that when there is open source code available, the

users will not develop the same code by themselves. Same as the above case, we assume  $\alpha_A > \alpha_B$  to simplify the computation.

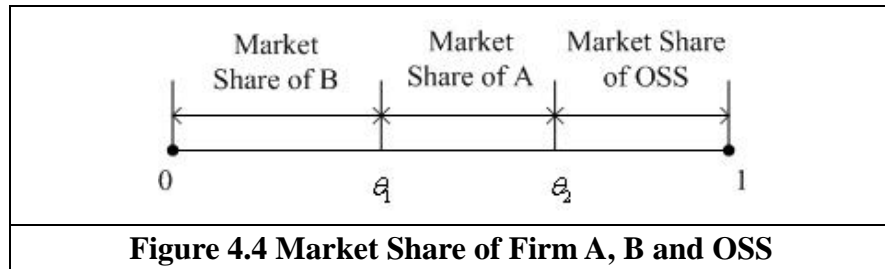
By letting  $U_A = U_B$ , we get the marginal user ( $\theta_1$ ) who is indifferent with option (1) and option (2). Similarly, by letting  $U_A = U_F$ , we get the marginal user ( $\theta_2$ ) who is indifferent with option (2) and option (3b). Users located at  $(0, \theta_1)$  are basic users who will choose option (2), since they have much higher cost but less benefit for using open source software. These basic users tend to use the software system with less open source part (firm B's product). Users located at  $(\theta_1, \theta_2)$  are advanced users who will choose option (1). Users located at  $(\theta_2, 1)$  are expert users who will choose pure open source product, since they can obtain more benefit from adopting open source but less cost for customization. We can solve the marginal user  $\theta_1$  and  $\theta_2$  as:

$$\theta_1 = \frac{(\alpha_A - \alpha_B)c + P_A - P_B}{(\alpha_A - \alpha_B)(s + c)}$$

$$\theta_2 = \frac{(1 - \alpha_A)c - P_A}{(1 - \alpha_A)(s + c)}$$

Firm A and B should optimally set the software price and degree of openness to maximize their profits. In order to focus on the competition between firm A and B and open source product, we assume that no consumer between 0 and 1 is "stranded" by three options discussed above. In the following part we will analyze firm A and B's optimal strategies to maximize their profits. The market size is normalized to 1, and thereby the market share of firm A and B and open source are  $\theta_2 - \theta_1$ ,  $\theta_1$  and  $1 - \theta_2$

respectively, which are shown in Figure 4.4.



We follow the same procedure as in Case 1 and solve the optimal strategies for both firms.

*Proposition 2.* When there are three competing systems in the software market: two of them are provided by two for-profit software firms, and one of them is a pure open source system, one of the software firms will not choose to open any part of the source codes; one of the firms will choose to open 43% of the source codes. ( $\alpha_B^* = 0$ ,  $\alpha_A^* = \frac{3}{7}$ )

It is interesting to find that when there is a competing pure open source system in the software market, the optimal degree of openness of firm A is not related to customization cost or additional benefit, i.e., it is constant. Since there are a pure open source and a pure proprietary product (provided by firm B) available in the market, firm A's product is in the middle of them. It gives users some flexibility benefit, but at the same time, users need to incur customization cost and pay for it. The degree of openness differentiates firm A's product from the pure open source and pure proprietary

product. If firm A opens more proportion, it will be more close to pure open source product, but more differentiated from firm B; if firm A opens less, it will be more close to firm B's product, but more differentiated from the open source product. Therefore, the optimal degree of openness is the key factor that best differentiates firm A's product from others. It is not surprised that the optimal degree of openness is constant in that the changes of  $c$  or  $s$  could not affect the differentiation between firm A and open source and firm B's product.

*Proposition 3.* When there are three competing systems in the software market: two of them are provided by two for-profit software firms: A and B, and one of them is a pure open source system, the optimal price, demand and profit for firm A is  $P_A^*$ ,  $D_A^*$ , and  $\pi_A^*$ , respectively; the optimal price, demand and profit for firm B is  $P_B^*$ ,  $D_B^*$ , and  $\pi_B^*$ , respectively. All the optimal solutions are shown in Table 4.6.

<b>Table 4.6 Optimal Strategy When There is a Competing Pure Open Source Product</b>			
	Firm A	Firm B	Comparison
Profit	$\pi_A^* = \frac{c^2}{48(c+s)}$	$\pi_B^* = \frac{7c^2}{48(c+s)}$	$\frac{\pi_B^*}{\pi_A^*} = 7$
Price	$P_A^* = \frac{c}{14}$	$P_B^* = \frac{c}{4}$	$\frac{P_B^*}{P_A^*} = \frac{7}{2}$
Demand	$D_A^* = \frac{7c}{24(c+s)}$	$D_B^* = \frac{7c}{12(c+s)}$	$\frac{D_B^*}{D_A^*} = 2$

Shown in proposition 3, the optimal strategies are determined by customization cost ( $c$ ) and additional benefit from adopting open source ( $s$ ). Through comparative static

analysis (shown in Table 4.7), we can find how these cost and benefit affect the optimal strategies of software firms, which are presented in Corollary 5, 6, 7.

<b>Table 4.7 Comparative Static Analysis</b>				
		Profit	Price	Demand
Firm A	$c$	$\frac{\partial \pi_A^*}{\partial c} = \frac{c(c+2s)}{48(c+s)^2} > 0$	$\frac{\partial P_A^*}{\partial c} = \frac{1}{14} > 0$	$\frac{\partial D_A^*}{\partial c} = \frac{7s}{24(c+s)^2} > 0$
	$s$	$\frac{\partial \pi_A^*}{\partial s} = -\frac{c^2}{48(c+s)^2} < 0$	$\frac{\partial P_A^*}{\partial s} = 0$	$\frac{\partial D_A^*}{\partial s} = -\frac{7c}{24(c+s)^2} < 0$
Firm B	$c$	$\frac{\partial \pi_B^*}{\partial c} = \frac{7c(c+2s)}{48(c+s)^2} > 0$	$\frac{\partial P_B^*}{\partial c} = \frac{1}{4} > 0$	$\frac{\partial D_B^*}{\partial c} = \frac{7s}{12(c+s)^2} > 0$
	$s$	$\frac{\partial \pi_B^*}{\partial s} = -\frac{7c^2}{48(c+s)^2} < 0$	$\frac{\partial P_B^*}{\partial s} = 0$	$\frac{\partial D_B^*}{\partial s} = -\frac{7c}{12(c+s)^2} < 0$

When there are three competing systems in the software market: two of them are provided by two for-profit software firms: A and B, and one of them is a pure open source system,

*Corollary 5a. (Profit and  $c$ )* firm A's optimal profit is monotonically increasing with the increase of  $c$ ; firm B's optimal profit is monotonically increasing with the increase of  $c$ .

*Corollary 5b. (Profit and  $s$ )* firm A's optimal profit is monotonically decreasing with the increase of  $s$ ; firm B's optimal profit is monotonically decreasing with the increase of  $s$ .

*Corollary 6a. (Price and  $c$ )* firm A's optimal price is monotonically increasing with the increase of  $c$ ; firm B's optimal price is monotonically increasing with the increase



of  $c$ .

*Corollary 6b. (Price and  $s$ )* firm A's optimal price is constant with the increase of  $s$ ; firm B's optimal price is constant with the increase of  $s$ .

*Corollary 7a. (Demand and  $c$ )* firm A's optimal demand is monotonically increasing with the increase of  $c$ ; firm B's optimal demand is monotonically increasing with the increase of  $c$ .

*Corollary 7b. (Demand and  $s$ )* firm A's optimal demand is monotonically decreasing with the increase of  $s$ ; firm B's optimal demand is monotonically decreasing with the increase of  $s$ .

*Proposition 4.* When there are three competing systems in the software market: two of them are provided by two for-profit software firms: A and B, and one of them is a pure open source system, both firms are profitable, but the optimal profit, price and demand of firm A are always lower than those of firm B: firm B's optimal profit is 7 times larger than firm A's; B's market share is twice of A's; B's price is 3.5 times higher than A's.

These results are reasonable and coincide with the reality. There are three competing software products in the market: one is a pure proprietary software product (provided by firm B), one is a pure open source software product, and one is between them -- a hybrid product of proprietary and open source software product (provided by firm A).

The designers of firm A need to consider both firm B and open source product when they design the product and sets price. In order to make sure that fewer users will choose open source system, they need to open more part and set lower price; at meanwhile, in order to capture more users from firm B, they need to open less part and set lower price. Firm A wants to squeeze both the demand for firm B and open source. Therefore, firm A faces more severe competitions from both firm B and open source.

Social welfare includes the total utilities of users, and the profits of firm A and firm B.

Thus, when market is fully covered, we define social welfare as:

$$W = \int_0^{\theta_1} U_B d\theta + \int_{\theta_1}^{\theta_2} U_A d\theta + \int_{\theta_2}^1 U_F d\theta + \pi_A + \pi_B$$

*Proposition 5.* If there is a competing pure open source product in the market, the social welfare when commercial firms adopt optimal strategies goes to:

$$W^* = \frac{24cv + 12s(s + 2v) - c^2}{24(c + s)}$$

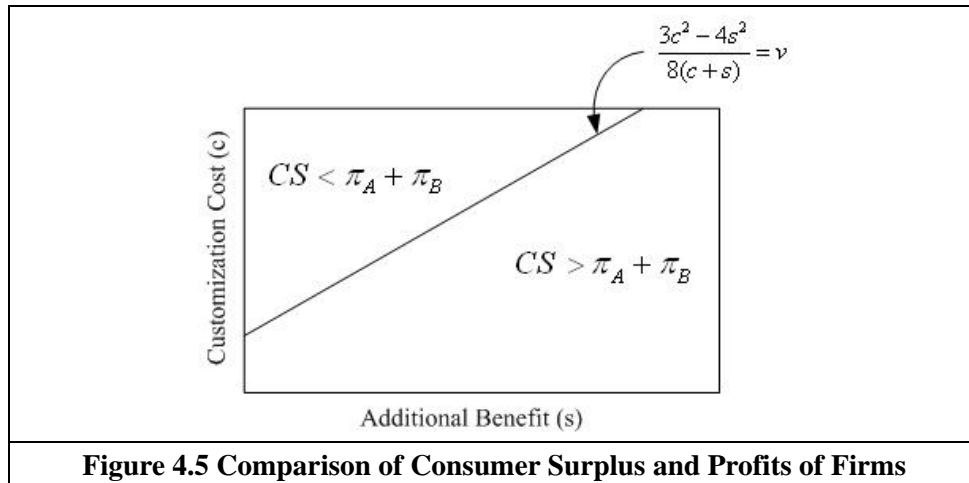
In which, consumer surplus is

$$CS^* = \frac{24cv + 12s(s + 2v) - 5c^2}{24(c + s)}$$

*Proposition 6.* When  $\frac{c}{s} \leq \frac{2}{\sqrt{3}}$ ,  $CS^* > \pi_A^* + \pi_B^*$ ; when  $\frac{c}{s} > \frac{2}{\sqrt{3}}$  and  $\frac{3c^2 - 4s^2}{8(c + s)} < v$ ,

$CS^* > \pi_A^* + \pi_B^*$ ; when  $\frac{c}{s} > \frac{2}{\sqrt{3}}$  and  $0 < v < \frac{3c^2 - 4s^2}{8(c+s)}$ ,  $CS^* < \pi_A^* + \pi_B^*$  (shown in

Figure 4.5).



Proposition 6 shows that when customization cost is relatively small to additional

benefit ( $\frac{c}{s} \leq \frac{2}{\sqrt{3}}$ ), the consumer surplus would be larger than firms' profits; when

customization cost is relatively large to additional benefit ( $\frac{c}{s} > \frac{2}{\sqrt{3}}$ ), customers'

valuation to the product determines whether the consumer surplus is larger than firms'

profits or not. If valuation is high ( $v > \frac{3c^2 - 4s^2}{8(c+s)}$ ), consumer surplus is larger than

profits; otherwise, consumer surplus is less than profits.

In summary, the model examines the optimal strategies of two commercial firms

competing with each other. Their strategies are determined by the customization cost,

development cost and additional benefit defined previously. It implies that the nature of

the product is important for commercial firms in product design and setting price. In the other words, the commercial firms' decision making on product design and price setting depends on the heterogeneity of the product. For example, for the complex software product, the customization cost will be high. If someone wants to self-develop it, the development cost will also be high. For the simple software product, the customization cost will be low, and if someone wants to self-develop it, the development cost will be low too. The strategies of commercial firms responding to complex software or simple software must be different. These differences have been clearly specified in the model. Furthermore, we compare the firms' optimal strategies under different situations: there is or is not a competing open source product available in the software market. The results suggest that when there is a competing pure open source product, one of the firms would not open any source code and provide a pure proprietary software system; the other firm would open the fixed source code, which is not related to the nature of the product. When there is no competing pure open source product in the market, one of the firm would also provide a pure proprietary software system, and the other firm would provide a hybrid product, of which how much of open would determined by the nature of the product.

#### **4.4 Concluding Remarks**

Open source software has been gaining popularity among individuals and organizations as a "free" alternative to traditional proprietary software. The popularity of open source

not only brings competition to proprietary software, but also awakes the software firms to adopt open source strategy to enhance their competitive advantage. Many industry practices suggest that participating open source may bring software firms profitability. How much does the firm earn from participating open source? To what extent the firm should participate open source? This research seeks to answer these questions by an economic modeling approach.

Our model examines the optimal strategies of two profit-oriented firms' competition. In order to have more competing advantages, they consider whether to adopt the open source strategy or not. In the other words, they need to find whether they should open part of the source code and how much they need to open. We find that the software firms' decision making on product design and price setting depends on the heterogeneity of the product. The characteristics of the product determine the strategies of software firms because these characteristics directly influence the customization cost, development cost and addition benefit, which are the key determinants of firms' strategies. The impact of these cost and benefit on the firms' strategies has been analyzed in the above corollaries. Some interesting results may shed light on pricing and product design of software firms when facing competitions from colleagues.

First, whatever there is a competing pure open source product or not, one of the firms would choose to provide pure proprietary software. The firm, which chooses not to open any part of the source code, targets at the most basic users in the software

market. This kind of users does not have any advanced computer skills, so that they do not have enough ability to customize the systems according to their own needs by themselves. Therefore, they do not have intention to use the products with the open source part. The firm knows this point! The firm will set a “good” price to let it get most profit and at the same time ensure these users to accept this price to buy its product.

Second, we find that when there are three competing systems in the software market: two of them are provided by two for-profit software firms: A and B, and one of them is a pure open source system, the optimal degree of openness of firm A is not related to customization cost or additional benefit, i.e., it is constant. Since there are a pure open source and a pure proprietary product (provided by firm B) available in the market, firm A's product is in the middle of them. It gives users some flexibility benefit, but at the same time, users need to incur customization cost and pay for it. The degree of openness differentiates firm A's product from the pure open source and pure proprietary product. If firm A opens more proportion, it will be more close to pure open source product, but more differentiated from firm B; if firm A opens less, it will be more close to firm B's product, but more differentiated from the open source product. Therefore, the optimal degree of openness is the key factor that best differentiates firm A's product from others. It is not surprised that the optimal degree of openness is constant in that the changes of  $c$  or  $s$  could not affect the differentiation between firm A and open source and firm B's product.

Third, when there are three competing systems in the software market: two of them are provided by two for-profit software firms: A and B, and one of them is a pure open source system, both firms are profitable, but the optimal profit, price and demand of firm A are always lower than those of firm B. The designers of firm A need to consider both firm B and open source product when they design the product and sets price. In order to make sure that fewer users will choose open source system, they need to open more part and set lower price; at meanwhile, in order to capture more users from firm B, they need to open less part and set lower price. Firm A wants to squeeze both the demand for firm B and open source. Therefore, firm A faces more severe competitions from both firm B and open source.

Our study has limitations that provide avenues for future research. We assume customers are uniformly located between 0 and 1 in the current study. We will use general form to analyze the characteristic of the firms' strategies in the future research. To extend this study, we will consider the network effects to see the impact of the results.

## References

Bessen, J. "Open source software: free provision of complex public goods," Working Paper, Available at SSRN <http://ssrn.com/abstract=588763>, July 2005.

Bonaccorsi A. and Rossi C. "Why open source software can succeed," *Research Policy* (32:7), July 2003, pp.1243-1258.

Bonaccorsi A., Giannangeli S., and Rossi C. "Entry strategies under competing standards: hybrid business models in the open source software industry," *Management Science* (52:7), July 2006, pp. 1085-1098.

Casadesus-Masanell R. and Ghemawat P. "Dynamic mixed duopoly: a model motivated by Linux vs. Windows," *Management Science* (52:7), July 2006, pp. 1072-1084.

Galli P "Open source code finds way into Microsoft product," eWeek.com article, September 2005.

Gaudeul A. "Competition between open-source and proprietary software: the (La)TeX case study," Working Paper 0409007, Industrial Organization from EconWPA, September 2004.



Roberts J., Hann I., and Slaughter S. "Understanding the motivations, participation and performance of open source software developers: a longitudinal study of the Apache projects," *Management Science* (52:7), July 2006, pp. 984-999.

Johnson J. P. "Collaboration, peer review and open source software," Working Paper, Available at SSRN <http://ssrn.com/abstract=535022>, May 2004.

Kuan J. "Open source software as consumer integration into production," Working Paper, Available at SSRN <http://ssrn.com/abstract=259648>, January 2001.

Lakhani R. K. and Wolf G. R. "Why hackers do what they do: understanding motivation and effort in free/open source software projects," in *Perspectives on Free and Open Source Software*, Joseph Feller et al. (Ed.), MIT Press, Cambridge, MA, 2005, pp. 3-22.

Lerner J. and Tirole J. "Some simple economics of open source," *Journal of Industrial Economics* (52:2), 2002, pp. 197-234.

Massel, D. "The changing face of open source," LinuxInsider.com article, October 2005.

Schmidt K. and Schnitzer M. "Public subsidies for open source? Some economic policy issues of the software market," CEPR Discussion Paper 3793, February 2003.

Schotter A. *Microeconomics: A Modern Approach*, Addison Wesley Longman, Hong Kong, 2001.

Shell, S. "Open source vs. commercial software: why proprietary software is here to stay," Informit.com article, October 2005.

Taft D.K. "The keys to open-source success," eWeek.com article, December 2005.

Wheeler D. A "Why open source software/free software (OSS/FS)? Look at the number!" Available at: [http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html), April 2007.

## Appendix

When there is no competing open source software available in the software market: first we solve maximization profit functions under the known degree of openness ( $\alpha_A, \alpha_B$ ), and get the pre-optimal price ( $P'_A, P'_B$ ) of each firm; second, we substitute price ( $P_A, P_B$ ) in the profit functions with pre-optimal price ( $P'_A, P'_B$ ), and solve optimal the degree of openness ( $\alpha_A^*, \alpha_B^*$ ) and then get ( $P_A^*, P_B^*$ ).

$$\text{Max}_{P_A} \pi_A = (\theta_2 - \theta_1)P_A$$

$$\text{Max}_{P_B} \pi_B = \theta_1 \cdot P_B$$

We get the pre-optimal price ( $P'_A, P'_B$ ) of each firm and substitute them into each profit function and obtain pre-optimal profit ( $\pi'_A, \pi'_B$ ).

$$\text{Max}_{\alpha_A} \pi'_A = \frac{(1 - \alpha_A)(\alpha_A - \alpha_B)(c(d - s) + 2ds)^2[\alpha_A(c - d) + d + s - \alpha_B(c + s)]}{(s + c)(s + d)[-4\alpha_B(c + s) + \alpha_A(4c - 3d + s) + 3(s + d)]^2}$$

$$\text{Max}_{\alpha_B} \pi'_B = \frac{(\alpha_A - \alpha_B)[2cd + cs + ds - 2\alpha_B c(c + s) + \alpha_A(c - d)(2c + s)]^2}{(s + c)[-4\alpha_B(c + s) + \alpha_A(4c - 3d + s) + 3(s + d)]^2}$$

We find that

$$s > 0 \ \& \ c \geq \frac{2s}{23}, \ \alpha_B^* = 0$$

$$s > 0 \ \& \ 0 < c < \frac{2s}{23} \ \& \ d \geq \frac{25cs}{2s - 23c}, \ \alpha_B^* = \alpha_{B1}^* \ \text{or} \ \alpha_{B2}^* \ \text{or} \ 0$$

$$s > 0 \& 0 < c < \frac{2s}{23} \& c < d < \frac{25cs}{2s-23c}, \alpha_B^* = 0$$

in which

$$\begin{aligned} \alpha_{B1} &= \frac{8aAc^2 + 5cd - 5aAcd + 7cs + aAcs - 2ds + 2aAds}{8c(c+s)} + \\ &\frac{1}{8} \\ &\sqrt{\left( \frac{1}{c^2(c+s)^2} (-23c^2d^2 + 46aAc^2d^2 - 23aA^2c^2d^2 - 2c^2ds + 4aAc^2ds - 2aA^2c^2ds - 44cd^2s + \right. \\ &\quad 88aAc d^2s - 44aA^2cd^2s + 25c^2s^2 - 50aAc^2s^2 + 25aA^2c^2s^2 - 52cds^2 + 104aAcds^2 - \\ &\quad \left. 52aA^2cds^2 + 4d^2s^2 - 8aAd^2s^2 + 4aA^2d^2s^2) \right)} \\ \alpha_{B2} &= \frac{8aAc^2 + 5cd - 5aAcd + 7cs + aAcs - 2ds + 2aAds}{8c(c+s)} - \\ &\frac{1}{8} \\ &\sqrt{\left( \frac{1}{c^2(c+s)^2} (-23c^2d^2 + 46aAc^2d^2 - 23aA^2c^2d^2 - 2c^2ds + 4aAc^2ds - 2aA^2c^2ds - 44cd^2s + \right. \\ &\quad 88aAc d^2s - 44aA^2cd^2s + 25c^2s^2 - 50aAc^2s^2 + 25aA^2c^2s^2 - 52cds^2 + 104aAcds^2 - \\ &\quad \left. 52aA^2cds^2 + 4d^2s^2 - 8aAd^2s^2 + 4aA^2d^2s^2) \right)} \end{aligned}$$

$$\text{Max}_{\alpha_A} \pi_A'(\alpha_B \rightarrow \alpha_{B1}^*, \alpha_B \rightarrow \alpha_{B2}^*)$$

We find that (1)  $\alpha_A^* = \alpha_{B1}^*$  or (2)  $\alpha_A^* = \alpha_{B2}^*$  or (3)  $\alpha_B^* = 0$ . The first two solutions are not satisfied with our requirements, because the special case that  $\alpha_A = \alpha_B$  have been discussed in Chapter 3. Here, we only examine the third solution. When  $\alpha_B^* = 0$ ,  $\alpha_A^*$  has three possible solutions:  $\alpha_A^* = \alpha_{A1}^*$ , or  $\alpha_A^* = \alpha_{A2}^*$ , or  $\alpha_A^* = \alpha_{A3}^*$ .

$$\begin{aligned}
aA1 = & -\frac{3(cd-d^2+cs-ds)}{4c^2-7cd+3d^2+cs-ds} - \\
& (2^{1/3}(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds)(-2cd+9d^2-2cs+16ds+7s^2))) / \\
& (3(4c^2-7cd+3d^2+cs-ds) \\
& (648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+648c^4s^2+ \\
& 1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3-1296c^2ds^3-1296cd^2s^3+ \\
& 1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4+ \\
& \sqrt{((648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+ \\
& 648c^4s^2+1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3- \\
& 1296c^2ds^3-1296cd^2s^3+1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4)^2+ \\
& 4(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds) \\
& (-2cd+9d^2-2cs+16ds+7s^2))^3})^{1/3}) + \\
& (648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+648c^4s^2+ \\
& 1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3-1296c^2ds^3-1296cd^2s^3+ \\
& 1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4+ \\
& \sqrt{((648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+ \\
& 648c^4s^2+1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3-1296c^2ds^3- \\
& 1296cd^2s^3+1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4)^2+ \\
& 4(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds)(-2cd+9d^2-2cs+16ds+7s^2))^3})^{1/3} / \\
& (32^{1/3}(4c^2-7cd+3d^2+cs-ds))
\end{aligned}$$

$$\begin{aligned}
aA2 = & -\frac{3(cd-d^2+cs-ds)}{4c^2-7cd+3d^2+cs-ds} + \\
& ((1+i\sqrt{3})(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds)(-2cd+9d^2-2cs+16ds+7s^2))) / \\
& (32^{2/3}(4c^2-7cd+3d^2+cs-ds) \\
& (648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+648c^4s^2+ \\
& 1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3-1296c^2ds^3-1296cd^2s^3+ \\
& 1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4+ \\
& \sqrt{((648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+ \\
& 648c^4s^2+1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3- \\
& 1296c^2ds^3-1296cd^2s^3+1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4)^2+ \\
& 4(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds) \\
& (-2cd+9d^2-2cs+16ds+7s^2))^3})^{1/3}) - \\
& ((1-i\sqrt{3}) \\
& (648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+648c^4s^2+ \\
& 1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3-1296c^2ds^3-1296cd^2s^3+ \\
& 1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4+ \\
& \sqrt{((648c^4d^2-1296c^3d^3+648c^2d^4+1296c^4ds-1296c^3d^2s-1296c^2d^3s+1296cd^4s+ \\
& 648c^4s^2+1296c^3ds^2-3888c^2d^2s^2+1296cd^3s^2+648d^4s^2+1296c^3s^3- \\
& 1296c^2ds^3-1296cd^2s^3+1296d^3s^3+648c^2s^4-1296cds^4+648d^2s^4)^2+ \\
& 4(-81(cd-d^2+cs-ds)^2+3(4c^2-7cd+3d^2+cs-ds) \\
& (-2cd+9d^2-2cs+16ds+7s^2))^3})^{1/3}) / (62^{1/3}(4c^2-7cd+3d^2+cs-ds))
\end{aligned}$$

$$\begin{aligned}
a\lambda_3 = & -\frac{3(c d - d^2 + c s - d s)}{4c^2 - 7cd + 3d^2 + cs - ds} + \\
& \left( (1 - i\sqrt{3}) (-81(c d - d^2 + c s - d s)^2 + 3(4c^2 - 7cd + 3d^2 + cs - ds)(-2cd + 9d^2 - 2cs + 16ds + 7s^2)) \right) / \\
& \left( 3^{2/3} (4c^2 - 7cd + 3d^2 + cs - ds) \right. \\
& (648c^4d^2 - 1296c^3d^3 + 648c^2d^4 + 1296c^4ds - 1296c^3d^2s - 1296c^2d^3s + 1296cd^4s + 648c^4s^2 + \\
& 1296c^3ds^2 - 3888c^2d^2s^2 + 1296cd^3s^2 + 648d^4s^2 + 1296c^3s^3 - 1296c^2ds^3 - 1296cd^2s^3 + \\
& 1296d^3s^3 + 648c^2s^4 - 1296cds^4 + 648d^2s^4 + \\
& \sqrt{((648c^4d^2 - 1296c^3d^3 + 648c^2d^4 + 1296c^4ds - 1296c^3d^2s - 1296c^2d^3s + 1296cd^4s + \\
& 648c^4s^2 + 1296c^3ds^2 - 3888c^2d^2s^2 + 1296cd^3s^2 + 648d^4s^2 + 1296c^3s^3 - \\
& 1296c^2ds^3 - 1296cd^2s^3 + 1296d^3s^3 + 648c^2s^4 - 1296cds^4 + 648d^2s^4)^2 + \\
& 4(-81(c d - d^2 + c s - d s)^2 + 3(4c^2 - 7cd + 3d^2 + cs - ds) \\
& \left. (-2cd + 9d^2 - 2cs + 16ds + 7s^2))^3 \right)^{1/3} - \\
& \left. \left( (1 + i\sqrt{3}) \right. \right. \\
& (648c^4d^2 - 1296c^3d^3 + 648c^2d^4 + 1296c^4ds - 1296c^3d^2s - 1296c^2d^3s + 1296cd^4s + 648c^4s^2 + \\
& 1296c^3ds^2 - 3888c^2d^2s^2 + 1296cd^3s^2 + 648d^4s^2 + 1296c^3s^3 - 1296c^2ds^3 - 1296cd^2s^3 + \\
& 1296d^3s^3 + 648c^2s^4 - 1296cds^4 + 648d^2s^4 + \\
& \sqrt{((648c^4d^2 - 1296c^3d^3 + 648c^2d^4 + 1296c^4ds - 1296c^3d^2s - 1296c^2d^3s + 1296cd^4s + \\
& 648c^4s^2 + 1296c^3ds^2 - 3888c^2d^2s^2 + 1296cd^3s^2 + 648d^4s^2 + 1296c^3s^3 - \\
& 1296c^2ds^3 - 1296cd^2s^3 + 1296d^3s^3 + 648c^2s^4 - 1296cds^4 + 648d^2s^4)^2 + \\
& 4(-81(c d - d^2 + c s - d s)^2 + 3(4c^2 - 7cd + 3d^2 + cs - ds) \\
& \left. (-2cd + 9d^2 - 2cs + 16ds + 7s^2))^3 \right)^{1/3} \left. \right) / (6^{1/3} (4c^2 - 7cd + 3d^2 + cs - ds))
\end{aligned}$$

When  $c, s, d$  has different combinations, the optimal degree of openness of firm A is one of the above three solutions.

## **Chapter 5. Conclusion and Future Work**

---

This thesis applies social network analysis and economic theory and methodology in Information Systems research to study issues associated with open source software projects and their applications in the software industry. Three essays examine three issues: survival of OSS, competition between OSS and proprietary software, and competition between two software firms. A few implications from these studies are summarized in the following sections.

### **5.1 Evaluating Longitudinal Success of Open Source Software Projects**

The main purpose of this study is to investigate the long term effects of communication pattern on the success of open source projects. We base our research on the theoretical study of social network theory. Generally speaking, by observing changes in communication patterns for an extended period, we find significant impacts of

communication patterns on the outcome of the project.

The findings of our research has implications for project managers and developers in open source environments, as well as for managers of commercial software firms, which are actively participating in open source projects. The project managers need to realize the importance of the communication pattern of project team. According to the objectives of projects, a proper and planned control for the communication among team members is crucial for the survivability of the open source projects. Since the view of project success from developers and general users are different, the project managers can reap the benefits if they structure their project teams with care.

From a theoretical standpoint, we apply social network theory into the information systems domain, in particular, into the study of success of OSS projects. Our results suggest several directions for theory development on the effect of communication pattern of the project team on project success. First, it is important to recognize that the effect of communication pattern varies with the indicators of project success. OSS success has different dimensions. A single measurement or operationalization will not be sufficient to completely represent success. Second, it is important to recognize that success is transient. The changes of project status are caused by the dynamic communication pattern. Examining the status of projects over an extended period is a more rigorous method to assess the long term evolving success of OSS projects. Our research takes some important steps in this direction and we hope that further investigations of long term success from social network perspective are explored in



the future research.

## **5.2 Optimal Software Design and Pricing**

In this study, we employ two models: a one-dimensional Hotelling model and a two-dimensional vertical differentiation model, to study the optimal strategies of commercial firms when consumers have different tastes. In particular, we seek to solve for the optimal design and pricing of proprietary software. Further, we analyze the impact of network externalty on the optimal strategy and profit of the commercial firm in the Hotelling model. The analysis in the Hotelling model suggests that the profit of the commercial firm is dependent on the fit cost and the positioning of open source software. Both the profit of the commercial firm and social welfare are maximized when the open source software targets more specialized users. From the second model, we find that the optimal strategies are not unique for a commercial firm. Different characteristics of open source product will lead to different strategies. Furthermore, consumers are always better off when the open source software provides better usability or functionality. It is a good situation for users that open source provides better usability and functionality. At the end, when the open source has sufficient better advantage over functionality relative to its disadvantage in user interface, the commercial firm will be driven out of the market.

Our research can be extended in the following directions. For model 1, first, we shall

analyze whether the commercial firm should reach out to attract open source users to compete. That is, whether all customers located between open source and proprietary software should be served. Secondly, we shall explore the possibility of different network intensity for open source and proprietary software. Finally, we shall provide answers to optimal strategies for the more complicate situation when the fit cost is high. For model 2, we will investigate a two-period model with network externality. In considering network externality, we will study the impact of new release (upgrade) of open source software. The commercial firm needs to optimally upgrade its software and set up a new price in order to maximize profit.

### **5.3 Partially Opening Source Code**

The popularity of open source not only brings competition to proprietary software, but also awakes the software firms to adopt open source strategy to enhance their competitive advantage. This model examines the optimal strategies of two profit-oriented firms' competition. In order to have more competing advantages, they consider whether to adopt the open source strategy or not. We find that the software firms' decision making on product design and price setting depends on the heterogeneity of the product. The characteristics of the product determine the strategies of software firms because these characteristics directly influence the customization cost, development cost and addition benefit, which are the key determinants of firms' strategies. We find that whatever there is a competing pure open source product or not,

one of the firms would choose to provide pure proprietary software. Furthermore, when there are three competing systems in the software market: two of them are provided by two for-profit software firms and one of them is a pure open source system, the optimal degree of openness of both firms are not related to customization cost or additional benefit, i.e., they are constant. In this situation, there are a pure open source, a pure proprietary product, and a hybrid product of open source and proprietary software.

This study has limitations that provide avenues for future research. We assume customers are uniformly located between 0 and 1 in the current study. We will use general form to analyze the characteristic of the firms' strategies in the future research. To extend this study, we will consider the network effects to see the impact of the results.