Dissertation

Essays on Software Market: Security, Liability, and Pricing

By

Byung Cho Kim

Committee

Chair: Professor Tridas Mukhopadhyay

Professor Pei-yu Chen

Professor Chris Forman

Reader

Professor Rahul Telang

Tepper School of Business

Carnegie Mellon University

August 2007

Dedication

This dissertation is dedicated to my parents, Sang Wook Kim and Chang Hee Kim,

for their unconditional sacrifice, endless support, and love.

Acknowledgements

Pursuing my doctoral degree at Carnegie Mellon has been a challenging yet rewarding experience of my life. All I had was luck, the best of which was in the form of my advisor, Professor Tridas Mukhopadhyay who is not only my academic advisor, but also my mentor. He has helped me go through all the difficulties I came across for the past five years at Carnegie Mellon. I would like to sincerely thank him for his continuous guidance and support throughout my doctoral program. He has spent countless hours discussing my research, sharing his insights and experiences with me, guiding me through tough times, and sharpening my learning skills. Without his support, patience, and confidence in me, this dissertation would not be possible. I feel honored to have had a chance to work with him and learn from him.

I am grateful for the contributions of Professor Pei-yu Chen who has always encouraged me with belief in me since my first day at Carnegie Mellon. She has helped me develop analytical skills as well as research ideas. I have benefited from every single interaction with her. She has dedicated her time and effort to help me become a critical thinker. My appreciation for her continuous guidance and support is immeasurable.

I would like to acknowledge with great appreciation the willingness of Professor Chris Forman to be my committee member. I have benefited from his insightful comments on my dissertation. I would also like to thank Professor Rahul Telang for his suggestions as an external reader. Interactions with him through his seminar have helped me develop research ideas in the area of information security.

I would like to express my gratitude to my brother, Byung Do Kim, who has been my role model. Without his guidance, pursuing an academic career would be like sailing a ship without a

map and compass. I would also like to thank my brother, Byung Joon Kim for his incredible support for the family, which made me stand strong throughout my doctoral program. I am grateful to my sister, Se Rin Kim for her love and support.

I owe a tremendous debt of gratitude to my wife, Jin Sun Lee who has supported me with incredible patience and unwavering confidence in me. Nothing would be possible without her sacrifice in this long journey. She has given me everything, best of which is our baby we are looking forward to meeting soon.

I dedicate this dissertation to my parents, Sang Wook Kim and Chang Hee Kim, who have supported me in prayer for such a long time. They never lost faith and confidence in me even when I could not quite believe in myself. I wish I could have more time with my father who deserves my planned repayment.

Finally, I am grateful for God’s provision of joys, challenges, and grace for growth.

Table of Contents

Abstract

Software has become an indispensable part of our daily lives. Not only large-scale operating systems in businesses but also home electric appliances such as television sets and telephones are powered by thousands of program instructions. In the new era of information technology, understanding economic issues becomes a key success factor for the software industry, along with maintaining technological superiority. Despite the economic nature and the increasing attention from the stakeholders in the software industry, many problems of the software market have not been examined by the economists and hence still remain unanswered especially in the emerging areas such as software security and open source software.

My dissertation consists of three essays which examine two issues through an economic lens: (i) software security and liability and (ii) open source software pricing. I aim to make both academic and practical contributions by applying economic tools to analyze problems that have been intensely discussed by computer scientists, software engineers, policy makers and jurists, and by justifying ongoing argument on the two issues with academic rigor. Essay I proposes a risk-sharing mechanism between software vendors and customers as a possible way to incentivize the software vendors to improve software security. Essay II examines a policy-oriented approach of software liability and identifies conditions under which the legal liability mechanism is effective in terms of improving software security. In Essay III, I study the pricing mechanisms newly introduced to the software market as a way to make money from open source software. Comparison between commercial pricing model with open source software pricing models such as dual-licensing model and support model leads to the conditions under which each pricing model is viable. Factors that affect the success of the open source software pricing models are identified.

1 Introduction

As software has become an indispensable part of our daily lives, accelerated with the increased capabilities of computing devices, the market for software has become large, mature, and complicated. As a result, software industry in the new era of information technology is dealing with more economic issues than ever. In the early days of packaged software, software developers might be concerned about technology development far more than economic issues such as marketability, return on investment, customers' incentive problems, social benefit, network externality, and even profitability. We are now living in the world, where every single moment of our lives is aided by systems, devices, and appliances powered by thousands of program instructions. From small home electric appliances such as television sets, microwaves, washers and dryers, and computers, of course, to large-scale operating systems such as national power and energy systems, software plays a crucial role in our daily lives. More software vendors are competing in the market, more users are demanding a variety of software, and more businesses are dependent upon software. Thus, understanding economic issues which will eventually lead to making rational decisions becomes a key success factor in software business, along with maintaining technological superiority. For example, large software companies are increasingly relying on acquisitions for growth. One instance is Oracle that recently acquired PeopleSoft for $11.1 billion and Siebel Systems, the customer relationship management giant, for $6.1 billion.

Not only the economic growth of the software industry but also unique characteristics of the software industry have captured economists' attention. Software is a popular example of information goods and its production requires large fixed cost but negligible marginal cost which is considered to be significant for most physical goods. Also, network externality plays an

important role in the software market since software users often enjoy additional benefit from large networks, and so appreciate network size. Thus, managing compatibility and network size is identified to be a strategic choice of the software vendors. Even, there are issues or phenomena in the software market, which economists are not able to fully understand and explain with existing theory. One example is the motivation of software developers in open source software projects, which has not been fully explained with economic utility theory. Although more economists pay attention to the software market and examine the issues from an economic perspective, many important questions regarding the rapidly growing software industry still remain unanswered.

My dissertation examines the issues of software security, liability and pricing from an economic point of view. Specifically, I explore two issues in the software market, which have been controversial among software engineers, computer scientists, policy makers, and jurists: (i) software security and liability and (ii) open source software pricing. Grounded on economic theory, I aim to answer two fundamental questions. How can we give software vendors an incentive to make their software more secure? How can we make money from open source software? Software security and open source software are still virgin, yet promising areas to academic researchers with both technical and economic backgrounds. The aim of this dissertation is to provide answers to questions, one in the area of software security and the other in open source software market, which have not been examined by economists despite the economic nature of the problems. I aim to make contributions to the literature of software economics, information security, and open source software by examining the problems through an economic lens, which have been controversial among software experts from outside the domain of economics. Practically, my dissertation aims to give a guideline to the software

vendors who want to be successful in business, customers who want to make a right choice of the software, policy makers who want to make our computer networks secure, and the business who want to make money from open source software.

First, I examine the issue of software security and liability. The abundance of flawed software has been identified as the main cause of the poor security of computer networks. However, in the current state of the software market, software vendors do not have enough incentive to apply the available techniques to prevent vulnerabilities of their software since customers do not appreciate secure and safe software and security development is costly. As an incentive mechanism for software security improvement, software liability has been intensely discussed among computer scientists, jurists, and policy makers for a long time. Despite the long years of debate, the researchers have not come to a conclusion about the effectiveness of the liability mechanism. I examine two different forms of imposing liability on software vendors, which are (i) risk-sharing contract between software vendors and customers and (ii) legal liability controlled by the government.

In Essay I, I propose a risk-sharing mechanism between software vendors and customers as a market-driven and voluntary approach to impose liability on software vendors. Grounded on economic theory, my model examines the incentive of software vendors to share risk with their customers. The impact of risk-sharing on software security is studied. Software is considered to have two dimensions of quality, which are functionality and software security, and the focus of this study is software security. I find that software security is underprovided in the monopoly market as has been observed in reality and that the monopolist software vendor does not have any incentive to share risk with its customers. In the case of competition, a software vendor who

enters the market but is not able to differentiate its product from the existing monopolist software vendor's software may be willing to share risk with customers to alleviate competition.

Essay II analyzes the impact of legal and uniform software liability on software security in the monopoly market. The effectiveness of the liability mechanism is examined in terms of improving software security and increasing market coverage of the monopolist software vendor, both of which are important because policy makers should make sure that the monopolist is willing to serve enough customers with providing reasonably secure software. Overall, I find that liability guarantees neither better software security nor higher social surplus. Factors that affect the effectiveness of the liability mechanism are identified, such as marginal willingness to pay of the customers, cost-effectiveness of security development and the security awareness of the customers. Essay II examines how these factors affect the effectiveness of the legal liability on software security.

The second problem that my dissertation examines is open source software pricing. Recently, a new and interesting movement has been observed in the open source software industry. That is, some vendors of open source software seek to capitalize on the publicity and popularity of their products unlike most of other open source software which has been viewed as free public goods. Two approached have been successful in making money from open source software. One is the dual-licensing model, which has been successfully adopted by MySQL, and the other is the support model, which has been represented by Red Hat's support service model for its Enterprise Linux. Essay III examines the viability of these open source software pricing mechanisms which are new to the software industry under various scenarios. The optimal pricing strategies of both the commercial and the open source software vendors are studied given the customers who have two sources of heterogeneity: taste and technical savviness.

The rest of my dissertation is organized as follows. Section 2 presents Essay I which examines the risk-sharing mechanism between software vendors and customers. Essay II studies the policy oriented approach of software liability in Section 3. Finally, Section 4 delivers Essay III that examines the issue of open source software pricing. Methodologically, I use analytical tools including game theory and industrial organization.

2 An Economic Analysis of the Software Market with a Risk-Sharing Contract

2.1 Introduction

As the Internet has revolutionized the way individuals, industry and the government communicate and conduct their daily business, the intensive interconnectivity has increased the vulnerability of computer systems. Consequently, network security becomes a major issue for electronic business and corporate communications. To cope with the new risk, the computer industry has attempted to develop new weapons such as firewalls, encryption techniques, access control mechanisms, and intrusion detection systems. The federal government has formed the Department of Homeland Security, and is developing a National Strategy to Secure Cyber Space. Despite these efforts, the security level of computer networks is still low, and the potential loss is enormous. Ernst & Young's Global Information Security Survey 2004 shows that only 20% of the respondents strongly agreed that their organizations perceive information security as a CEO level priority and that less than half agreed that they could continue business operations in the event of a serious disruption (Ernst & Young 2004). The Computer Security Institute and the Federal Bureau of Investigation reported that total losses due to computer crimes and computer security incidents for the respondents to the 2004 survey were over 141 million US dollars (Gordon et al. 2004).

Fisk (2002) argues that there are well known technical and procedural techniques for preventing computer system vulnerability. However, applying these techniques can be resource-intensive and will not be done without a sufficient incentive. One major reason for poor security is that the software industry is at a sub-optimal, but self-supporting equilibrium that does not

support the efforts required for software security improvement. System Security Study Committee of the Computer Science and Telecommunications Board (CSTB) believes that customers do not have good enough safeguards, both because available options on the security market seem to be ineffective and too expensive, and the value of running safe operation is not fully appreciated (National Research Council 1991). Software users have learned to tolerate low-quality software, enabling the vendors to be successful without improving the quality of their products. On the vendor side, a perceived low willingness-to-pay for security quality in the market together with high development costs have made producing highly secure software a significant risk.

Slow growth of the security market and low quality of software have been identified as main causes of the poor state of network security (Yurcik and Doss 2002). However, software vendors have no incentive to improve the quality of their products since they are not directly liable for any loss due to poor quality. To solve this problem, security experts suggest legal liability and cyber insurance as possible solutions (Schneier 2004, Varian 2000). Unfortunately, not much research has been done on this problem from an economic perspective. According to Fisher (2002), some companies are already demanding liability clauses in contracts with vendors, holding the vendors responsible for any security breach connected to their software, quoting Karl Keller, president of IS Power Inc., who says, “Contractual liability is a great motivator. I’m encouraged that liability for vulnerabilities is entering into contracts.”

In this paper, we propose a risk-sharing mechanism between software vendors and customers as an alternative solution to improve software quality. With an economic model of the software market, we examine the incentive of software vendors to share risk with their customers and study the impact of risk-sharing on software quality. Our model accommodates two

dimensions of software quality, which are functionality and security quality, but our focus throughout the paper is security quality. Hence we use software quality and security quality interchangeably. We find the evidence of under-provided quality of software under monopoly, as has been observed in the market. The results show that the social planner who maximizes the benefit of the entire society offers a higher-quality product than the software vendor who pursues profit. We also find that imposing more risk on the software vendor does not guarantee higher quality. This intuition provides insights for the otherwise unexpected result that neither the social planner nor the monopolist has any incentive to bear the risk. In welfare economics, social planner is a decision maker who pursues the best outcome for all parties in the market. In the real-world software market, it can be the government whose goal is maximizing the sum of the software users' benefits and the software vendors' profit. Open source software vendors can be another example in that the individual software developers in the open source project often put priority on user benefit over cost. Our findings are interesting in the sense that even for such social planner, sharing risk with the customers is not an optimal strategy, although risk-sharing mechanisms such as warranties are widely used in other industries.

In the case of competition, a software vendor may have an incentive to share risk. We start by examining the case where a new vendor brings a product with the same quality level as the established monopolist vendor who does not want to share any risk. Unlike the monopoly case, we find that the new vendor has an incentive to introduce positive risk-sharing to alleviate competition, and that the risk-sharing level increases with the quality level. Next we extend this scenario to the case where two vendors with products of the same quality can differentiate their products by offering different levels of risk-sharing. We find that neither the high-value vendor nor the low-value vendor has an incentive to share any risk with customers. In other words, risk-

sharing may not act as a product differentiator without any regulation, meaning that the government should play a role in the software market as a market regulator.

We then examine the policy implications of the proposed risk-sharing mechanism. While we have shown that neither a monopolist software vendor nor a social planner would share risk when it is voluntary, the quality offered by a monopolist is lower than the socially optimal level. We investigate whether regulation on risk-sharing level gives an incentive for quality improvement to the monopolist. Interestingly, we find that the monopolist has an incentive to increase the quality when a certain level of risk-sharing is imposed by the government. Our results show that when security investment is more cost effective, that is, imposing the same amount of money on security development prevents higher loss, the government's imposing more liability on the software vendor leads to better outcome, that is, better software security. We also contrast this finding to government subsidy. Policy makers and computer security experts have suggested that the government should offer tax incentives to businesses for spending on security (Harmon 2003). We examine how the government subsidy affects the quality of the software in a monopoly. We find that a subsidizing policy creates no incentive for the vendor to bear risk and worse, it reduces the quality level. Although the subsidizing policy may lead to higher level of customer security awareness, it may make the situation worse in terms of software quality.

Finally, we discuss other possibilities in the software market. We first examine the software market where there exists information asymmetry between the monopolist software vendor and its customers. That is, customers are not capable of estimating the loss due to software quality while the software vendor has the correct expectation of the loss. Unlike the monopoly case with perfect information, the monopolist software vendor has an incentive to take full responsibility

when customers' perception of the loss is high. However, when customers underestimate the loss, the monopolist software vendor does not want to share any risk with its customers. It is consistent with the widely accepted security experts' argument that low level of software users' security awareness contributes to the poor software security (Siponen 2001). We then investigate whether the availability of the third party security solution such as cyber insurance and security software can put pressure on the monopolist software vendor to develop security. Interestingly, we find that the software vendor welcomes such third parties in the market so that customers manage their own security through such third party solutions. In that way, the monopolist software vendor does not have to invest on security development.

The rest of the paper is organized as follows. We present our model in section 2.2, and examine the incentives to adopt risk-sharing under the cases of a monopolist software vendor, a social planner, and duopoly competition in section 2.3. In section 2.4, we examine policy implications of the risk-sharing mechanism. Section 2.5 discusses two important issues regarding software liability: third party solution and information asymmetry. Finally, section 2.6 concludes the paper.

2.2 Model

We analyze a software vendor's decision on the quality and the risk-sharing levels, using an economic model of quality differentiation (Mussa and Rosen 1978, Ronnen 1991). There are two types of players in the market: a software vendor and customers. It is shown that the best strategy for the software vendors is to introduce their products as early as possible and then to patch them later (Arora et al. 2006). These findings are consistent with the penetrate-and-patch strategy which is often observed in the industry where entering the market early brings extra benefit to

the firms. Anderson (2001) argues that the prevalence of the penetrate-and-patch strategy makes security development difficult. Brady et al. (1999) discusses security development using the penetrate-and-patch strategy and the way to deal with vulnerabilities in a quantitative way. In the real world, we observe that software patching is a common way to deal with software security problems. For example, Microsoft issues patches on an almost daily basis (Economic Times 2002).

Consequently, the initial quality of software products is low. Security experts argue that the defects of such software are exploited by malicious hackers to attack computer systems, and that the quality of the general software in terms of security should be improved. Customers in our model are considered to be firms that are likely to have higher incentive to adopt security solutions than do individual consumers, whose awareness of security in general is still low in reality and may also have low bargaining power in demanding a risk-sharing clause. We assume that increasing the level of security quality of software reduces the expected loss from cyber attack in the life-span of the product. This is reasonable in the sense that attacks on computers or systems with more secure software are less likely to be successful.

2.2.1 Customer's Benefit Function

We consider two dimensions of software quality: functionality and security quality. In a memo in January 2002, Bill Gates instructed Microsoft to shift its top priority from adding new features to ensuring that software is secure and to stop all Windows feature development, and focus only on security improvement (Markoff 2002). Our model considers a vendor that emphasizes security quality given a certain level of functionality ($V > 0$). Throughout this paper, software quality denotes security quality, not functionality. Let q be the security quality of the software product

where $q \in [0, 1]$. Security quality measures vulnerability of the software to attacks at the product launch. Bug-free software can be considered to be of perfect security quality, given a certain level of functionality (V). Following the argument of security experts that the initial quality of the software at the product launch matters, and that the availability of patching mechanism may worsen the situation, we focus on the initial quality in our model. Let $K(q)$ be the expected loss in the product life-span when q-quality software is installed under non-targeted attacks that exploit the vulnerabilities of the software due to imperfect code. Under the proposed risk-sharing mechanism, the vendor takes some proportion of the risk, denoted by r where $r \in [0, 1]$. If any attack exploiting the vulnerability of the installed software is successful and incurs loss, then the vendor shares the responsibility with its customers. Thus, the expected benefit of a customer who purchases the software with price p is

$$E(U) = \theta[V - (1-r)K(q)] - p.$$

$K(q)$ in our model is based on a certain period of time. It is reasonable since most software products are licensed to the corporate customers. Thus, the life-span of the software is considered to be the licensing period. We assume that $K'(q) < 0$ and $K''(q) > 0$, so that the expected loss decreases as the quality level increases at a diminishing rate. θ captures customer heterogeneity indicating how much utility a customer derives from the software's functionality. The same attack may cause more severe damage to some firms than others. If θ is high, the customer is more sensitive to security features of the product, in that she enjoys more utility from the product, but also suffers more disutility from a successful attack. It holds in reality that some firms are more sensitive to security than others. For example, banks may be such customers with high θ. We assume that θ is uniformly distributed on [0, 1]. Customers who have expected utility greater than zero buy the software, whereas others do not.

2.2.2 Vendor's Profit Function

The software vendor's expected profit is

$$E(\pi(p,q,r)) = D(p,q,r)(p - rK(q)) - C(q)$$

where $D(p, q, r)$ is the demand for the product, p is the price and $C(q)$ represents the initial cost for producing software with quality level q. Production of information goods such as software involves high initial cost but low marginal cost. In other words, the cost of producing the original copy is substantial whereas the cost of producing additional copies is negligible. According to Varian (2001), this cost structure of constant fixed cost and zero marginal cost is common assumption for information goods, but is rarely observed for physical products since there are capacity constraints in nearly every production process. As a result, given the context of software product, the cost does not depend on quantity, that is, the marginal cost of production is zero. We assume a convex cost function, that is, $C'(q) > 0$ and $C''(q) > 0$, so that the cost increases as the quality level rises at a growing rate. $rK(q)$ is the expected loss, for which the vendor is responsible per unit of the product. Although the variable cost of production is assumed to be zero, $rK(q)$ plays a role of the variable cost in our model. Table 1 summarizes the notations used in the baseline model.

q	security quality
$K(q)$	expected loss when q-quality software is installed
$C(q)$	fixed cost to produce software with security quality q
r	proportion of the risk for which the software vendor is responsible
p	software price
θ	customer taste

V	Functionality
$D(p,q,r)$	demand for software

Table 1. Notations Used in the Baseline Model

2.3 Risk-Sharing Mechanisms: Incentives

2.3.1 Market Outcome under Monopoly

The monopoly case is quite relevant to software industry. Consider the case of Microsoft, which dominates the PC operating systems market. Reflecting reality, we set up a model in the following way. First, the software vendor decides the quality level q and the risk-sharing level r, simultaneously, and then, it sets up the price p for the software. Finally, the customers decide whether or not to buy the software based on price, quality and the vendor's risk-sharing level. Demand for the software product offered by a monopolist is derived from the expected utility of the customers. The demand is

$$D(p,q,r) = 1 - \frac{p}{V - (1-r)K(q)}.$$

Then the expected profit for the monopolist becomes

$$\begin{aligned} E(\pi(p,q,r)) &= D(p,q,r)(p - rK(q)) - C(q) \\ &= \left(1 - \frac{p}{V - (1-r)K(q)}\right)(p - rK(q)) - C(q). \end{aligned}$$

The first-order condition for p is

$$\frac{\partial E(\pi(p,q,r))}{\partial p} = 1 - \frac{2p}{V - (1-r)K(q)} + \frac{rK(q)}{V - (1-r)K(q)} = 0$$

and the second-order condition for p is

$$\frac{\partial^2 E(\pi(p,q,r))}{\partial p^2} = -\frac{2}{V-(1-r)K(q)} < 0.$$

Thus, solving for the first-order condition for *p* yields the optimal price that maximizes the vendor's profit as follows:

$$p^* = \frac{V-(1-2r)K(q)}{2}.$$

Substituting p^* in the software vendor's expected profit function leads to

$$E(\pi(q,r)) = \frac{(V-K(q))^2}{4(V-(1-r)K(q))} - C(q).$$

Note that $r^* = \arg\max_r \pi(q,r) = 0$ since $(V-K(q))^2 > 0$ and $K(q) > 0$.

The first-order condition for *q* is

$$\frac{\partial E(\pi(q,r))}{\partial q} = -\frac{K'(q)(V-K(q))}{2(V-(1-r)K(q))} + \frac{(1-r)K'(q)(V-K(q))^2}{4(V-(1-r)K(q))^2} - C'(q) = 0.$$

Let *M* in the subscript represent the monopolist software vendor. At equilibrium, we have $r_M^* = 0$ and $\frac{K'(q_M^*)}{C'(q_M^*)} = -4$. The ratio can be interpreted as the cost-effectiveness of security quality improvement in terms of reducing the risk. The larger ratio implies that increasing the same level of security quality of software prevents more loss at lower cost. The results imply that the monopolist has no incentive to voluntarily share any risk although the risk-sharing mechanism is adopted. Since one vendor dominates the market in this scenario, it may be satisfied with the current state of the market where the entire risk is on the customer side. Without sharing any risk with its customers, the software vendor can maximize its profit when it dominates the market. For the monopolist that already enjoys all the benefits in the monopoly market, risk-sharing only hurts its profit. The optimal security quality is chosen when the

marginal benefit increase from lowering loss due to better security is four times the marginal cost increase on security quality. This implies that the monopolist software vendor maximizes its profit when it chooses the security quality at which it enjoys loss decrease four times more than cost increase. Based on the functional form, we further analyze the quality choice of the monopolist in the next section.

2.3.2 Social Planner's Solution

A social planner is one whose priority is to maximize the benefit of the entire society, not like the vendor who pursues profit. In reality, it can be a software vendor that tries to maximize customers' benefit while it maximizes its own benefit. Open source software vendors can be such examples in the sense that they attempt to maximize user benefit and their own benefit, which is not necessarily monetary profit. The benefit of the entire society in the software market means the sum of all the customers' benefit and the software vendor's profit. In the economics literature, the social benefit is maximized when the price is set up at the marginal cost, which is $rK(q)$ in our model. Hence, the social benefit can be written as

$$S(q,r) = \int_{\theta^*}^{1} \theta[V-(1-r)K(q)]d\theta - (1-\theta^*)rK(q) - C(q) \quad \text{where} \quad \theta^* = \frac{rK(q)}{V-(1-r)K(q)}.$$

It simplifies to

$$S(q,r) = \frac{(V-K(q))^2}{2(V-(1-r)K(q))} - C(q).$$

Proof.

Let $X = V-(1-r)K(q)$ and $Y = rK(q)$. Then

$$S(q,r)=\int_{\frac{Y}{X}}^{1}\theta X d\theta-\left(1-\frac{Y}{X}\right)Y-C(q)=\frac{(X-Y)^2}{2X}-C(q)=\frac{(V-K(q))^2}{2(V-(1-r)K(q))}-C(q).$$

This completes proof. ***QED.***

Note that $r^*=\arg\max_r \pi(q,r)=0$ since $(V-K(q))^2>0$ and $K(q)>0$.

The first-order condition for q is

$$\frac{\partial S(q,r)}{\partial q}=-\frac{K'(q)(V-K(q))}{(V-(1-r)K(q))}+\frac{(1-r)K'(q)(V-K(q))^2}{2(V-(1-r)K(q))^2}-C'(q)=0.$$

Let *SP* in the subscript denote the social planner in the monopoly market. At equilibrium, we have $r_{SP}^*=0$ and $\frac{K'(q_{SP}^*)}{C'(q_{SP}^*)}=-2$. We find that the neither the social planner nor the software vendor under monopoly wants to share any risk with the customers. This implies that sharing risk with customers may be less cost-efficient than increasing software security to achieve their goals: maximizing profit or social welfare. We include the social planner case to compare the monopolist software vendor's security quality with the socially desirable level of security quality. Comparison of the optimal quality levels by the software vendor and the social planner leads to the following proposition.

Proposition 1: *In a software market with a risk-sharing mechanism, neither the monopolist nor the social planner is willing to share any risk. However, the social planner offers a higher-quality product than the monopolist.*

Proof.

Recall that $\frac{K'(q_M^*)}{C'(q_M^*)}=-4$ and $\frac{K'(q_{SP}^*)}{C'(q_{SP}^*)}=-2$. Denote $F=\frac{K'(q)}{C'(q)}$. Showing that F is an increasing function of q completes the proof.

$$\frac{\partial F}{\partial q} = \frac{K''(q)}{C'(q)} - \frac{K'(q)}{C'(q)^2} C''(q) = \frac{K'(q)}{C'(q)} \left(\frac{K''(q)}{K'(q)} - \frac{C''(q)}{C'(q)} \right).$$

Note that $C'(q) > 0, C''(q) > 0, K'(q) < 0$ and $K''(q) > 0$ by assumption. Thus,

$$\frac{K'(q)}{C'(q)} < 0 \text{ and } \frac{K''(q)}{K'(q)} < 0 < \frac{C''(q)}{C'(q)}.$$

Thus, we have $\frac{\partial F}{\partial q} > 0$, which implies that F is an increasing function of q. Therefore, $q_{SP}^* > q_M^*$. ***QED.***

This is interesting in the sense that neither the social planner nor the monopolist has any incentive to bear the risk. While it seems surprising that the social planner will not share any risk, this is actually a result of optimal resource allocation. The socially optimal outcome is arrived at when the social planner allocates all resources to develop better quality rather than to share risk. Recall that the cost of producing additional copies of information goods such as software is negligible. In other words, the marginal cost of production is near zero. However when the software vendor shares any risk with its customers, it faces the expected loss for each copy sold. Thus, although the marginal cost of production is zero, the cost of selling additional copy is crucial in the case of risk-sharing. Note that sharing no risk allows the social planner to face zero marginal cost and to cover the entire market. The social planner is left with no resource to share the loss when it serves the entire market by offering a price at marginal cost. Thus, it turns out that even the social planner does not want to share any risk. Interestingly, we find that the factors that increase the risk-sharing level do not always result in higher quality. Proposition 1 provides evidence of under-provided quality of software under monopoly, as has been observed in the market. Figure 1 illustrates the relationship between the quality of a monopolist and a social planner.

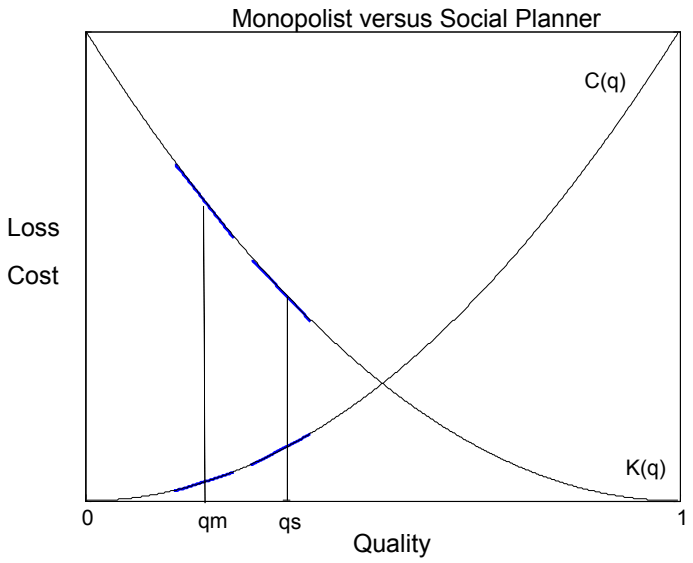


Figure 1. Equilibrium Quality of the Monopolist and the Social Planner

We next examine the incentive to share risk when there is competition in the market.

2.3.3 Competition

Incumbent and Entrant with the Same Quality Software Product

We first study the case of a duopoly market where a new vendor brings the same-quality software product as the established vendor. In this scenario, there are an established monopolist vendor that has no incentive to share the risk and a new vendor that enters the market bringing the same-quality software product as the incumbent in terms of functionality and security quality. We call the established vendor an *incumbent* and the new vendor an *entrant*. We investigate whether the entrant has an incentive to share any risk to be competitive against the incumbent when it is not easy for the new vendor to produce better-quality software due to various reasons such as technological barrier. We are also interested in the level of risk-sharing offered by the entrant if any. In this case, the entrant chooses its optimal risk-sharing level first. Then both the incumbent and the entrant set up their own prices simultaneously. Finally, customers decide

whether to buy from the incumbent or the entrant or neither.

Let *I* in the subscript denote the incumbent and *E* represent the entrant. Table 2 shows the notations used in the model for the incumbent and the entrant scenario. Other parameters stay the same as in the baseline model explained in Table 1.

$\bar{q}$	security quality provided by both the incumbent and the entrant
$K(\bar{q})$	expected loss when software is installed
$C(\bar{q})$	fixed cost to produce software
p_I	price for incumbent's software product
p_E	price for entrant's software product
v_I	total value offered to a customer by incumbent's software $(=V-K(\bar{q}))$
v_E	total value offered to a customer by incumbent's software $(=V-(1-r)K(\bar{q}))$

Table 2. Notations Used in the Competition Model

The expected utility offered by the incumbent is

$$E(U_I)=\theta[V-K(\bar{q})]-p_I.$$

Similarly, the expected utility offered by the entrant is

$$E(U_E)=\theta[V-(1-r)K(\bar{q})]-p_E.$$

In order to derive the demand function for both vendors, we presume that customers can choose buying from the incumbent, the entrant or neither. Recall that *v* is the total value offered to the customer. That is, $v_I=V-K(\bar{q})$ and $v_E=V-(1-r)K(\bar{q})$. *v* is the net benefit that the software provides, which is, the software functionality that a customer enjoys minus the

customer-side risk. Since customers are heterogeneous in their valuation and loss, they perceive the value differently as captured in our taste parameter (θ). The customers will buy from the entrant when $\theta v_E - p_E \geq \theta v_I - p_I$ and they will buy from the incumbent when $\theta v_I - p_I > \theta v_E - p_E$ and $\theta v_I - p_I > 0$.

The demands for both firms can be derived from the above conditions.

$$D_E = 1 - \frac{p_E - p_I}{v_E - v_I} \text{ and } D_I = \frac{p_E - p_I}{v_E - v_I} - \frac{p_I}{v_I}.$$

The expected profits are

$$E(\pi_E) = \left(1 - \frac{p_E - p_I}{v_E - v_I}\right)(p_E - rK(\bar{q})) - C(\bar{q}) \text{ and } E(\pi_I) = \left(\frac{p_E - p_I}{v_E - v_I} - \frac{p_I}{v_I}\right)p_I - C(\bar{q}).$$

The first order conditions for p_E and p_I are

$$\frac{\partial E(\pi_E)}{\partial p_E} = 2 - \frac{2p_E}{v_E - v_I} + \frac{p_I}{v_E - v_I} = 0 \text{ and } \frac{\partial E(\pi_I)}{\partial p_I} = \frac{p_E}{v_E - v_I} - \frac{2p_I}{v_E - v_I} - \frac{2p_I}{v_I} = 0.$$

Solving the first order conditions p_E and p_I gives the optimal prices charged by both vendors:

$$p_E^* = \frac{4v_E(v_E - v_I)}{4v_E - v_I} \text{ and } p_I^* = \frac{2v_I(v_E - v_I)}{4v_E - v_I}.$$

Substituting p_E^* and p_I^* in the expected profit functions yields

$$\begin{aligned} E(\pi_E(r)) &= \left(1 - \frac{p_E^* - p_I^*}{v_E - v_I}\right)(p_E^* - rK(\bar{q})) - C(\bar{q}) \\ &= (V - K(\bar{q}))^2 \frac{rK(\bar{q})}{(3V - 3K(\bar{q}) + 4rK(\bar{q}))^2} - C(\bar{q}). \end{aligned}$$

The first order condition for *r* is

$$\frac{\partial E(\pi_E(r))}{\partial r} = (V - K(\bar{q}))^2 \frac{K(\bar{q})}{(3V - 3K(\bar{q}) + 4rK(\bar{q}))^2} - (V - K(\bar{q}))^2 \frac{8rK(\bar{q})^2}{(3V - 3K(\bar{q}) + 4rK(\bar{q}))^3} = 0.$$

Arranging the first order condition shown above leads to the optimal risk-sharing level:

$$r^* = \frac{3(V - K(\bar{q}))}{4K(\bar{q})}.$$

Note that

$$\frac{\partial r^*}{\partial \bar{q}} = -\frac{3K'(\bar{q})}{4K(\bar{q})} - \frac{3(V - K(\bar{q}))}{4K(\bar{q})^2}K'(\bar{q}) = -\frac{3K'(\bar{q})}{4K(\bar{q})^2}V.$$

Since $V > 0, K(\bar{q})^2 > 0,$ and $K'(\bar{q}) < 0,$ we have $\frac{\partial r^*}{\partial \bar{q}} > 0.$ The analysis that we went through leads to the following proposition.

Proposition 2: *In the presence of competition, the entrant offering the same quality software as the incumbent sharing no risk has an incentive to introduce positive risk-sharing to alleviate competition. Moreover, as the quality level increases, the risk-sharing level also increases.*

In contrast to the monopolist and the social planner, the entrant in this scenario has an incentive to share the risk with its customers. When the market is dominated by a single software vendor, customers may have to be more content with the product since they have only two options: buy the software from the monopolist or not buy any. But if competition comes into play, customers face more options, implying that they can choose from either the incumbent or the entrant. In order to compete against an established incumbent vendor, the entrant may need to give more value to the customers to attract them. Our results show that the entrant has an incentive to share positive risk with the customers as a way to survive the competition against an incumbent when both provide the same quality software product. It follows that the risk-sharing level increases with the quality level. This result is quite interesting in the sense that without the risk-sharing mechanism, the entrant may have less incentive to enter the market because its entry may trigger intensive price competition. However, with the risk-sharing mechanism, the entrant

can use risk-sharing to alleviate the price competition and to be competitive against the incumbent. Interestingly, we find that the entrant's optimal risk-sharing level increases as the quality level increases. Thus, customers may use the entrant's risk-sharing level as a proxy of the quality level given in the market.

In reality, it is not easy for the customers to evaluate quality before using it, especially when the product is technology-intensive. Customers realize the quality of such products after using them. Thus, customers use a proxy for the product quality in many cases. For example, price can be a good proxy in the sense that customers expect more expensive product to have higher quality. The brand name of the vendor is another possibility. It is often hard for the vendor with weak reputation to make customers appreciate its high product quality. Our results show that a risk-sharing mechanism gives an opportunity to such vendors since customers evaluate the quality by observing the risk-sharing level. We can easily find an example in the automobile industry. When KIA entered the U.S. automobile market, it suffered from weak reputation and customers did not trust the quality of KIA cars. By adopting a 10-year warranty, KIA was able to project the quality of its products by showing its confidence. The 10-year warranty turned out to be an effective strategy in terms of increasing reputation as well as market share.

Duopoly Competition with Same Quality but Different Risk-Sharing

This scenario serves as an extended case to the previous one. In this scenario, two vendors compete against each other with products of the same quality and they are allowed to differentiate their products by offering different levels of risk-sharing. That is, one vendor shares higher risk with its customers than the other. In this scenario, both vendors decide the risk-sharing level given a certain level of quality. Then, they choose the optimal prices. Finally,

customers make their purchasing decisions. We label the vendor sharing high risk, hence offering high value to customers, as an *H* vendor and denote the other vendor sharing low risk, hence offering low value to customers, as an *L* vendor. Then the total value offered to the customer is

$$v_H = V - (1 - r_H)K(\bar{q}) \quad \text{and} \quad v_L = V - (1 - r_L)K(\bar{q}) \quad \text{where} \quad r_H \geq r_L.$$

Case 1: Both vendors share the same level of risk ($r_H = r_L$).

In this case, both vendors face the same optimization problem. Let *S* in the subscript denote this symmetric case. The expected profit for each vendor becomes

$$E(\pi_S) = \frac{1}{2}\left(1 - \frac{p}{V - (1-r)K(\bar{q})}\right)(p - rK(\bar{q})) - C(\bar{q}).$$

It can be readily verified that

$$p_S^* = \frac{V - (1-2r)K(\bar{q})}{2}.$$

Substituting p^* in the expected profit function yields

$$E(\pi_S) = \frac{(V - K(\bar{q}))^2}{8(V - (1-r)K(\bar{q}))} - C(\bar{q}).$$

Thus, $r^* = \arg\max E(\pi(r)) = 0.$ When two vendors share the same level of risk, no risk-sharing is optimal for both vendors. The profit for each vendor is as follows:

$$E(\pi_S) = \frac{(V - K(\bar{q}))}{8} - C(\bar{q}).$$

Case 2: Both vendors offer different levels of risk-sharing ($r_H > r_L$).

Following the same logic as in the incumbent and the entrant scenario leads to the following expected profit functions:

$$E(\pi_H) = \left(1 - \frac{p_H - p_L}{v_H - v_L}\right)(p_H - r_H K(\bar{q})) - C(\bar{q})$$
$$E(\pi_L) = \left(\frac{p_H - p_L}{v_H - v_L} - \frac{p_L}{v_L}\right)(p_L - r_L K(\bar{q})) - C(\bar{q}).$$

It can be verified that the interior solution for the risk-sharing level does not exist.[1] Thus, we examine the case where the solution is a boundary one. Since we assume that $r_H > r_L$ in this scenario, r_L^* cannot be 1. Therefore, one last possibility is $r_L^* = \arg\max E(\pi_L) = 0$. Then we find the best response of the high-type vendor.

Substituting r_L^* in $\frac{\partial E(\pi_H)}{\partial r_H}$ yields

$$r_H^* = \frac{3(V - K(\bar{q}))}{4K(\bar{q})}.$$

Then the high-value vendor and the low-value vendor expect the following profit:

$$E(\pi_H) = \frac{(V - K(\bar{q}))}{48} - C(\bar{q})$$
$$E(\pi_L) = \frac{7(V - K(\bar{q}))}{48} - C(\bar{q})$$

Note that the optimal strategy for the low-value vendor is always sharing no risk no matter whether the competing vendor wants to differentiate its risk-sharing level from the low-value vendor so that it can become the high-value vendor or not. In other words, when a software vendor perceives that the competing vendor is capable of producing same quality software product and sharing higher risk with its customers, the software vendor is better off sharing no risk with its customers. The high-value vendor shares no risk when it is allowed to offer the same

[1] Optimal prices for both the high-value and the low-value vendors can be obtained. Plugging p_H^* and p_L^* in the expected profit functions lead to the functions of r_H and r_L. No closed form solutions for r_H and r_L are obtainable to maximize $E\pi_H(r_H)$ and $E\pi_L(r_L)$.

level of risk-sharing as the low-value vendor. When both vendors differentiate their risk-sharing levels, the high-value vendor shares a positive level of risk. The incentive for the high-value vendor to offer positive risk-sharing is decided by the profit comparison. Note that

$$E(\pi_S) = \frac{(V - K(\overline{q}))}{8} - C(\overline{q}) \quad > \quad E(\pi_H) = \frac{(V - K(\overline{q}))}{48} - C(\overline{q}).$$

Since the high-value vendor makes higher profit in Case 1 than in Case 2, the optimal strategy is sharing no risk with the customers as the low-value vendor does. The following Proposition summarizes the results.

Proposition 3: *In the presence of competition where two vendors offer same-quality software, sharing no risk is the optimal choice for both firms* $(r_H^* = r_L^* = 0)$.

Proposition 3 shows that risk-sharing may not act as a product differentiator in the software industry. When we examine the case where a new vendor enters a market with an established vendor, we find that the new vendor has an incentive to share the risk to compete with the established vendor that shares no risk, hence achieving higher profit. The result implies that there exists a first-mover advantage in the software market. In other words, a vendor gains advantage by being first to the software market with a new product or service. In this section, we examine the case where both vendors make decision simultaneously. We find that neither the high-value vendor nor the low-value vendor wants to share any risk. Although customers may be better off with a risk-sharing mechanism, the vendors do not want to share any risk even in this competition scenario. This result is consistent with the monopoly case in the sense that risk-sharing is a risky choice for the software vendors since it imposes high cost on them but customers do not appreciate it much. So, unless a software vendor has high incentive to differentiate its software product from its competitor as discussed in the incumbent and the

entrant scenario, software vendors do not have any incentive to voluntarily share risk with their customers.

The problem of the software market is that vendors do not want to produce software with high quality. We find that the vendors will not voluntarily share any risk with customers unlike many other industries. One of the factors that differentiate information goods from other physical goods is the cost structure. Production of information goods such as software involves high initial cost such as innovation cost but low marginal cost. That is, the fixed cost such as investment on innovation is substantial while the cost of making additional copies is insignificant. Our model captures the fixed cost, that is, the innovation cost for the software with quality level q by $C(q)$ and we assume zero marginal cost of production. This kind of cost structure might lead to results that the software vendors do not want to share risk with their customers unlike other industries for physical goods such as the automobile industry and electronics industry. In the software industry, vendors perceive the fixed cost $C(q)$ as the main cost factor while they do not take the marginal cost seriously. However, risk-sharing with each of the customers incurs significant expected cost $rK(q)$ for each of the copies that the software vendor sells. In the presence of risk-sharing, software vendors face both substantial fixed cost and marginal cost, which is not common for information goods. Thus, sharing risk with customers is not a pleasant business to the software vendor that already bears high innovation cost especially when the software vendor dominates the market. In the next section, we examine the policy-oriented approach to this problem. We investigate whether a certain type of government policy such as regulation on risk-sharing and subsidy to the customers is an effective way to solve the problem of underprovided software security.

2.4 Policy Implications

In the monopoly scenario, we show that neither a monopolist software vendor nor a social planner would share risk when it is voluntary and that the quality offered by a monopolist is much lower than the socially optimal level. This finding is consistent with the economics literature in that product characteristics such as product quality are not optimally set under regimes of monopoly. For example, Spence (1975) argues that unregulated monopolist's selection of product characteristics is likely to be biased away from the social optimum, which turns out to be true in the software market according to our findings. As a possible solution to the problem of this underprovided software security, legal liability on software vendors has been discussed among computer scientists, software engineers, and security experts. Fisk (2002) argues that only legal and economic systems such as liability have the potential to change the status of poor network security. Ryan (2003) also argues that software vendors should be liable for any defects of their products under government regulation. Schneier (2006) discusses the right form of federal regulation and argues that imposing liability on the software vendors is more effective than requiring individual users to secure themselves. Other policy-oriented approaches to improve software security include government subsidy to software users, which is controversial at present. The logic behind this fact is that the software users prefer functionality to security in the current state. However, if we allow users to have higher budget, they may be more willing to pay for security. Academic researchers have found that under certain circumstances, a subsidy policy is effective. For example, August and Tunca (2006) argue that a subsidy on patching is desirable when security risk and patching cost are high.

In this section, we first examine the effectiveness of one form of government regulation, which is regulation on the risk-sharing level. That is, when loss occurs from a cyberattack which

exploits a vulnerability of the software, the software vendor is forced to be responsible for a certain proportion of the loss. We investigate whether such regulation on risk-sharing gives a monopolist software vendor any incentive to improve security quality of its software product. Then we discuss the effectiveness of government subsidy in the context of the software market. We investigate whether subsidizing software users in order to give them incentives to spend more for security is an effective policy in terms of improving security quality of the software product. We discuss the effectiveness of government subsidy in the context of the software market.

2.4.1 Regulation on Risk-Sharing

We first investigate whether the federal regulation on risk-sharing creates an incentive for the monopolist to increase quality. This is a policy that directly regulates the software vendor that dominates the market without any competitor. In order to derive more economically interpretable results and to illustrate our findings, we assume the expected loss and the cost to be quadratic functions of security quality. In that way, we still keep the convexity of the cost function and the convexity of the expected loss function, which we assumed for our baseline scenario. Quadratic functions for cost and utility are often used in the literature of information systems such as in Jones and Mendelson (1998), Sundararajan (2004) and Telang et al. (2004). Despite its less generalizability than the general convex function, the quadratic function is often used in economics, both theoretical and applied since it allows meaningful interpretability of the results without loss of too much generalizability. The expected loss function and the cost function are as follows:

$$K(q) = V(1-q)^2, \quad V > 0$$
$$C(q) = cq^2, \quad c > 0.$$

Note that V is the functionality of the software as defined earlier. Suppose that the government imposes regulation on risk-sharing. As discussed earlier, imposing liability on the software vendors under the legal system has been identified as an effective solution to the problem of underprovided security of software products. We examine the effectiveness of one form of such legal liability mechanisms, which is government regulation on the risk-sharing level. Under this mechanism, software vendors are forced to be responsible for a certain proportion of the loss which incurs due to the vulnerabilities of their software products. Let *r* be the risk-sharing level the government forces the software vendor to be responsible for. The expected profit for the monopolist is

$$E(\pi(q,r)) = \frac{(V-K(q))^2}{4(V-(1-r)K(q))} - C(q).$$

Given *r*, the first-order condition for *q* is

$$\frac{\partial E(\pi(q,r))}{\partial q} = -\frac{K'(q)(V-K(q))}{2(V-(1-r)K(q))} + \frac{(1-r)K'(q)(V-K(q))^2}{4(V-(1-r)K(q))^2} - C'(q) = 0.$$

Thus, we have

$$K'(q)(V-K(q))\{(V-K(q)) + r(V+K(q))\} + 4C'(q)(V-(1-r)K(q))^2 = 0$$
$$\Leftrightarrow V(q-1)(q-2)\{q(q-2) - r(1+(1-q)^2)\} + 4c\{1-(1-r)(1-q)^2\}^2 = 0.$$

Further derivation leads to the following Proposition.

Proposition 4: *When the government imposes risk-sharing between 0 and* $\bar{r}$ *where* $\bar{r} = \frac{1}{256}\left(\frac{V}{c}\right)^2\left(\frac{V}{c}+8\right)^2$*, the monopolist increases the quality of the product. As the security quality improvement is more cost-effective, that is, investing the same amount of money on security*

development prevents higher loss $\left(\frac{V}{c}\uparrow\right)$*, government can impose more liability on the software vendor to give an incentive for security quality improvement.*

Proof.

Note that the expected profit is maximized at q which satisfies

$$V(q-1)(q-2)\{q(q-2)-r(1+(1-q)^2)\}+4c\{1-(1-r)(1-q)^2\}^2=0.$$

Solving the above equation for r yields

$$r_1=\frac{(q-2)}{8c(q-1)^3}\left\{q^2(V+8c)-2q(V+4c)+2V-\sqrt{V^2q^4-4V^2q^3+8V(V+2c)q^2-8V(V+2c)q+4V^2}\right\}$$

$$r_2=\frac{(q-2)}{8c(q-1)^3}\left\{q^2(V+8c)-2q(V+4c)+2V+\sqrt{V^2q^4-4V^2q^3+8V(V+2c)q^2-8V(V+2c)q+4V^2}\right\}$$

Denote q_M be the monopolist's equilibrium quality when there is no regulation. Then

$$r_1(q_M)=0$$
$$\Leftrightarrow q_M^2(V+8c)-2q_M(V+4c)+2V=\sqrt{V^2q_M^4-4V^2q_M^3+8V(V+2c)q_M^2-8V(V+2c)q_M+4V^2}.$$

Thus, $r_2(q_M)=\frac{(q_M-2)}{4c(q_M-1)^3}\left\{q_M^2(V+8c)-2q_M(V+4c)+2V\right\}$.

Solving for the optimal quality of the monopolist yields

$$\frac{K'(q_M)}{C'(q_M)}=-4\Leftrightarrow q_M=\frac{V}{V+4c}.$$

Thus,

$$r_2(q_M)=\frac{\left(\frac{V}{V+4c}-1\right)}{4c\left(\frac{V}{V+4c}-1\right)^3}\left\{\left(\frac{V}{V+4c}\right)^2(V+8c)-2\left(\frac{V}{V+4c}\right)(V+4c)+2V\right\}$$
$$=\frac{1}{256}\left(\frac{V}{c}\right)^2\left(\frac{V}{c}+8\right)^2.$$

Let $x=\frac{V}{c}$, then $\bar{r}=\frac{x^2(x+8)^2}{256}$. Thus we have

$$\frac{\partial \bar{r}}{\partial x} = \frac{1}{64} x(x+4)(x+8) > 0.$$

Thus, $\bar{r}$ increases as $\frac{V}{c}$ increases. ***QED.***

The results show the optimal level of regulation on risk-sharing that creates an incentive for the monopolist to increase quality. Intuitively, as the proportion of the expected loss to the cost for security quality development becomes higher, the admissible range of risk-sharing levels that lead to higher quality becomes wider. In other words, when the security development is more cost-effective, that is, investing the same amount of money on the security development brings higher benefit in terms of preventing higher loss, it is desirable to impose higher liability on the software vendor so that it can improve the security level of its software product. Consistent with what has been argued by security experts (Fisk 2002, Ryan 2003, Schneier 2006), our findings support that software liability as a regulation under the legal system is an effective incentive mechanism for software security improvement. However, imposing too much risk-sharing on the monopolist may make the situation worse by making the monopolist reduce security quality. This may discourage the software vendor by incurring too much cost.

Our findings may give policy makers a starting point when they want to regulate the monopolized software market. In the software market that we study throughout the paper, the software vendor's concern is security quality development for the software product for which functionality development is already done. For example, in early 2002, Microsoft focused on security development without adding new features to its Windows product. Our results may apply to the software market for such software products. There is a debate going on the issue of legal liability. An opposite view is that regulating software industry may impede innovation (Heckman 2003). Figure 2 illustrates how quality changes as the government imposes different

levels of risk-sharing on the monopolist. It has a policy implication that regulation on risk-sharing may be a good way for quality improvement but imposing too much risk-sharing on the monopolist may make the situation even worse.

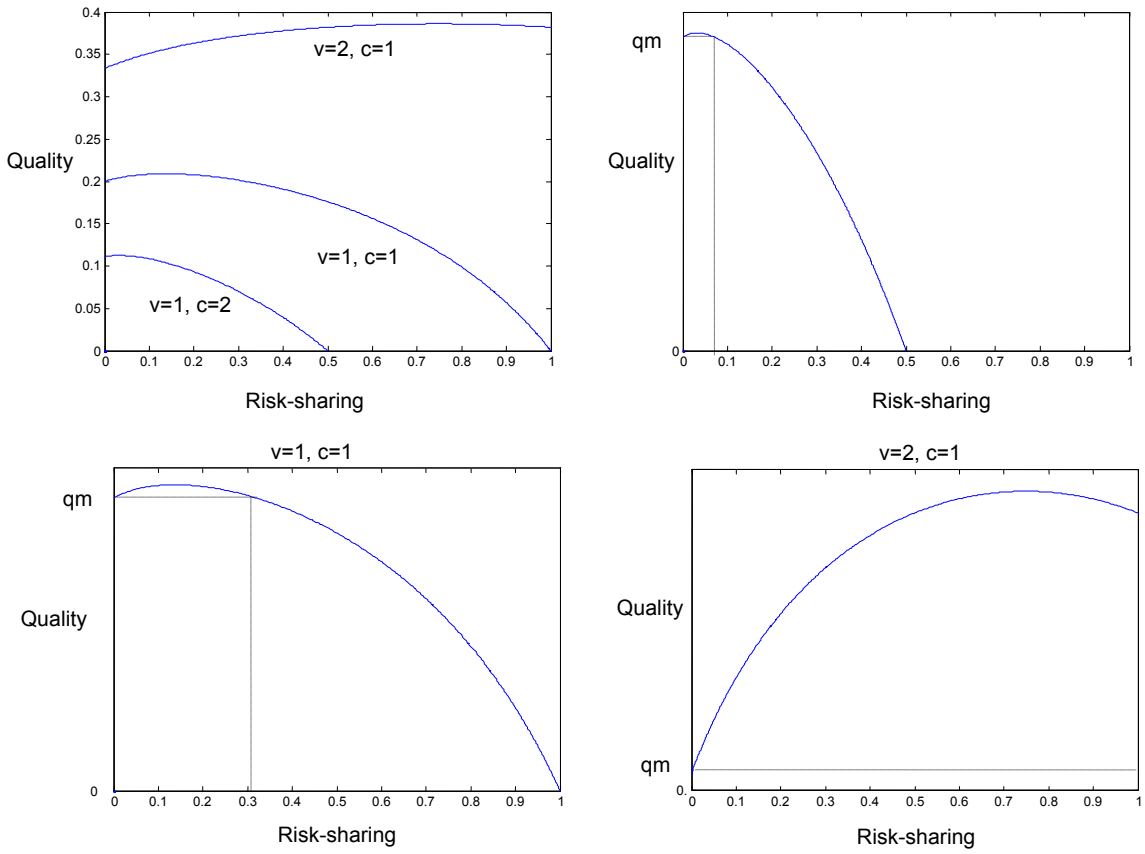


Figure 2. Monopoly under Regulation

We next examine government subsidy policy.

2.4.2 Subsidy for Customers

Proposals for government action being discussed by policy makers and computer security experts include offering tax incentives to businesses for spending on security (Harmon 2003). In the current state of the software market, software users care more about functionality of the software than security quality. Government's subsidy to customers has been proposed as a possible

solution to the problem of poor software security. However, some experts do not agree upon this proposal. For example, Orszag (2004) argues that providing a tax subsidy to firms for security costs would not be compelling enough to encourage the desirable level of security. In this section, we examine how the government subsidy to customers for spending security affects the quality level of the software under monopoly. Consider a scenario under which customers get subsidy such as tax incentives when they buy the software product. Let s be the subsidy for each customer who makes a purchase of the software. Then the expected utility of the customer is

$$E(U) = \theta[V - (1-r)K(q)] - p + s.$$

The demand derived from the above expected utility is

$$D(p,q,r) = 1 - \frac{p-s}{V-(1-r)K(q)}.$$

A monopolist vendor's expected profit is then

$$\begin{aligned} E(\pi(p,q,r)) &= D(p,q,r)(p - rK(q)) - C(q) \\ &= \left(1 - \frac{p-s}{V-(1-r)K(q)}\right)(p - rK(q)) - C(q). \end{aligned}$$

At equilibrium, we have $r^* = 0$ and

$$\frac{C'(q^*)}{K'(q^*)} = -\frac{1}{4} + \frac{s^2}{4(V-K(q^*))^2}.$$

Analyzing the above condition leads to the following proposition.

Proposition 5: *In the software market, the monopolist reduces the security quality of its software product when the government subsidizes the customers. In terms of quality improvement, government's subsidizing policy makes the problem worse in the monopoly case.*

Proof.

Let q_1 be the optimal quality of the monopolist without subsidy and q_2 be the optimal

quality of the monopolist with subsidy. Then we have

$$\frac{C'(q_1)}{K'(q_1)} = -\frac{1}{4} \quad \text{and} \quad \frac{C'(q_2)}{K'(q_2)} = -\frac{1}{4} + \frac{s^2}{4(V - K(q_2))^2}.$$

Thus,

$$\frac{C'(q_2)}{K'(q_2)} - \frac{C'(q_1)}{K'(q_1)} = \frac{s^2}{4(V - K(q_2))^2} > 0.$$

In the proof of proposition 1, we have shown that $\frac{K'(q)}{C'(q)}$ is increasing in q, that is, $\frac{C'(q)}{K'(q)}$ is decreasing in q. Therefore, $q_1 > q_2$. ***QED.***

Interestingly, we find that the government subsidy policy creates no incentive for the vendor to bear the risk and worse, it reduces the quality level. Low quality of available options in the software market has been identified as the main cause of poor security by security experts (National Research Council 1991). Although the subsidy policy may motivate customers to buy the software, it may make the situation worse because the problem is still on the vendor side. Our findings imply that subsidizing the customers may not work effectively. Rather, finding a way to penalize the vendor that produces low-quality products may be a more effective policy to make the software secure. Software liability under government control is one way to penalize such software vendors who have not been responsible for any loss due to the vulnerabilities of their low-quality products in the current software market. We consider government regulation on risk-sharing as a way to impose liability on the software vendor and examine the effectiveness of such mechanism under monopoly setting. Our results show that regulation on risk-sharing level as a way to penalize software vendors works better than the subsidy policy in terms of quality improvement. It implies that government should directly penalize the monopolist who produces low-quality product rather than giving more incentive to the customer for spending on security.

2.5 Discussion

2.5.1 Information Asymmetry

In reality, evaluating the likelihood of being attacked by hackers due to software vulnerability is not an easy task for software users while software vendors have some expertise of predicting such likelihood. In other words, there exists information asymmetry between the software vendor and its customers in the sense that customers may not be well informed about the possible loss due to software vulnerability. In the classical economic theory, Spence (1977) argues that imposing liability on the producer is an effective strategy to achieve socially optimal quality in the presence of information asymmetry, meaning that buyers are misinformed about the product characteristics such as quality. In this section, we investigate whether this economic theory holds in the context of software security. That is, we examine the feasibility and effectiveness of the risk-sharing mechanism when there exists information asymmetry between a monopolist software vendor and its customers. In section 2.3.1, we show that the risk-sharing mechanism is not feasible in the monopoly software market with perfect information. Our key question in this section is whether the misperception of customers about the probability of successful attack due to software vulnerability allows the viability of the risk-sharing mechanism in the monopoly software market. We also examine the monopolist software vendor's incentive for security development with respect to the level of information asymmetry.

Recall that $K(q)$ is the expected loss when the security quality of software is q. That is, $K(q) = P(q)L$ where $P(q)$ is the probability of successful attack and L is the loss which incurs in case of successful attack. Suppose that customers are not able to correctly expect the loss while the monopolist software vendor is. Let $\lambda(q)$ denote the customer's perception of the probability of successful attack $(= P(q))$ and $R(q)$ be the customer's perception of the expected

loss $(= K(q))$. That is, $R(q) = \lambda(q)L$. We consider this information asymmetry by modeling the discrepancy between $K(q)$ and $R(q)$. A in the subscript denotes the case of information asymmetry. When the risk-sharing mechanism is available in the presence of information asymmetry, a customer who buys the software enjoys utility:

$$E(U_A) = \theta(V - (1-r)R(q)) - p.$$

Then the demand for the software is $D_A(p,q,r) = 1 - \frac{p}{V-(1-r)R(q)}$. The expected profit for the monopolist becomes

$$\begin{aligned} E(\pi_A(p,q,r)) &= D_A(p,q,r)(p - rK(q)) - C(q) \\ &= \left(1 - \frac{p}{V-(1-r)R(q)}\right)(p - rK(q)) - C(q). \end{aligned}$$

The first-order condition for p is

$$\frac{\partial E(\pi_A(p,q,r))}{\partial p} = 1 - \frac{2p}{V-(1-r)R(q)} + \frac{rK(q)}{V-(1-r)R(q)} = 0.$$

and the second-order condition for p is

$$\frac{\partial^2 E(\pi_A(p,q,r))}{\partial p^2} = -\frac{2}{V-(1-r)R(q)} < 0.$$

Thus, solving for the first-order condition yields the optimal price:

$$p_A^* = \frac{V-(1-r)R(q) + rK(q)}{2}.$$

The expected profit at the optimal price is then

$$E(\pi_A(q,r)) = \left(1-\frac{1}{2}-\frac{rK(q)}{2(V-(1-r)R(q))}\right)\left(\frac{1}{2}(V-(1-r)R(q)+rK(q))-rK(q)\right)-C(q)$$
$$= \frac{1}{4}\left(1-\frac{rK(q)}{V-(1-r)R(q)}\right)(V-(1-r)R(q)-rK(q))-C(q)$$
$$= \frac{1}{4}\frac{(V-(1-r)R(q)-rK(q))^2}{V-(1-r)R(q)}-C(q).$$

We now investigate whether the monopolist software vendor has any incentive to share risk with its customers. Note that the profit maximization problem with respect to the risk-sharing level has a boundary solution since the second order condition is positive ($\frac{\partial^2 E(\pi_A(q,r))}{\partial r^2} > 0$). Comparison of the expected profits at the full and the zero levels of risk-sharing leads to conditions under which either boundary solution is optimal. The expected profits are $E(\pi_A(q,r=1)) = \frac{1}{4}\frac{(V-K(q))^2}{V} - C(q)$ and $E(\pi_A(q,r=0)) = \frac{1}{4}(V-R(q)) - C(q)$. Note that $E(\pi_A(q,r=1)) > E(\pi_A(q,r=0))$ when $R(q) > K(q)(2-\frac{K(q)}{V})$, implying that full risk-sharing is the optimal choice of the software vendor. Otherwise, $E(\pi_A(q,r=1)) < E(\pi_A(q,r=0))$. The results are summarized in Proposition 6.

Proposition 6: *In the presence of information asymmetry, the monopolist software vendor has incentive to fully share the risk with its customers if the customers overestimate the risk* $(R(q) > K(q)(2-\frac{K(q)}{V}))$. *If the customers' level of risk perception is not high enough* $(R(q) < K(q)(2-\frac{K(q)}{V}))$, *the monopolist software vendor does not want to share any risk.*

Proposition 6 implies that unlike the case of monopoly market with perfect information, the monopolist software vendor has incentive to share positive risk with its customers when the level of customers' perception of the risk is high. This is interesting because customers' misperception

of the risk allows the viability of the risk-sharing mechanism, which is not accepted by not only the monopolist software vendor but also the social planner in the monopoly market with perfect information. If we view this risk-sharing mechanism as a form of regulation though it is voluntary and market-driven, our finding is somewhat consistent with the classical economic theory in the sense that the monopoly market may need regulation to achieve social optimum. We also find that the monopolist software vendor does not want to share any risk when the level of customers' risk perception is not large enough. The result implies that the real world software vendors' unwillingness to share the risk with their customers may be explained by the customers under-appreciation of the risk associated with software security. Our finding justifies security experts' view that low level of software users' security awareness contributes to the poor software security (Siponen 2001).

2.5.2 Third Party Solution

Security experts and policy makers suggest software liability as one possible solution to the problem of poor software security. In this paper, we propose a risk-sharing contract as one way to impose liability on software vendors and investigate whether the voluntary and market-driven risk-sharing mechanism gives software vendors incentive to make their software products more secure. In the real-world software market, other ways to deal with the software security problem may exist. Examples include cyber insurance offered by the insurance companies and security software provided by third party software vendors.

In the current state, the market for cyber insurance is not yet mature compared to other insurance markets such as health insurance and auto insurance. Insurance companies can be a market lever to encourage sound security by setting up standards for best practices, applying

pressure on firms to reduce insurance premiums, and providing incentives for software companies to offer secure products. However, the victims of computer mishaps are reluctant to make their information public by reporting to a third party such as an insurance company. The insurance companies are still reluctant to get involved in the software industry due to uncertainty, lack of adequate statistics and technological changes, although some companies (such as Safeware, American Insurance Group, and Zurich) are offering policies ranging from hardware replacement to information-asset protection on a limited basis. Security software such as anti-virus software can be another example. Software users who care about security and do not trust the quality of the out-of-the-box software products often manage their system using such security software. By having such shield, users want to minimize the loss due to vulnerabilities of other software that they have on their system. Certainly, the security software based system protection is on the customers' own expense.

Though such third party solutions may have great potential as effective mechanisms to achieve better software security, the fundamental issue is that the risk associated with poor software security is entirely on the customer, not on the software vendor who creates the source of the problem. It has been intensely discussed for a decade that software vendors are the ones who can best understand the risk associated with their software products and best control the problem by providing better security quality at the introduction of their software products. Cyber insurance is fundamentally different from other insurances such as health and auto insurance in the sense that cyber risk is caused by the negligence of software vendors and it can be significantly improved if they want to do so. Regarding the automobile industry, most accidents do not occur due to the malfunction of automobiles. The logic of industry experts who champion software liability is simple in that poor software security is an incentive problem which can be

improved if we give software vendors such incentive for security development.

So far, we show that under certain conditions, the risk-sharing mechanism is effective in motivating software vendors' security improvement. Our focus in this section is to investigate whether the availability of such third party solutions gives the software vendor any incentive for security improvement. When third party solutions are available, the software vendor may still avoid being liable for any loss due to poor security of their products since they may keep providing poor security software and customers go find third party solution on their own expense. Though there are numerous factors we need to consider the impact of the third party solutions on the current regime, we aim to model the core factors to answer the key question.

Suppose that there exists a third party which provides customers a solution to security problem which occurs due to software vulnerability. For example, it can be an insurance company that offers full coverage or security software that prevents any loss due to vulnerability of software that the customer has. Certainly, there exists significant difference between the two examples. In this discussion section, we do not specifically model each case. Rather, we aim to capture the core impact of such third party solution on software security.

Denote T be the expense that accrues to a customer who adopts such third party solution. T can be the insurance premium or price of security software. We assume that such third parties observe the software market before they enter the market, and they have the ability or technology to offer their service or software product at the price below the expected loss at the optimal quality level offered by a monopolist software vendor without sharing any risk. If a customer would adopt a third party solution, the loss would be taken care of. Let q_M^* be the optimal security quality in a monopoly market without a third party solution. Our assumption implies that $T < K(q_M^*)$, which is realistic in the sense that the customers do not want any third party solution

and rather bear the loss if it incurs higher cost than the expected loss. We investigate whether the software vendor has an incentive to increase security quality when a third party solution is available. Consider a monopoly software market without software liability and with third party solutions.

T in the subscript represents the case where customers choose a third party solution and *NT* denotes the case where they do not adopt third party solution. Note that a customer enjoys utility from buying software and adopting third party solution as $E(U_T) = \theta(V - T) - p$. If a customer adopts software and bears the expected loss, then the utility becomes $E(U_{NT}) = \theta(V - K(q)) - p.$ A customer adopts the third party solution if $E(U_T) > E(U_{NT})$, that is, $T < K(q).$ Thus, the software vendor has two choices. First, the software vendor provides low security quality so that $T < K(q)$ and let customers get aid from third parties. Second, by offering high security quality such that $T > K(q),$ the software vendor makes the third party solution not attractive to the customers and let the customers bear the risk themselves.

Suppose that the software vendor wants the customer to use a third party solution. Then customers should face higher expected loss than the price for the third party solution $(T < K(q)).$ Thus the demand for the software is $D_T = 1 - \frac{p}{V - T}$. The expected profit for the monopolist software vendor becomes

$$E(\pi_T) = D_T p - C(q) = \left(1 - \frac{p}{V - T}\right) p - C(q).$$

The first-order condition for p is $\frac{\partial E(\pi_T)}{\partial p} = -\frac{p}{V - T} + 1 - \frac{p}{V - T} = 0.$ Thus, solving for the first

order condition yields the optimal price, $p_T^* = \frac{V-T}{2}$. Then the expected profit at the optimal price is $E(\pi_T) = \frac{V-T}{4} - C(q)$. Thus, $E(\pi_T)$ is maximized when $C(q)$ is minimized. Or, $q_T^* = 0$. Recall that $T < K(q_M^*)$ and $K'(q) < 0$. Thus, $T < K(q_M^*) < K(q_T^* = 0)$. Therefore, the software vendor's choice of zero security quality let the customers get help from third parties when they are available. In this case, the software vendor's expected profit is $E(\pi_T) = \frac{V-T}{4}$.

Consider the case where the software vendor does not want a third party to get involved and lets the customer bear all the risk. In this case, the software vendor provides high security quality so that the expected loss does not exceed the price for the third party solution ($T > K(q)$). Let q_{NT} be the security quality that the software vendor provides. Since $K(q_{NT}) < T < K(q_M^*)$ and $K'(q) < 0$, the software vendor should improve security quality from the optimal level in the monopoly market without any third party solution ($q_{NT} > q_M^*$). Then the demand for the software is $D_{NT} = 1 - \frac{p}{V - K(q)}$. The expected profit for the monopolist software vendor becomes

$$E(\pi_{NT}) = D_{NT}(q)p - C(q) = \left(1 - \frac{p}{V - K(q)}\right)p - C(q).$$

The first-order condition for p is $\frac{\partial E(\pi_{NT})}{\partial p} = -\frac{2p}{V - K(q)} + 1 = 0$. Thus, solving for the first order condition yields the optimal price, $p_T^* = \frac{V - K(q)}{2}$. Then the expected profit at the optimal price is $E(\pi_T) = \frac{V - K(q)}{4} - C(q)$. The first order condition is $\frac{\partial E(\pi_{NT})}{\partial q} = -\frac{K'(q)}{4} - C'(q) = 0$. The second order condition is $\frac{\partial^2 E(\pi_{NT})}{\partial q^2} = -\frac{K''(q)}{4} - C''(q) < 0$ since $K''(q) > 0$ and $C''(q) > 0$.

Note that q_M^* is the monopolist's optimal quality, meaning that $-\frac{K'(q_M^*)}{4} - C'(q_M^*) = 0$ and $E(\pi_T)$ is decreasing in q_{NT} when $q_{NT} > q_M^*$. Thus, at the limit, the optimal security quality is one such that q_{NT} such that $K(q_{NT}^*) = T$. Thus, the expected profit at the optimal price is $E(\pi_{NT}) = \frac{V - K(q_{NT}^*)}{4} - C(q_{NT}^*) = \frac{V - T}{4} - C(q_{NT}^*)$.

Note that $E(\pi_T) - E(\pi_{NT}) = \frac{V-T}{4} - \frac{V-T}{4} + C(q_{NT}^*) = C(q_{NT}^*) > 0$. The software vendor is better off having the third party solution in the market since it enjoys a higher profit when customers get aid from a third party. Thus, the optimal choice of the software vendor is not investing any on security development $q^* = q_T^* = 0$ and let customers manage their security with third party solutions.

Interestingly, the monopolist software vendor wants the third party to help customers manage their security problem which occurs due to the vulnerability of its software product so that the monopolist can avoid any cost for security improvement. If the industry for cyber insurance gets mature, this may solve the problem eventually in a way that the insurance helps customers who have software from the software vendor with poor reputation. Unfortunately, in the current state, the cyber insurance industry is quite new with a small number of providers. By modeling a static game, our model captures the current state of the software market with such third party security solutions. Consistent with the argument of security experts who champion software liability, our result shows that the software security does not get better in the presence of the third party solutions, with which the cost is still imposed on the customer side.

2.6 Conclusion

To enhance the poor state of network security, one needs to solve the fundamental problem: giving software vendors an incentive to increase the quality of their products. As a possible solution, software liability has been discussed for years among software engineers, jurists, and policy makers. In this paper, we propose a risk-sharing mechanism as a market-driven method to impose software liability, and analyze the software market with the proposed mechanism under various scenarios. We present an economic model of the software market, which considers the strategic interplay of risk-sharing and quality. We first compare the monopolist's and the social planner's solutions. Our results give evidence of under-provided software quality under monopoly, as has been observed in the market. We find that the social planner who maximizes social surplus offers higher-quality product than the monopolist and that neither the social planner nor the monopolist has any incentive to bear the risk. This is interesting in the sense that even for the social planner, sharing risk with the customers is not optimal at equilibrium although risk-sharing mechanisms such as warranties are widely used in other industries. This can be explained by the intuition that risk-sharing and quality are not strategic complements.

We extend the model to duopoly competition. In the case where the entrant brings a product of the same quality as the incumbent who does not want to share any risk, we find that the entrant has an incentive to introduce positive risk-sharing to alleviate competition and that the risk-sharing level increases as the quality level increases. However, in the presence of competition where two vendors can differentiate their products not by quality but by risk-sharing, neither the high-value vendor nor the low-value vendor has an incentive to share the risk with customers, implying that the government should get involved in the software market as a market regulator.

We examine how different forms of government policy affect software quality under monopoly. First, we investigate whether regulation on risk-sharing creates an incentive for the monopolist to increase quality. Our findings show that a certain level of regulation on risk-sharing leads to higher quality, and it becomes more effective as the expected loss becomes higher compared to the cost for quality improvement. Then, we analyze the software market where the government subsidizes customers who buy the software. Unlike the prediction of practitioners, we find that government subsidy policy creates no incentive for the monopolist to bear the risk and even reduces the quality level. This result implies that the government should adopt the policy to directly penalize the monopolist that produces low-quality software rather than subsidizing the customers.

We discuss other possibilities in the software market: information asymmetry and third party solutions. We examine the software market where there exists information asymmetry between the monopolist software vendor and its customers. We find that the monopolist software vendor has an incentive to share positive risk with its customers when the level of customers' risk perception is high. Otherwise, the monopolist software vendor does not want to share any risk with its customers. As pointed out by security experts, our result shows that a low level of customers' security awareness contributes to poor software security. We then examine the impact of third party solutions on software security and find that the software vendor does not want to spend on security development when third party security solutions are available to customers.

Software liability has been an important issue among software developers and policy makers. Nevertheless, no effective liability-imposing mechanism has been found yet. We propose a risk-sharing mechanism as a possible solution. Our research contributes to the literature in the

following ways. First, we provide an economic framework for analyzing a security issue where little research has dealt with the problem of the software market from an economic perspective although the possible solution to this problem is economic rather than technical. Second, our results suggest that a risk-sharing mechanism is promising under certain conditions as a form of market-driven regulation. Finally, our results have managerial implications for software vendors. Especially, a vendor who enters the market competing with an established incumbent may want to consider sharing the risk with its customers.

While significant, this study can be improved in several ways. First, developing a way to apply the risk-sharing mechanism may form a promising research area. For example, loss measurement and risk analysis are prerequisites for the proposed risk-sharing mechanism. Second, more rigorous analysis of the effectiveness of other alternatives to software liability such as cyber insurance may bring more insights. Throughout the paper, our focus is on the risk-sharing mechanism. As we mentioned earlier, security experts have suggested other ways to achieve better software security. In this paper, we discuss how the availability of such alternative solutions may affect the dynamics of the software market by only considering the core characteristics. Deeper understanding of the cyber insurance market and more realistic model set up will bring more meaningful outcomes. Third, considering a mechanism for patching and examining how the availability of patching affects the interplay of risk-sharing and software quality will be interesting. Patching is an emerging topic in the security literature and researchers examine the software vendor's choice of quality when patching is available (Arora et al. 2006). Viewing patching as a way to improve software security over time and modeling the software quality in a multi-period game will be an interesting research topic.

References

Anderson, R. Why Information Security is Hard-An Economic Perspective, *Proceedings of the 17th Annual Computer Security Applications Conference*, December 10-14, 2001, p. 358

Arora, A., Calkins, J. and Telang, R. Sell First, Fix Later: Impact of Patching on Software Quality. *Management Science*, 52:3, 2006, 465-471.

August, T., and Tunca, T. Network Software Security and User Incentives. *Management Science*, 52, 11 (2006) 1703-1720.

Brady, R., Anderson, R. and Ball, R. Murphy's law, the fitness of evolving species, and the limits of software reliability, Cambridge University Computer Laboratory Technical Report no. 476 (1999), at http;//www.cl.cam.ac.uk/~rja14

Economic Times. Microsoft Security Push Cost $100 Million, July 19, 2002.

Ernst & Young (www.ey.com). Global Information Security Survey, 2004.

Fisher, D. Contracts Getting Tough on Security. *eWeek*, April 15, 2002.

Fisk, M. Causes and Remedies for Social Acceptance of Network Insecurity. *Workshop on Economics and Information Security*, University of California, Berkeley, May 16-17, 2002.

Gordon, L. A., Loeb, M. P., Lucyshyn, W. and Richardson, R. 2004 CSI/FBI Computer Crime and Security Survey. Computer Security Institute, 2004.

Harmon, A. Digital Vandalism Spurs a Call for Oversight. *New York Times,* September 1, 2003, Section A, Page 1, Column 6.

Heckman, C. Two Views on Security Software Liability: Using the Right Legal Tools. *IEEE Security & Privacy* 1:1 (2003), 73-75.

Jones, R. and Mendelson, H. Product and Price Competition for Information Goods, Technical Report, Stanford University, 1998.

Markoff, J. Microsoft Programmers Hit the Books in a New Focus on Secure Software. *New York Times*, April 8, 2002.

Mussa, M. and Rosen, S. Monopoly and Product Quality. *Journal of Economic Theory* 18 (1978), 301-317.

National Research Council, Computers at Risk: Safe Computing in the Information Age, National Academy Press, Washington, DC, 1991

Orszag P. Homeland Security: The Problems with Providing Tax Incentives to Private Firms. Testimony before the House Committee on Small Business, Subcommittee on Rural Enterprise, Agriculture and Technology, Washington, DC July 21, 2004.

Ronnen, U. Minimum Quality Standards, Fixed Costs, and Competition. *The RAND Journal of Economics* 22 (1991), 490-504.

Ryan, D. Two Views on Security Software Liability: Let the Legal System Decide. *IEEE Security & Privacy* 1:1 (2003), 70-72.

Schneier, B. Information Security: How Liable Should Vendors Be? *Computer World*, October 28, 2004.

Schneier, B. Do Federal Security Regulations Help? *Information Security*, November 2006.

Spence, M. Monopoly, Quality, and Regulation. *The Bell Journal of Economics* 6:2 (1975), 417-429.

Siponen, M. Five Dimensions of Information Security Awareness, *ACM SIG CAS Computers and Society* 31:2 (2001), 24 - 29

Spence, M. Consumer Misperceptions, Product Failure and Producer Liability. *The Review of Economic Studies* 44:3 (1977), 561-572.

Sundararajan, A. Nonlinear Pricing of Information Goods. *Management Science* 50:12 (2004), 1660 – 1673.

Telang, R., Rajan, U. and Mukhopadhyay, T. The Market Structure for Internet Search Engines. *Journal of Management Information Systems* 21:2 (2004), 137 – 160.

Varian, H. R. Managing Online Security Risks. *New York Times*, June 1, 2000.

Varian, H.R. Economics of Information Technology. *Mattioli Lecture*, Bocconi University, Milan, Italy, November 2001.

Yurcik, W. and Doss, D. Cyberinsurance: A Market Solution to the Internet Security Market Failure. *Workshop on Economics and Information Security*, University of California, Berkeley, May 16-17, 2002.

3 The Effect of a Liability Mechanism on Software Security: The Monopoly Case

3.1 Introduction

Software has become an indispensable part of our daily lives. Not only large-scale operating systems in businesses but also home electric appliances such as television sets and telephones are powered by thousands of program instructions. As our society depends more on software, its malfunction becomes more disastrous. For example, the task force investigating the cause of the electricity blackout on August 14, 2003, which crippled much of the Northeast USA and parts of Canada concluded that a software failure at Akron, Ohio-based FirstEnergy Corporation might have contributed significantly to the power outage (Verton 2003). The explosion of software usage has increased the vulnerability of computer systems, highlighting security concerns.

According to Microsoft Progress Report on Security, malicious software code has been around for decades, but only in the last few years, the Internet, high-speed connections and millions of new computing devices have converged to create a truly global computing network in which a virus or worm can circle the world in a matter of minutes. Consequently, external attacks such as viruses and worms that exploit the vulnerabilities of the operating systems get more serious. Ernst & Young's Global Information Security Survey 2004 found that 77% of the respondents rated major viruses and worms to have the highest threat intensity (Ernst & Young 2004).

The abundance of flawed software has been identified as the main cause of the poor security of computer networks (Yurcik and Doss 2002). In September 2003, Steven Adler, senior security strategist for Microsoft Corporation apologized for the damage and losses caused by the

onslaught of computer viruses that have attacked his company's software. According to Fisk (2002), the software industry is at a sub-optimal, but self-supporting equilibrium that does not support the efforts required for developing highly secure software. He also argues that the software vendors do not have enough incentive to apply the available techniques to prevent software vulnerability. Until recently, low security awareness of customers has been blamed for discouraging software vendors to invest on security improvement. Since customers are not capable of predicting security loss and thus, do not appreciate the value of secure software, the vendor perceives low demand for security. As a result, developing highly secure software is a significant risk to the vendors given the small market and the high development cost.

However, low security awareness does not seem to fully explain poor software quality. Although individual consumers may be unaware of security issues, it is not the case for the firms. Especially since 9/11, security has become a major issue for business, and managers have become much more aware of security concerns. In spite of increasing security awareness, the quality of software is still in question. Security researchers argue that unless software vendors are held liable for their products, they are not willing to produce secure software (Armour and Humphrey 1993, Harmon 2003, Sager and Green 2002, Schneier 2004, Varian 2000). Since the risk is entirely on the customer side, unlike other industries, such as an auto industry where the manufacturers are accountable for any defect of their products, the software vendors may not have enough incentive to develop secure software especially if the vendor dominates the entire market.

As an incentive mechanism for software security quality improvement, software liability has been intensely discussed among computer scientists, jurists, and policy makers for years. In spite of the long years of debate, the researchers have not come to a conclusion about whether

imposing liability on the software vendor leads to highly secure software. In a recent article, two experts in the field take opposite viewpoints regarding software liability (Heckman 2003, Ryan 2003). Ryan (2003) argues that it is not practical for consumers to create their own security software, and thus it is reasonable to assume that manufacturers of such software should ensure the reliability of their products. On the other hand, Heckman (2003) argues that software does not have the characteristics of other products such as automobiles that would support legal liability, and thus, liability is not the appropriate tool for reducing the number and severity of software security holes.

Despite long years of argument among security experts, the effectiveness of software liability is a question that still remains unanswered. There are prerequisites for the software liability mechanism. For example, we should be able to quantify security measures such as loss associated with a certain attack. Quantifying loss itself is not an easy task. But the problem is that the effectiveness of software liability is questionable even assuming full feasibility of software liability. The questions that we aim to answer in this paper are whether software liability is beneficial, and if so, under what conditions. Answers to our research questions may give policy makers an idea about which factors decide the viability of the software liability, which has been considered to be a potentially effective mechanism for better security.

Although the software vendors have not faced liability for security failures yet, some companies are demanding liability clauses in contracts with vendors, holding them responsible for any security breach connected to their software (Fisher 2002). Consequently, software vendors face potential threat of liability. In October 2003, Microsoft was sued in California, based on the claim that its market-dominant software is vulnerable to viruses and worms capable of triggering massive, cascading failures in global networks. The lawsuit comes in the wake of

two major worms that have exploited vulnerabilities in Microsoft software: Slammer and Blaster. Due to potential liability as well as increasing customer security awareness, some leading software vendors started making efforts to develop secure software. In the now famous memo in January 2002, Bill Gates stated as follows:

"In the past, we've made our software and services more compelling for users by adding new features and functionality, and by making our platform richly extensible. We've done a terrific job at that, but all those great features won't matter unless customers trust our software. So now, when we face a choice between adding features and resolving security issues, we need to choose security. Our products should emphasize security right out of the box, and we must constantly refine and improve that security as threats evolve."

During February and March 2002, Microsoft stopped developing Windows features and focused on improving the security quality of Windows. For a monopolistic vendor pursuing profit such as Microsoft, this shows how serious the security problem is. This memo clearly shows that the vendor faces a new era when the development of secure software is a significant issue for business.

In this paper, we investigate whether imposing liability motivates the monopolist to improve the security quality of its software, and determine the conditions under which the liability mechanism is effective in terms of security improvement. Then we examine how the liability mechanism affects the monopolist's market coverage. Both security quality and market coverage are important to evaluate the effectiveness of the liability mechanism since policy makers need to make sure that the monopolist has incentive to serve enough customers with providing reasonably secure software with the liability mechanism. There are two issues surrounding the liability mechanism: effectiveness and feasibility. The focus of this paper is the former, i.e.,

whether liability leads to higher security quality or higher social surplus. Implementing the liability mechanism is certainly not an easy problem. For example, quantifying loss is difficult. But only when the mechanism is proven to be effective, it is worth discussing the implementation issue. In spite of a long debate by computer scientists, jurists, and policy makers, there has not been an agreement on the effectiveness of the liability mechanism. From an economic perspective, we analyze the software market to evaluate the effectiveness of software liability. We model the case where the vendor focuses on security development, and identify the factors that affect the monopolist's decision on security quality and market coverage.

We first consider the case where both the vendor and customers are fully aware of security issues. In other words, both parties have rational expectation of the loss due to security breaches of the software. Unlike direct contractual risk-sharing that may impose partial liability on the vendor, we examine the legal and uniform liability that makes the vendor fully responsible for the entire loss. We determine the conditions that affect the monopolist's market coverage. We find that high marginal willingness to pay for the software leads to full market coverage when the customers are responsible for the entire risk. Once liability is imposed on the monopolist, full market coverage obtains for any level of marginal willingness to pay depending on conditions, that is, the monopolist covering the partial market without liability may have an incentive to fully increase its market coverage when it becomes liable. The monopolist offers even higher security quality in the full market with liability than in the partial market without liability. This result implies that under certain conditions, avoiding liability is not beneficial to the the software vendor in the sense that the liability mechanism may give the monopolist an opportunity to have more customers.

Overall, we find that liability does not always lead to high security quality, nor does it always lead to high social surplus. We identify the factors that affect the monopolist's decision on security quality level. We find that no matter which party is liable for the loss in the fully covered market, security quality increases with the proportion of loss to cost and decreases with the baseline utility from the software. We also find that high marginal willingness to pay leads to high security quality in the current market without liability. This finding implies that educating the users to appreciate security may be an effective way to motivate the vendor to improve security quality. We examine the welfare implications of the liability mechanism. Our results show that security quality is underprovided in the market where customers are responsible for the entire risk, while socially optimal level of security quality is offered once liability is imposed on the monopolist. We also find that the social planner offers higher security quality in the market without liability than with liability. This result implies that there is a tradeoff between security quality and risk-sharing, even for the social planner. We investigate whether imposing liability on the vendor leads to better security or higher consumer surplus in the fully covered market. Interestingly, our results indicate that imposing liability may discourage the monopolist from improving security contrary to security practitioners' expectation, whereas consumer surplus is higher with the liability mechanism. When the marginal willingness to pay is relatively low, the liability mechanism brings higher social surplus. Finally, we consider the case where customers have poor security awareness, and thus, underestimate the risk. We find that the liability mechanism is effective in improving security quality as well as consumer surplus when customers are not capable of properly predicting the loss unlike the vendor. This paper contributes to the literature in that it not only gives a clear picture of liability in the software

market from an economic perspective but also provides implications to managers and policy makers.

The rest of the article is organized as follows. Section 3.2 describes the general model. In order to get additional insights on the impact of the liability mechanism, we assume some reasonable functional forms for our model in Section 3.3 and Section 3.4. In Section 3.3, we examine the software market without liability, and examine the conditions under which market coverage is endogenously determined. We then obtain the optimal levels of security quality of the monopolist's software product in the partially covered market and the fully covered market. In Section 3.4, we analyze the market with liability. We examine the policy implications of the liability mechanism in Section 3.5. We analyze the impact of software liability on the levels of security quality, consumer surplus and social welfare in the fully covered market. We then investigate whether considering poor security awareness of customers changes the results. In Section 3.6, we discuss robustness of our results when functionality is endogenized along with security quality. Lastly, we offer our concluding remarks in Section 3.7.

3.2 General Model

We analyze a software market dominated by a monopolistic software vendor, and investigate whether imposing liability gives the vendor an incentive to improve security quality of the software. We set up a model using a theoretical framework built on the models of vertical quality differentiation (Mussa and Rosen 1978, Ronnen 1991, Spence 1975). There are two types of players in the market: a monopolistic software vendor and customers. The monopoly case is relevant in the software industry. Consider the dominating position of Microsoft in the PC operating systems market, for example. Customers in our model are firms that are likely to have

higher incentive to invest on security than do individuals whose security awareness level is low in reality. We first consider the case where both the vendor and customers are aware of security issues. In other words, they have rational expectations of the loss due to security breaches of the software product. Let q be security quality scaled between 0 and 1. Security quality measures how vulnerable the software is at the product launch. Bug-free software can be considered to be of perfect security quality (q = 1). $L(q)$ represents the loss function, which is convex: $L'(q) < 0$ and $L''(q) > 0$. As security quality increases, the loss decreases at a diminishing rate. The loss function is set up in a way that increasing security quality reduces the expected loss from cyberattack in the life-span of the product, and that the loss is entirely preventable if the installed software exhibits perfect security quality ($L(1) = 0$). This is consistent with security experts' view such as Fisk (2002).

Arora et al. (2006) argue that the best strategy for software vendors is to introduce the product as early as possible and then patch it later. Consequently, security quality of the software at the product launch is lower than expected. Security experts argue that the software vendors should improve the initial security quality of their products since the defects of poorly written software are exploited by the malicious hackers to attack the computer systems (Harmon 2003). Until recently, software vendors have been able to avoid liability for any defect of their products and thus, they have little incentive for security quality improvement. However, software vendors are now facing potential threat of liability as demonstrated in the Microsoft lawsuit case in October 2003. As a result, software vendors are more concerned about the security quality of their products. As noted before, in early 2002, Microsoft stopped developing Windows feature and focused on improving security quality of Windows. Our model captures the current phenomenon by analyzing the market where the vendor emphasizes security quality. V is the

baseline utility that a customer gains from the software product if it is perfectly secure. Let $C(q,V)$ be the cost function and assume that $C(q,V)$ is a convex function of q. Denote $C'(q,V)$ be $\frac{\partial C(q,V)}{\partial q}$ and $C''(q,V)$ be $\frac{\partial^2 C(q,V)}{\partial q^2}$. The convexity of the cost function means $C'(q,V) > 0$ and $C''(q,V) > 0$. This implies that the cost increases as the security quality increases at a growing rate. We also assume that $\frac{\partial C(q,V)}{\partial V} > 0$, implying that offering higher utility to customers costs more. This is realistic in the sense that it is more costly to allow the same level of security to more complicated software with more features.

3.2.1 Baseline

We examine the market where the entire risk is on the customer side, which models the current state of the software market. Our model assumes that customers have rational expectations of the loss due to the vulnerability of the software. Customers value security quality in that q reduces expected loss. We model the case where the vendor focuses on security quality. In the market where the customers are responsible for any possible loss due to security breaches of the software, the expected utility of a customer is defined as follows:

$$EU = \theta(V - L(q)) - p.$$

Recall V denotes the baseline utility that a customer gains from the software if it is perfectly secure. $L(q)$ represents the expected loss when q is the level of security quality scaled between 0 and 1. p is the price and θ captures customer heterogeneity indicating how much utility a customer derives from the software. Losses arise out of disruption to business activities. Thus, the same attack may lead to more damage to some firms than to others. If θ is high, the firm cares more about security quality of the software, in that it enjoys more utility from the product,

but also suffers more disutility from a successful attack. It reflects that some firms are more concerned about security issues than others. Firms in the financial industry or in health care industry are examples of high-θ firms. We assume that θ is distributed on the interval, $[\underline{\theta},\infty)$. Let $f(\theta)$ be the probability density function of θ and $F(\theta)$ be the corresponding cumulative distribution function of θ. Then

$$\int_a^b f(\theta)d\theta = \Pr[a \le \Theta \le b] \quad \text{and} \quad F(\theta) = \int_{\underline{\theta}}^{\theta} f(y)dy.$$

Since the entire risk is on the customers, the vendor is not responsible for any possible loss that may occur due to software vulnerabilities. The production of information good such as software involves high fixed costs but low variable costs. In other words, the cost of producing the original copy is substantial whereas the cost of making additional copies is negligible. As a result, given the context of software product, the cost does not depend on quantity. Thus, in our model, it is reasonable to assume zero variable cost. Then, the software vendor's expected profit is

$$E\pi = px - C(q,V)$$

where x is demand for the product and $C(q,V)$ represents the production cost of software with security quality level q. This convex cost function implies that the cost increases as security quality level rises at a growing rate and that the cost to offer a certain level of security quality becomes higher when the software provides higher utility ($V \uparrow$).

3.2.2 Software Liability

In this section, we model the market where liability is imposed on the vendor side. This is common in some industries. For example, car manufacturers are legally responsible not only for

the product defects but also for any injury or loss that is caused by the defects. However, unlike other industry, software vendors have been able to avoid liability for any vulnerability of the product. Accordingly, security experts and jurists argue that imposing liability on the vendor may work well as an incentive mechanism for security quality improvement of the software products (Ryan 2003, Schneier 2004). We model the market where the vendor is accountable for any loss due to vulnerability of its software product. In this market, the customers do not have to take the expected loss into account when they decide whether or not to buy the software. They only care about the baseline utility and price since the expected loss does not affect total customer utility:

$$EU = \theta V - p.$$

In this market, where the risk is on the software vendor, the cost function for the vendor has two factors: the fixed cost for initial security quality development and the expected loss which plays a role of a variable cost. The expected profit of the vendor is then:

$$E\pi = (p - L(q))x - C(q, V).$$

3.2.3 Equilibrium Analysis

In this section, we analyze the monopolistic software vendor's optimal choices on price and security quality level and examine the effect of a liability mechanism on the security quality. Grounded on the models of vertical quality differentiation (Mussa and Rosen 1978, Ronnen 1991, Spence 1975), we set up a game, which consists of three stages. At the first stage, the monopolistic vendor decides the level of security quality q. At the second, the vendor sets up the price p. Then the customers decide whether or not to buy the software product. We focus on the comparison of optimal qualities in the fully covered market in order to reflect what happens in

some real-world software markets such as operating systems software market. We also argue the issues of partially covered market to a certain extent.

Full Market Case

Consider the no liability case. Let $\hat{\theta}$ define the marginal willingness to pay of the customer who is indifferent between buying and not buying. Then

$$EU = \hat{\theta}(V - L(q)) - p = 0 \Leftrightarrow \hat{\theta} = \frac{p}{V - L(q)}.$$

Note that x=1 for all p, V, q such that $\hat{\theta} \le \underline{\theta}$. Then the monopolist charges the highest possible price, $\overline{p}$:

$$\hat{\theta} \le \underline{\theta} \Leftrightarrow \frac{p}{V - L(q)} \le \underline{\theta} \Leftrightarrow \overline{p} = \underline{\theta}(V - L(q)).$$

Thus, under full market coverage, the expected profit is

$$E\pi = \overline{p} - C(q,V) = \underline{\theta}(V - L(q)) - C(q,V).$$

Let 1 in the subscript denote the case without liability and F represent full market. The first order condition for q is

$$\frac{\partial E\pi}{\partial q} = -\underline{\theta}L'(q) - C'(q,V) = 0 \Leftrightarrow \frac{L'(q_{1F}^*)}{C'(q_{1F}^*,V)} = -\frac{1}{\underline{\theta}}.$$

Now, consider the case with liability. Then the willingness to pay of the marginal customer is $\hat{\theta} = \frac{p}{V}$. Note that x=1 for all p, V, q such that $\hat{\theta} \le \underline{\theta}$. Thus, the software vendor maximizes profit at the optimal price, $\overline{p}$:

$$\hat{\theta} \le \underline{\theta} \Leftrightarrow \frac{p}{V} \le \underline{\theta} \Leftrightarrow \overline{p} = \underline{\theta}V.$$

Then the expected profit is

$$E\pi = \bar{p} - L(q) - C(q,V) = \underline{\theta}V - L(q) - C(q,V).$$

Let 2 in the subscript denote the case with liability. The first order condition for q is

$$\frac{\partial E\pi}{\partial q} = -L'(q) - C'(q,V) = 0$$

which leads to the following condition:

$$\frac{L'(q_{2F}^*)}{C'(q_{2F}^*,V)} = -1.$$

Proposition 1: *In a fully covered market by a monopolistic software vendor, imposing liability on the vendor for any loss due to security breaches leads to higher security quality when customers have high marginal willingness to pay* $(\underline{\theta} > 1)$. *Otherwise* $(\underline{\theta} < 1)$, *security quality may decrease with the liability mechanism.*

Proof. See the Appendix.

The result holds when the monopolistic software vendor covers the full market in both cases: with and without liability. With these generalized functional forms, obtaining clear-cut full market conditions is not tractable as will be seen in the next section. Intuitively, it is reasonable that a vendor has stronger incentive to serve more customers when customers value its software more. In other words, higher θ of the customers is more likely to lead to full market than lower θ. Our result shows that $q_{1F}^* > q_{2F}^*$ when the lower bound of θ is greater than 1, which is more likely to be the full market condition for both with and without liability cases than $\underline{\theta} < 1$..

Partial Market Case

We examine the partial market case and argue that comparison of security quality with and without liability is not decisive, that is, security quality can either increase or decrease when liability is imposed on the monopolistic software vendor. In this section, we show that there exist conditions under which imposing liability on the monopolistic software vendor leads to higher

security quality in the partially covered market. Consider the case without liability first. The willingness to pay of the marginal customer is $\hat{\theta} = \frac{p}{V - L(q)}$. Thus, the demand for the software is

$$x = \int_{\hat{\theta}}^{\infty} f(t)dt = F(\infty) - F(\hat{\theta}) = 1 - F(\frac{p}{V - L(q)}).$$

The expected profit for the vendor is then

$$E\pi = px - C(q,V) = p\left(1 - F(\frac{p}{V - L(q)})\right) - C(q,V).$$

The first order condition for p is

$$\frac{\partial E\pi}{\partial p} = 1 - F(\frac{p}{V - L(q)}) - \frac{p}{V - L(q)} f(\frac{p}{V - L(q)}) = 0.$$

Rearranging the above condition leads to

$$f(\frac{p}{V - L(q)}) = \frac{V - L(q)}{p}\left(1 - F(\frac{p}{V - L(q)})\right).$$

The first order condition for q is

$$\frac{\partial E\pi}{\partial q} = -\left(\frac{p}{V - L(q)}\right)^2 f(\frac{p}{V - L(q)})L'(q) - C'(q,V) = 0.$$

Substituting $f(\frac{p}{V - L(q)})$ in the above equation leads to

$$-\left(\frac{p}{V - L(q)}\right)^2 \frac{V - L(q)}{p}\left(1 - F(\frac{p}{V - L(q)})\right)L'(q) - C'(q,V) = 0$$

$$\Leftrightarrow -\frac{p}{V - L(q)}\left(1 - F(\frac{p}{V - L(q)})\right)L'(q) - C'(q,V) = 0.$$

Let P in the subscript denote partial market. Then,

$$\frac{L'(q_{1P}^*)}{C'(q_{1P}^*,V)} = -\frac{V - L(q_{1P}^*)}{p_{1P}^*} \frac{1}{1 - F(\frac{p_{1P}^*}{V - L(q_{1P}^*)})}.$$

Now, consider the case with liability. The willingness to pay of the marginal customer is $\hat{\theta} = \frac{p}{V}$ and the corresponding demand for the software is $x = \int_{\hat{\theta}}^{\infty} f(t)dt = 1 - F(\frac{p}{V})$. The expected profit for the vendor is given by

$$E\pi = (p - L(q))x - C(q,V) = (p - L(q))\left(1 - F(\frac{p}{V})\right) - C(q,V).$$

The first order condition for *p* is

$$\frac{\partial E\pi}{\partial p} = 1 + L(q)f(\frac{p}{V})\frac{1}{V} - F(\frac{p}{V}) - pf(\frac{p}{V})\frac{1}{V} = 0.$$

However, we do not need the above condition for our purpose because $f(\frac{p}{V})$ does not appear in the first order condition for *q* as shown below:

$$\frac{\partial E\pi}{\partial q} = -L'(q)\left(1 - F(\frac{p}{V})\right) - C'(q,V) = 0.$$

From the above condition,

$$\frac{L'(q_{2P}^*)}{C'(q_{2P}^*,V)} = -\frac{1}{1 - F(\frac{p_{2P}^*}{V})}.$$

With general functional forms, it is not feasible to obtain the optimal quality in a closed form. Also, it is difficult to compare security quality with or without liability in the partially covered market. We show that there exists a case where imposing liability gives incentive to provide higher security quality. Suppose that the following condition holds:

$$\frac{p_{1P}^*}{V - L(q_{1P}^*)} = \frac{p_{2P}^*}{V} = \hat{\theta}.$$

Then the optimal qualities meet the following conditions:

$$\frac{L'(q_{1P}^*)}{C'(q_{1P}^*,V)} = -\frac{V}{p_{2P}^*}\frac{1}{1-F(\frac{p_{2P}^*}{V})}$$

$$\frac{L'(q_{2P}^*)}{C'(q_{2P}^*,V)} = -\frac{1}{1-F(\frac{p_{2P}^*}{V})}.$$

Note that θ is distributed on $[\underline{\theta},\infty)$. Thus, any case is possible. Suppose that $\underline{\theta} \le \hat{\theta} < 1$. Then $\frac{L'(q_{1P}^*)}{C'(q_{1P}^*,V)} < \frac{L'(q_{2P}^*)}{C'(q_{2P}^*,V)}$, which implies, $q_{1P}^* < q_{2P}^*$. When $\hat{\theta} > 1$, $q_{1P}^* > q_{2P}^*$. In the fully covered market where customers are likely to have higher marginal willingness to pay, our result shows that security quality decreases when liability is imposed on the vendor. Our partial market analysis shows that there exists a case where security quality with liability is strictly greater than security quality without liability. Unlike fully covered market, the liability mechanism may bring higher security quality in the partially covered market depending on certain conditions. From this finding, we can infer security quality from the perspective of a social planner. Social planner is like a monopolist with full liability, except that the social planner will have a smaller willingness to pay from a marginal customer. Thus, the social planner provides lower security quality than the fully liable monopolist. Comparing to a monopolist without liability, the social planner offers higher security quality if and only if the average customer served by the social planner has a higher willingness to pay than the marginal buyer served by the monopolist. With these general functional forms, obtaining full market conditions in a closed form is not feasible. In order to get additional insights, we assume linear loss and quadratic cost functions for further analysis and examine how our result binds with market coverage conditions in the next section.

3.3 Software Market Without Liability

With the generalized functional forms in the original setting, it is not feasible to obtain a full market condition in a closed form. For further analysis, we make assumptions on the functional forms. Let $a(1-q)$ be the expected loss when a is a parameter capturing the degree of loss due to security breaches ($a>0$) and q is the level of security quality scaled between 0 and 1. This implies that loss is entirely preventable if the installed software exhibits perfect security quality. Recall that V is the baseline utility that a customer gains from the software if it is perfectly secure. Denote tVq^2 be the cost function for security quality development where t is a parameter modeling the degree of the development cost for the software with security baseline utility V and security quality q ($t>0$). We also assume θ to follow a uniform distribution on $(\overline{\theta}-1,\overline{\theta})$. These assumptions are often made in the domain of applied economics. We first analyze the market where the entire risk is on the customers.

3.3.1 Partial Market Coverage

Recall that the game consists of three stages. At the first stage, the monopolistic vendor decides the level of security quality q. At the second, the vendor sets up the price p. Then the customers decide whether or not to buy the software product. Let $\hat{\theta}$ define the marginal willingness to pay of the customer who is indifferent between buying and not buying. Then the expected utility of the marginal customer is

$$EU=\hat{\theta}(V-a(1-q))-p=0\Leftrightarrow\hat{\theta}=\frac{p}{V-a(1-q)}.$$

The customers who have higher marginal willingness to pay than $\hat{\theta}$, will buy the software product. Then the demand for the software is $x = \bar{\theta} - \frac{p}{V - a(1-q)}$. The expected profit for the monopolist becomes $E\pi = p(\bar{\theta} - \frac{p}{V - a(1-q)}) - tVq^2$. The first order condition for p is $\frac{\partial E\pi}{\partial p} = \bar{\theta} - \frac{2p}{V - a(1-q) = 0}$. Thus, the optimal price is $p^* = \frac{\bar{\theta}}{2}(V - a(1-q))$. Substituting p^* in the profit function leads to $E\pi = \frac{\bar{\theta}^2}{4}(V - a(1-q)) - tVq^2$. The first order condition for q is $\frac{\partial E\pi}{\partial q} = \frac{a\bar{\theta}^2}{4} - 2tVq = 0$. Recall that 1 denotes the case without liability and P in the subscript represents partial market coverage. Solving the FOC yields the optimal security quality as follows:

$$q_{1P}^* = \frac{a\bar{\theta}^2}{8tV}.$$

The optimal level of price is then

$$p_{1P}^* = \frac{\bar{\theta}}{2}(V - a(1 - \frac{a\bar{\theta}^2}{8tV})).$$

3.3.2 Full Market Coverage

In this section, we examine the conditions under which the market is fully covered by the monopolist and then analyze the monopolist's decision on security quality in the fully covered market. In the context of software, this is a relevant case since full market coverage is quite often achieved in reality. When $x \geq 1$ in the partial market, full market coverage obtains from the

outset. We first analyze the market without liability. In Section 3.3.1, we obtained the demand in the partially covered market as follows:

$$x_{1P} = \bar{\theta} - \frac{p}{V - a(1-q)}.$$

The optimal level of price is

$$p_{1P}^{*} = \frac{\bar{\theta}}{2}(V - a(1-q)).$$

Substituting p in the demand with the equilibrium price yields $x_{1P}^{*} = \frac{\bar{\theta}}{2}$. Note that the taste parameter of the marginal customer who is indifferent between buying or not is contingent on the vendor's quality and price decisions. Since, the demand becomes a function of the upper bound of the taste parameter at the optimal price level, the demarcation of market coverage is endogenous. Recall that $q_{1P}^{*} = \frac{a\bar{\theta}^2}{8tV}$. Thus,

$$\hat{\theta} = \frac{p_{1P}^{*}}{V - a(1-q_{1P}^{*})} = \frac{\frac{\bar{\theta}}{2}(V - a(1-\frac{a\bar{\theta}^2}{8tV}))}{V - a(1-\frac{a\bar{\theta}^2}{8tV})} = \frac{\bar{\theta}}{2}.$$

Since $\hat{\theta} = \frac{\bar{\theta}}{2}$, the partial market coverage occurs when $\hat{\theta} = \frac{\bar{\theta}}{2} > \bar{\theta} - 1 \Leftrightarrow \bar{\theta} < 2$, and the full market coverage occurs when $\bar{\theta} > 2$. This implies that the monopolist software vendor chooses to cover the full market when customers have high marginal willingness to pay for the software. Assume that customers have high enough marginal willingness to pay ($\bar{\theta} > 2$). Then the monopolist decides to cover the entire market and charges the highest possible price, $\bar{p}$:

$$\hat{\theta} \le \bar{\theta} - 1$$
$$\Leftrightarrow \frac{p}{V - a(1-q)} \le \bar{\theta} - 1$$
$$\Leftrightarrow p \le (\bar{\theta} - 1)(V - a(1-q))$$
$$\Leftrightarrow \bar{p} = (\bar{\theta} - 1)(V - a(1-q)).$$

Under full market coverage, the monopolist's expected profit at the optimal price is $E\pi = \bar{p} - tVq^2 = (\bar{\theta} - 1)(V - a(1-q)) - tVq^2$. The first order condition for q is $\frac{\partial E\pi}{\partial q} = a(\bar{\theta} - 1) - 2tVq = 0.$ Recall that 1 denotes the case without liability. Let F in the subscript stand for full market coverage. Solving the first order condition yields the optimal security quality:

$$q_{1F}^{*} = \frac{a(\bar{\theta} - 1)}{2tV}.$$

The optimal level of price is then

$$p_{1F}^{*} = (\bar{\theta} - 1)(V - a(1 - \frac{a(\bar{\theta} - 1)}{2tV})).$$

The monopolist's expected profit in the fully covered market without liability is

$$E\pi_{1F}^{*} = (\bar{\theta} - 1)(V - a(1 - \frac{a(\bar{\theta} - 1)}{2tV})) - \frac{a^2(\bar{\theta} - 1)^2}{4tV}$$
$$= (\bar{\theta} - 1)(V - a) + \frac{a^2(\bar{\theta} - 1)^2}{4tV}.$$

3.4 Software Market With Liability

3.4.1 Partial Market Coverage

We analyze the market where the vendor is liable for the security breaches of its software product, assuming all losses are verifiable. Under the liability mechanism, the vendor is supposed to cover the entire loss incurred due to the attacker's exploitation of the software vulnerability. Since the risk is not on the customers but on the vendor, the utility function of customers is not affected by the expected loss. The customer's utility function is $EU = \theta V - p.$ Although the variable cost of production is assumed to be zero, the liability mechanism makes the expected loss be the variable cost. Thus, the vendor's expected profit function is $E\pi = (p - a(1-q))x - tVq^2.$ Then the willingness to pay of the marginal customer is

$$EU = \hat{\theta}V - p = 0 \Leftrightarrow \hat{\theta} = \frac{p}{V}.$$

The customers who have higher marginal willingness to pay than $\hat{\theta}$ will buy the software product whereas others will not. Thus, the demand is $x = \bar{\theta} - \frac{p}{V}.$ The expected profit for the monopolist who is liable for the loss is $E\pi = (p - a(1-q))(\bar{\theta} - \frac{p}{V}) - tVq^2.$ The first order condition for the price is $\frac{\partial E\pi}{\partial p} = \bar{\theta} - \frac{2p}{V} + \frac{a(1-q)}{V} = 0.$ The optimal price level is $p^* = \frac{V\bar{\theta} + a(1-q)}{2}.$ Replacing price in the expected profit with p^* yields

$$E\pi = \frac{1}{4V}(V\bar{\theta} - a(1-q))^2 - tVq^2.$$

Then the first order condition for q is

$$\frac{\partial E\pi}{\partial q} = \frac{a}{2V}(V\bar{\theta} - a(1-q)) - 2tVq = 0$$
$$\Leftrightarrow q(a^2 - 4tV^2) = a^2 - aV\bar{\theta}.$$

Assume that $a < 2\sqrt{tV}$. Then the second order condition for q is

$$\frac{\partial E\pi^2}{\partial^2 q} = \frac{a^2}{2V} - 2tV < 0.$$

Recall that 2 represents the case with liability. The optimal security quality is

$$q_{2P}^* = \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2}.$$

Suppose that $a > 2\sqrt{tV}$. Then the expected profit is maximized at either $q = 0$ or $q = 1$. The conditions for each boundary solution are as follows:

$$\frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} > \frac{1}{2} \Rightarrow q_{2P}^* = 0$$
$$\frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} < \frac{1}{2} \Rightarrow q_{2P}^* = 1.$$

Proposition 2: *When the entire risk is on the customer side in the partially covered market, security quality* (q_{1P}^*) *increases with the proportion of loss to cost* (a/t) *and marginal willingness to pay* $(\bar{\theta})$ *and decreases with utility* (V). *When liability is imposed on the vendor and the expected loss is small* $(0 < a < 2\sqrt{tV})$, *he optimal quality is an interior solution, which increases with marginal willingness to pay* $(\bar{\theta})$. *When the expected loss is large* $(a > 2\sqrt{tV})$, *high* $\bar{\theta}$ *leads to* $q_{2P}^* = 1$ *and low* $\bar{\theta}$ *yields* $q_{2P}^* = 0$. *With medium* $\bar{\theta}$, $q_{2P}^* = 1$ *if the expected loss is bounded and* $q_{2P}^* = 0$ *otherwise.*

Proof. See the Appendix.

Without a liability mechanism, the monopolist improves the security quality with the proportion of loss to cost $(\frac{a}{t})$, which can be interpreted as the efficiency of the security development. We find that higher marginal willingness to pay leads to higher security quality with or without liability, implying that educating customers to appreciate security may be an effective way to motivate the software vendor to improve the security quality of its software. When the entire risk is on the customer side, the software vendor always offers positive security quality. When liability is imposed on the software vendor and the expected loss is small, the monopolistic software vendor still has incentive to offer positive security quality. However, when the expected loss is large and the marginal willingness to pay of the customers is small, the vendor may offer zero security quality so that it exits the market and makes zero profit.

3.4.2 Full Market Coverage

In this section, we analyze the market with liability and examine the conditions under which the market is fully covered by the monopolist. We model the market where the vendor is responsible for the entire loss due to security breaches which are exploited by malicious agents. The full market coverage condition is $x \geq 1$. The demand for the software in the partially covered market is

$$x_{2P} = \bar{\theta} - \frac{p}{V}.$$

In Section 3.4.1, we derived the equilibrium price as follows:

$$p_{2P}^{*} = \frac{V\bar{\theta} + a(1-q)}{2}$$

Suppose that $a < 2\sqrt{tV}$. Then the optimal quality is

$$q_{2P}^{*} = \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2}.$$

We can compute the demand by substituting price and quality with optimal levels.

$$x_{2P}^{*} = \bar{\theta} - \frac{p_{2P}^{*}}{V} = \frac{\bar{\theta}}{2} - \frac{a(1-q_{2P}^{*})}{2V} = \frac{1}{2V}\left(\bar{\theta}V - \frac{a^2V\bar{\theta} - 4atV^2}{a^2 - 4tV^2}\right).$$

The full market condition for the market with liability is then

$$x_{2P}^{*} = \frac{1}{2V}\left(\bar{\theta}V - \frac{a^2V\bar{\theta} - 4atV^2}{a^2 - 4tV^2}\right) \geq 1$$
$$\Leftrightarrow \frac{(\bar{\theta}V - 2V)(a^2 - 4tV^2) - a^2V\bar{\theta} + 4atV^2}{a^2 - 4tV^2} \geq 0.$$

Proposition 3: *In the market without liability, the monopolistic software vendor covers the full market when the customers have high marginal willingness to pay* ($\bar{\theta} \geq 2$). *When the customers have low marginal willingness to pay* ($1 \leq \bar{\theta} < 2$), *some customers are left out. Assume that the possible expected loss is bounded* ($a \geq 2\sqrt{tV}$). *In the market with liability, the monopolistic software vendor has an incentive to cover the full market for any level of marginal willingness to pay of the customers. Full market coverage obtains when the following conditions hold:*

$$\bar{\theta} > 2 + \frac{t}{2}, \qquad a \in (0, 2\sqrt{tV})$$
$$2 < \bar{\theta} < 2 + \frac{t}{2}, \quad a_{+} > 2\sqrt{tV}, \qquad a \in (0, \mathrm{Min}\{2\sqrt{tV}, a_{-}\})$$
$$2 < \bar{\theta} < 2 + \frac{t}{2}, \quad a_{+} < 2\sqrt{tV}, \qquad a \in (0, a_{-}) \text{ or } a \in (a_{+}, 2\sqrt{tV})$$
$$1 < \bar{\theta} < 2, \qquad a_{+} < 2\sqrt{tV}, \qquad a \in (a_{+}, 2\sqrt{tV})$$

where $a_{-} = tV\left(1 - \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right)$ and $a_{+} = tV\left(1 + \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right)$.

Proof. See the Appendix.

When the customers are responsible for the entire risk and the marginal willingness to pay is high, the monopolist has an incentive to cover the full market. Otherwise, the monopolist covers only the partial market. This is quite intuitive in the sense that full market coverage obtains when the customers value the product more. Proposition 3 shows that the marginal willingness to pay for the software is the only factor that decides the market coverage when the software vendors can avoid liability. However, it does not hold once liability is imposed on the vendor. The results show that the monopolist is willing to cover the full market given any level of the marginal willingness to pay in the market with liability. We examine the case where the optimal quality is an interior solution in the full market with liability. Depending on the conditions, full market coverage obtains even for the customers with low marginal willingness to pay. This is interesting in the sense that the monopolist who avoids liability and serves partial market may want to increase the market coverage even when liable. Thus, the liability mechanism may give an opportunity for the monopolist to serve more customers who may be better off since they do not have to worry about the possible loss any more. Interestingly, there exist conditions under which the monopolist covers the full market no matter what level of the baseline utility is given. Since the vendor is responsible for the entire loss, it has an incentive to cover the full market only when the maximum possible loss is bounded. In the market without liability, the monopolist covers the full market when customer valuation is high no matter how high the maximum possible loss is. However, once liability is imposed on the vendor, the vendor never serves the full market when it faces catastrophic loss.

Suppose that the full market condition holds. Let $\hat{\theta}$ be the marginal willingness to pay of the customer who is indifferent between buying and not buying: $\hat{\theta} = \frac{p}{V}$. Consider the case where $\bar{\theta}$ and a meet the full market condition. Then $x = 1$ for all p, V, q and a such that $\hat{\theta} \leq \bar{\theta} - 1$. Then

the monopolist charges the highest possible price, $\bar{p} = (\bar{\theta} - 1)V$. Under full market coverage, the monopolist maximizes the expected profit: $E\pi = \bar{p} - a(1-q) - tVq^2 = (\bar{\theta} - 1)V - a(1-q) - tVq^2$. The first order condition for q is $\frac{\partial E\pi}{\partial q} = a - 2tVq = 0$. Recall that 2 denotes the case with liability. Then solving the first order condition yields the optimal security quality:

$$q_{2F}^{*} = \frac{a}{2tV}.$$

Note that

$$p_{2F}^{*} = (\bar{\theta} - 1)V.$$

The monopolist's expected profit in the fully covered market with liability is

$$E\pi_{2F}^{*} = (\bar{\theta} - 1)V - a(1 - \frac{a}{2tV}) - \frac{a^2}{4tV} = (\bar{\theta} - 1)V - a + \frac{a^2}{4tV}.$$

Proposition 4: No matter w*hich party is liable for the loss in the fully covered market, security quality* (q^{*}) *increases with the proportion of loss to cost* (a/t) *and decreases with baseline utility* (V). *When the risk is entirely on the customer, security quality increases with the marginal willingness to pay* $(\bar{\theta})$ *which does not affect security quality under the liability mechanism.*

Proof. See the Appendix.

The findings are consistent with the security experts' argument that it is harder to make the software with more features secure since breaches are more likely to be found in such software. Bruce Schneier, founder and chief technology officer of Internet security firm Counterpane said, "Complexity is the enemy of security. As systems get more complex, they get less secure," in an interview with PCWorld.com (Zetter 2001). Proposition 4 supports the argument that software vendors often sacrifice security quality to provide high functionality. We also find that when loss

is relatively large compared to cost $(a/t \uparrow)$, meaning that investment on security development is efficient, the monopolist offers high security quality no matter what mechanism is adopted. In other words, the vendor is willing to produce software with higher security quality when the same amount of investment is expected to prevent larger loss. Our results show the interplay of the marginal willingness to pay for the software and the optimal level of security quality. When there is no vendor liability, customers must value the software more to get higher security. However, once liability is imposed on the vendor, marginal willingness to pay does not affect security quality. This is because the customers do not appreciate security quality when the entire risk is on the vendor side.

Proposition 5: *When full market coverage obtains with or without a liability mechanism, imposing liability on the monopolist reduces security quality* $(q_{2F}^* < q_{1F}^*)$ *and increases price* $(p_{2F}^* > p_{1F}^*)$. *The liability mechanism allows higher profit to the monopolist when the marginal willingness to pay is low while it leads to lower profit when the marginal willingness to pay is high:*

$$E\pi_{2F}^* > E\pi_{1F}^* \quad \text{if} \quad 2 < \bar{\theta} < \frac{4tV}{a}$$
$$E\pi_{2F}^* < E\pi_{1F}^* \quad \text{if} \quad \bar{\theta} > \frac{4tV}{a}.$$

Proof. See the Appendix.

This is interesting in the sense that unlike the security experts' belief that the liability mechanism gives the software vendors an incentive to improve security quality (Armour and Humphrey 1993), imposing liability may discourage the vendor to develop security in the fully covered market. In the current market where customers are responsible for the entire risk, the expected loss directly affects customer valuation as well as vendor cost. When liability is

imposed on the vendor, customers do not have to care about the expected loss, and they do not appreciate the vendor's effort for security development. Thus, the expected loss only affects the vendor's development cost. Consequently, the vendor has a higher incentive to develop security to maximize its profit when customers are liable. When liability is imposed on the monopolist, it reduces the security quality but charges higher price for its software. The monopolist gets higher profit in the market with liability when the marginal willingness to pay of the customers is relatively low.

Proposition 6: *The monopolist covering partial market without liability may have an incentive to cover the full market with liability by increasing quality* $(q_{2F}^* > q_{1P}^*)$.

Proof. See the Appendix.

Recall that the monopolist who covers the partial market without liability may have an incentive to cover the full market once liability is imposed. Proposition 6 shows that the monopolist offers higher security quality in the full market with liability than in the partial market without liability. This result implies that the monopolist may want to minimize the expected loss by improving security quality to increase market coverage when it is liable for the loss. Since the loss directly affects the monopolist's profit in the market with liability, the monopolist may have an incentive to offer more secure software compared to the current market without liability where the loss is not a concern of the vendor.

3.5 Policy Implication

3.5.1 Welfare Analysis

We examine the social planner's decision on the security quality and investigate whether the monopolist offers socially optimal level of quality. First, we examine the case where no liability

is imposed on the software vendor. Unlike the monopolist who maximizes the profit, the social planner maximizes the social welfare by pricing at the marginal cost, which is zero. Recall that 1 denotes the case without liability. The social surplus can be written as the difference between the gross consumer surplus and the production cost:

$$S_1 = \int_{\bar{\theta}-1}^{\bar{\theta}} \theta(V - a(1-q))d\theta - tVq^2 = \frac{1}{2}(2\bar{\theta} - 1)(V - a(1-q)) - tVq^2.$$

The first order condition for q is

$$\frac{\partial S_1}{\partial q} = \frac{a}{2}(2\bar{\theta} - 1) - 2tVq = 0.$$

Let SP represent the social planner's solution. The optimal security quality of the social planner is as follows:

$$q_{1SP}^{*} = \frac{a(2\bar{\theta} - 1)}{4tV}.$$

We examine the market where liability is imposed on the software vendor. We consider the social planner who covers the full market, and maximizes the social surplus by pricing at the marginal cost which is the expected loss in this case. Recall that 2 denotes the case with liability. Then the social planner in the fully covered market with a liability mechanism maximizes the following social surplus:

$$S_2 = \int_{\bar{\theta}-1}^{\bar{\theta}} \theta V d\theta - a(1-q) - tVq^2 = \frac{1}{2}(2\bar{\theta} - 1)V - a(1-q) - tVq^2.$$

The first order condition for q is then

$$\frac{\partial S_2}{\partial q} = a - 2tVq = 0.$$

Thus, the optimal security quality of the social planner is

$$q_{2SP}^{*} = \frac{a}{2tV}.$$

Proposition 7: *When the market is fully covered, the social planner offers higher security quality in the market without liability than with liability. Without liability, security quality is underprovided by the monopolist compared to the social optimum. However, once liability is imposed, the monopolist offers socially optimal level of security quality.*

Proof. See the Appendix.

We find that the social planner offers higher security quality in the market without liability than with liability. This implies that the social planner may need to lower the security quality to take the responsibility for the entire risk. Proposition 7 provides an evidence of underprovided security quality of software under monopoly, as has been observed in the current market. It shows that imposing liability on the monopolist may give an incentive to offer socially optimal level of security quality. This result has a policy implication in that liability should be imposed on the vendor to achieve socially optimal level of security when the market is fully covered by the monopolist.

Recall that when the full market condition holds for both mechanisms, imposing liability may lead to worse security quality. We investigate whether a liability mechanism is beneficial to customers and to the entire society. Under the mechanism in the current market, customers are responsible for the entire risk which is not a concern for the vendor. However, once liability is imposed on the vendor, the customers do not take the loss into account. The comparison of consumer surplus is not trivial in that although liability mechanisms may make customers free from loss, it leads to higher price than the current market.

The consumer surplus in the market without liability is

$$CS_{1F} = \int_{\bar{\theta}-1}^{\bar{\theta}} \theta(V - a(1-q))d\theta - \bar{p} = \frac{1}{2}(V - a(1-q)).$$

Substituting q with the equilibrium quality leads to the following consumer surplus:

$$CS_{1F}^{*}=\begin{cases}\frac{1}{2}\left(V-a+\frac{a^{2}(\bar{\theta}-1)}{2tV}\right) & \text{if} \quad \frac{a(\bar{\theta}-1)}{2tV}<1 \\ \frac{1}{2}V & \text{if} \quad \frac{a(\bar{\theta}-1)}{2tV}\geq 1.\end{cases}$$

Consider the case where liability is imposed on the monopolist. Then the consumer surplus can be written as

$$CS_{2F}^{*}=\int_{\bar{\theta}-1}^{\bar{\theta}}\theta V d\theta-\bar{p}=\frac{1}{2}V.$$

The following proposition compares consumer surplus in both mechanisms.

Proposition 8: *When full market coverage obtains with or without liability, imposing liability on the monopolist may increase consumer surplus. When the marginal willingness to pay is relatively low* $(2<\bar{\theta}<\frac{4tV}{a})$, *the liability mechanism leads to higher social surplus as well.*

Proof. See the Appendix.

Schneier (2004) argues that the customers will pay for security once liability is imposed on the software vendor, which means security is not free even in the market where the vendor is liable. Since the vendor passes some portion of security development cost onto the customers, it is not trivial whether consumer surplus becomes higher because of customers being free from the loss or lower due to the increased price. In the fully covered market, we find that the monopolist reduces security quality when liable. Unlike security quality, consumer surplus becomes higher when liability is imposed on the vendor. Although the customers will pay for security, the results indicate that the customers are better off under the liability mechanism. When the marginal willingness to pay is low, the liability mechanism brings higher social surplus as well as higher consumer surplus.

3.5.2 Information Asymmetry

Varian (2000) argues that liability should be assigned to the party that can do the best job of managing risk. Other security experts who champion the liability mechanism argue that liability should be imposed on the software vendor since the vendor is the only party to have an ability to understand the vulnerabilities, to predict the possible damage due to the defects, and to know how to fix them (Ryan 2003). In other words, customers are not perfectly aware of security issues, and thus they are not able to predict possible loss as accurate as the software vendor. In this section, we examine the software market in the presence of information asymmetry between a software vendor and its customers. In other words, customers may be misinformed about the possible loss due to software vulnerability. We investigate whether the information asymmetry changes our results. According to the economics literature, federal regulation is effective when customers have misperception of the product characteristics (Spence 1977). We study how software liability, which is one form of regulation, works in the software market with misinformed customers. We model customer's misperception of the loss by incorporating a parameter *r*, which denotes security awareness of the customers. Then the expected loss perceived by the customers is $r(1-q)$.

So far, we investigated the case where $r = a$. Note that the parameter *r* does not affect the optimal quality in the market with liability since the entire loss is borne by the vendor, and thus it is not in the customer's utility function. After going through some algebra, we can obtain the optimal quality in the market without liability. The optimal security qualities are as follows:

$$q_{1F}^{*} = \frac{r(\bar{\theta} - 1)}{2tV}$$

$$q_{2F}^{*} = \frac{a}{2tV}.$$

Proposition 9: *When full market coverage obtains with or without liability and the customers underestimate the expected loss* ($r < \frac{a}{\bar{\theta} - 1}$), *imposing liability on the monopolist may increase security quality as well as consumer surplus.*

Proof. See the Appendix.

Proposition 9 supports security experts' arguments that imposing liability on the software vendor gives an incentive for security quality improvement especially when the customers are not capable of understanding the risks and predicting the loss due to software vulnerabilities (Ryan 2003, Schneier 2004, Varian 2000). We find that unlike the case where customers are fully aware of security issues, liability mechanism is effective in terms of quality improvement as well as consumer surplus increase. No matter how much customers are aware of security, a liability mechanism makes the customers better off by increasing consumer surplus. Moreover, particularly when the customers underestimate security concerns, the liability mechanism is what policy makers need to consider to have software products more secure.

3.6 Discussion: Endogenous Software Functionality

Recall that V is the baseline utility that a customer gains from the software and that V is assumed to be exogenous. We take V as a summary measure of all non-security aspects of software quality and we highlight security quality of software by modeling q in the model. In this section, we examine the case where V is also a choice of a software vendor. For example, among non-security features of software, functionality may seriously affect utility of customers. If V represents software functionality, in reality, the firm may choose V along with security quality q. In that case, the effect of the liability mechanism on security quality of software may depend on how security quality (q) and functionality (V) interact in the firm's production function. In this

section, we analyze the effect of endogenous software functionality on our findings to a certain extent. In reality, a software vendor decides functionality level first, then chooses security level next. Consider patching. Arora et al. (2006) argues that a software vendor's best strategy is to introduce the product to the market as early as possible then patch it later. To capture this aspect in our model, we analyze a four-stage game, in which the software vendor makes sequential decisions. At the first stage, the monopolist software vendor decides functionality level, V. At the second, the vendor chooses security quality level, q, and at the third, it sets up the price, p. Finally, customers decide whether or not to buy the software. Recall that the assumptions are

$$q \in [0,1], \quad \theta \sim \text{Uniform}(\bar{\theta}-1, \bar{\theta}), \quad \text{Expected Loss} = a(1-q), \quad \text{and} \quad \text{Cost} = tVq^2.$$

We also assume that $V \in [\underline{V} - \bar{V}]$ where $\underline{V} > 0$, implying that lowest possible functionality should be positive since the software provides negative utility otherwise and that maximum possible functionality is bounded.

Full Market Case

Consider the case without liability. In section 3.3.2, we show that the optimal security quality and price are as follows:

$$q_{1F}^* = \frac{a(\bar{\theta}-1)}{2tV} \quad \text{and} \quad p_{1F}^* = (\bar{\theta}-1)(V - a(1-\frac{a(\bar{\theta}-1)}{2tV})).$$

At the optimal security and price, the monopolist's expected profit in the fully covered market without liability is

$$E\pi_{1F}^* = (\bar{\theta}-1)(V-a) + \frac{a^2(\bar{\theta}-1)^2}{4tV}.$$

Consider the first stage at which the vendor chooses functionality level. The first order condition for V is

$$\frac{\partial E\pi_{1F}^{*}}{\partial V} = (\bar{\theta} - 1) - \frac{a^2(\bar{\theta} - 1)^2}{4tV^2} = 0.$$

The second order condition for V is

$$\frac{\partial^2 E\pi_{1F}^{*}}{\partial V^2} = \frac{a^2(\bar{\theta} - 1)^2}{2tV^3} > 0.$$

Thus, the profit maximization function has a boundary solution. Note the solution to the first order condition, $\frac{\partial E\pi_{1F}^{*}}{\partial V} = 0$, is $\frac{a}{2}\sqrt{\frac{\bar{\theta} - 1}{t}}$. Thus,

$$V_{1F}^{*} = \underline{V} \text{ if } \frac{\bar{V} - \underline{V}}{2} < \frac{a}{2}\sqrt{\frac{\bar{\theta} - 1}{t}} \text{ and } V_{1F}^{*} = \bar{V} \text{ if } \frac{\bar{V} - \underline{V}}{2} > \frac{a}{2}\sqrt{\frac{\bar{\theta} - 1}{t}}.$$

Rearranging the above conditions yields

$$a > \sqrt{\frac{t(\bar{V} - \underline{V})^2}{\bar{\theta} - 1}} \Rightarrow V_{1F}^{*} = \underline{V} \text{ and } q_{1F}^{*} = \frac{a(\bar{\theta} - 1)}{2t\underline{V}}$$

$$a < \sqrt{\frac{t(\bar{V} - \underline{V})^2}{\bar{\theta} - 1}} \Rightarrow V_{1F}^{*} = \bar{V} \text{ and } q_{1F}^{*} = \frac{a(\bar{\theta} - 1)}{2t\bar{V}}.$$

Now, consider the case with liability. The optimal security quality and price are shown in section 3.4.2:

$$q_{2F}^{*} = \frac{a}{2tV} \quad \text{and} \quad p_{2F}^{*} = (\bar{\theta} - 1)V.$$

The monopolist's expected profit is

$$E\pi_{2F}^{*} = (\bar{\theta} - 1)V - a + \frac{a^2}{4tV}.$$

At the first stage, the software vendor chooses V to maximize its expected profit. The corresponding first order condition for V is

$$\frac{\partial E\pi_{2F}^{*}}{\partial V} = (\bar{\theta} - 1) - \frac{a^2}{4tV^2} = 0.$$

The second order condition for V is

$$\frac{\partial^2 E\pi_{2F}^*}{\partial V^2} = \frac{a^2}{2tV^3} > 0.$$

Similar to the no liability case, the profit maximization function has a boundary solution. The solution to the first order condition is $\frac{a}{2}\sqrt{\frac{1}{t(\bar{\theta}-1)}}$. Thus,

$$a > \sqrt{t(\bar{\theta}-1)(\bar{V}-\underline{V})^2} \Rightarrow V_{2F}^* = \underline{V} \text{ and } q_{2F}^* = \frac{a}{2t\underline{V}}$$

$$a < \sqrt{t(\bar{\theta}-1)(\bar{V}-\underline{V})^2} \Rightarrow V_{2F}^* = \bar{V} \text{ and } q_{2F}^* = \frac{a}{2t\bar{V}}.$$

Suppose that $\bar{\theta} > 2$, which is a full market condition for the previous analysis where V is exogenous. As will be explained in the next section, obtaining full market condition in a closed form with endogenous V is not feasible. Thus, we show that our previous result for quality comparison for full market holds for reasonably high θ. Since $\bar{\theta} > 2$ by assumption, $\bar{\theta} - 1 > \frac{1}{\bar{\theta}-1}$. Thus, $\sqrt{t(\bar{\theta}-1)(\bar{V}-\underline{V})^2} > \sqrt{\frac{t(\bar{V}-\underline{V})^2}{\bar{\theta}-1}}$.

Case 1: Small Expected Loss $(a < \sqrt{\frac{t(\bar{V}-\underline{V})^2}{\bar{\theta}-1}})$

In this case, the monopolistic vendor provides maximum functionality, that is, $V_{1F}^* = \bar{V} = V_{2F}^*$. Security quality comparison is as follows:

$$q_{1F}^* = \frac{a(\bar{\theta}-1)}{2t\bar{V}} > \frac{a}{2t\bar{V}} = q_{2F}^*.$$

Case 2: Medium Expected Loss $(\sqrt{\frac{t(\bar{V}-\underline{V})^2}{\bar{\theta}-1}} < a < \sqrt{t(\bar{\theta}-1)(\bar{V}-\underline{V})^2})$

When the possible expected loss is medium, $V_{1F}^* = \underline{V} < \bar{V} = V_{2F}^*$ and the optimal security quality is

given by

$$q_{1F}^{*} = \frac{a(\bar{\theta}-1)}{2t\underline{V}} > \frac{a}{2t\bar{V}} = q_{2F}^{*}.$$

Case 3: Large Expected Loss $(a > \sqrt{t(\bar{\theta}-1)(\bar{V}-\underline{V})^2})$

When the expected loss can be large, the software vendor offers minimum functionality, that is, $V_{1F}^{*} = \underline{V} = V_{2F}^{*}$. Security quality is

$$q_{1F}^{*} = \frac{a(\bar{\theta}-1)}{2t\underline{V}} > \frac{a}{2t\underline{V}} = q_{2F}^{*}.$$

By examining all the cases possible, we find that security quality decreases when liability is imposed on the monopolistic software vendor covering full market with customers who have fairly high willingness to pay. The result from the endogenous functionality case is consistent with our previous findings. This justifies robustness of our result to a certain extent. In other words, endogenizing functionality does not affect our result on security quality comparison in the full market, which is a basis for the analysis throughout the paper.

Partial Market Case

In this section, with endogenous functionality, we derive the optimal security quality in the partially covered market and discuss why obtaining clear-cut full market condition is difficult. Consider no liability case first. In section 3.3.1, we derive the optimal security quality and price,

$$q_{1P}^{*} = \frac{a\bar{\theta}^2}{8tV} \quad \text{and} \quad p_{1P}^{*} = \frac{\bar{\theta}}{2}(V - a(1 - \frac{a\bar{\theta}^2}{8tV})).$$

The expected profit at the optimal q_{1P}^{*} and p_{1P}^{*} is

$$E\pi_{1P}^{*} = \frac{\bar{\theta}^2}{4}V - \frac{a\bar{\theta}^2}{4} + \frac{a^2\bar{\theta}^4}{64tV}.$$

Consider the first stage at which the vendor chooses optimal functionality. The first order

condition for V is

$$\frac{\partial E\pi_{1P}^*}{\partial V} = \frac{\bar{\theta}^2}{4} - \frac{a^2\bar{\theta}^4}{64tV^2} = 0.$$

The second order condition for V is

$$\frac{\partial^2 E\pi_{1P}^*}{\partial V^2} = \frac{a^2\bar{\theta}^4}{32tV^3} > 0.$$

Similar to the full market case, the expected profit is maximized at boundary values of V. Note that the solution to the first order condition is $\frac{a\bar{\theta}}{4\sqrt{t}}$. Thus,

$$V_{1P}^* = \underline{V} \text{ if } \frac{\bar{V}-\underline{V}}{2} < \frac{a\bar{\theta}}{4\sqrt{t}} \text{ and } V_{1P}^* = \bar{V} \text{ if } \frac{\bar{V}-\underline{V}}{2} > \frac{a\bar{\theta}}{4\sqrt{t}}.$$

Rearranging the above conditions yields

$$a > \frac{2\sqrt{t}(\bar{V}-\underline{V})}{\bar{\theta}} \Rightarrow V_{1P}^* = \underline{V} \text{ and } q_{1P}^* = \frac{a\bar{\theta}^2}{8t\underline{V}}$$
$$a < \frac{2\sqrt{t}(\bar{V}-\underline{V})}{\bar{\theta}} \Rightarrow V_{1P}^* = \bar{V} \text{ and } q_{1P}^* = \frac{a\bar{\theta}^2}{8t\bar{V}}.$$

Now, we examine the case with liability. In section 3.4.1, we show that $p_{2P}^* = \frac{V\bar{\theta} + a(1-q)}{2}$ and that the optimal security quality is as follows:

$$q_{2P}^* = 0 \text{ and } a > 2\sqrt{t}V \text{ and } \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} > \frac{1}{2}$$
$$q_{2P}^* = 1 \text{ and } a > 2\sqrt{t}V \text{ and } \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} < \frac{1}{2}$$
$$q_{2P}^* = \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} \text{ when } a < 2\sqrt{t}V.$$

Case 1: Boundary Solution ($q_{2P}^* = 0$)

Note that the expected profit at the optimal price, $p_{2P}^* = \frac{V\bar{\theta} + a(1-q)}{2}$ is

$$E\pi_{2P} = \frac{1}{4V}(V\bar{\theta} - a(1-q))^2 - tVq^2.$$

Since $q_{2P}^* = 0$, $E\pi_{2P}^* = \frac{1}{4V}(V\bar{\theta} - a)^2$. Going through algebra shows the following optimal functionality:

$$V_{2P}^* = \underline{V} \text{ if } \frac{\bar{V} - \underline{V}}{2} < \frac{a}{\bar{\theta}} \text{ and } V_{2P}^* = \bar{V} \text{ if } \frac{\bar{V} - \underline{V}}{2} > \frac{a}{\bar{\theta}}.$$

Case 2: Boundary Solution ($q_{2P}^* = 1$)

In this case, the expected profit is given by

$$E\pi_{2P}^* = \frac{1}{4V}(V\bar{\theta})^2 - tV.$$

Since the profit function is linear in V, the optimal functionality is

$$V_{2P}^* = \underline{V} \text{ if } \bar{\theta} > 2\sqrt{t} \text{ and } V_{2P}^* = \bar{V} \text{ if } 1 < \bar{\theta} < 2\sqrt{t}.$$

Case 3: Interior Solution ($q_{2P}^* = \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2}$)

The interior case is not trivial. The expected profit at the optimal security quality is

$$E\pi_{2P}^* = \frac{1}{4V}\left(V\bar{\theta} - a + \frac{a^2(a - V\bar{\theta})}{a^2 - 4tV^2}\right)^2 - tV\left(\frac{a^2(a - V\bar{\theta})}{a^2 - 4tV^2}\right)^2.$$

Let $K(V) = \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2}$, $K'(V) = \frac{\partial K(V)}{\partial V}$, and $K''(V) = \frac{\partial^2 K(V)}{\partial V^2}$. Then

$$E\pi_{2P}^* = \frac{1}{4V}\left(V\bar{\theta} - a + a^2K(V)\right)^2 - a^4tVK(V)^2.$$

The optimal functionality denoted by V_{2P}^* is V which satisfies the following first order and second order conditions:

$$\frac{\partial E\pi_{2P}^{*}}{\partial V} = \frac{\bar{\theta}^2}{4} - \frac{a^2}{4V^2} + 2a^4K(V)K'(V)(\frac{1}{4V} - tV) - a^4K(V)^2(\frac{1}{4V} + t) + \frac{a^2}{2}K'(V)(\bar{\theta} - \frac{a}{V}) + \frac{a^2}{2}K(V)(\frac{a}{V^2}) = 0$$

$$\frac{\partial^2 E\pi_{2P}^{*}}{\partial V^2} = \frac{a^2}{2V^3} + 2a^4K'(V)^2(\frac{1}{4V} - tV) + 2a^4K(V)K''(V)(\frac{1}{4V} - tV) - 4a^4K(V)K'(V)(\frac{1}{4V^2} + t)$$
$$+ a^4K(V)^2(\frac{1}{2V^3}) + \frac{a^2}{2}K''(V)(\bar{\theta} - \frac{a}{V}) + a^2K'(V)(\frac{a}{V^2}) - a^2K(V)(\frac{a}{V^3}) < 0.$$

In this section, we show that profit maximization function for a monopolist software vendor in the partial market case with liability may have an interior solution for V unlike other cases that lead to boundary solution. Since the optimal functionality for the interior q_{2P}^{*} case is not trivial, obtaining clear-cut full market condition is not tractable.

3.7 Concluding Remarks

Low quality of software has been blamed for poor security of computer networks in the sense that the defects of poorly written software are exploited by malicious hackers to attack the computer systems. As an incentive mechanism for software security quality improvement, software liability has been intensely discussed among computer scientists, jurists and policy makers (Armour and Humphrey 1993, Harmon 2003, Heckman 2003, Ryan 2003, Schneier 2004, Varian 2000). Unfortunately, not much research has been done on this issue from an economic perspective. In this paper, we analyze the software market dominated by a monopolistic vendor, and investigate whether imposing liability motivates the monopolist to improve the security quality of the software, and makes the customers and the entire society better off. We model the case where the vendor focuses on security quality improvement and both the vendor and customers are fully aware of security issues. In other words, both parties have rational expectations of the loss caused by security breaches of the software. Unlike direct contractual

risk-sharing that may impose partial liability on the vendor, we examine the legal and uniform liability that makes the vendor fully responsible for the entire loss incurred by security breaches of its software. We identify the factors that affect the monopolist's decision on the security quality.

We determine the conditions under which the monopolist has incentive to cover the full market. We find that high marginal willingness to pay leads to full market coverage when the customers are responsible for the entire risk. Once liability is imposed on the monopolist, full market coverage obtains for any level of marginal willingness to pay depending on conditions. Our welfare analysis shows that the social planner offers higher security quality in the market without liability than with liability and that security quality is underprovided by the monopolist in the market where customers are responsible for the entire risk, while socially optimal level of security quality is offered once liability is imposed on the monopolist. Interestingly, our results indicate that imposing liability may discourage the monopolist improving security contrary to security practitioners' expectation, whereas consumer surplus is higher with a liability mechanism. When the marginal willingness to pay is relatively low, the liability mechanism leads to higher social surplus. Finally, we consider the case where customers have poor security awareness. We find that the liability mechanism leads to higher security quality as well as higher consumer surplus when customers are not fully aware of security issues, and thus underestimate the associated risk. Then we analyze our model with endogenous software functionality. We find that endogenizing software functionality as well as security quality does not affect our basic results.

Several limitations of our study deserve further discussion. First, our model does not consider patch management which can be an alternative way of achieving security. Setting up a

multi-period model and analyzing how the endogenous patching quality can affect the initial security quality will be an interesting question for future research. Second, the implicit assumption in our model is that the expected loss can be estimated. However, this is not an easy problem. Quantifying loss itself is a hard problem and many researchers have been working on this issue, but there is no standardized method as yet. Security experts argue that the prerequisite for the software liability mechanism is measuring the loss and estimating the likelihood. Despite these limitations, our paper contributes to the literature in the following ways. First, our paper analyzes the problem of software liability from an economic perspective where not much economic research has been done on this issue although it has been intensely argued by computer scientist, jurists, and policy makers. Second, we provide a clear picture of the software market, and give a guideline to managers and policy makers who try to make software secure. Our results help policy makers decide which mechanism they need to choose under certain conditions. Third, our paper explains the huge security investment of the monopolistic software vendors by identifying the factors that affect the vendors' decision on security quality such as baseline utility of the software product, security awareness of the customers, and the potential threat to be liable for the loss caused by the software vulnerabilities.

Appendix

Proof of Proposition 1

Let $Y = \frac{L'(q)}{C'(q,V)}$. Then

$$\frac{\partial Y}{\partial q} = \frac{L''(q)}{C''(q,V)} - \frac{L'(q)}{C'(q,V)^2} C''(q,V) = \frac{L'(q)}{C'(q,V)} \left(\frac{L''(q)}{L'(q)} - \frac{C''(q,V)}{C'(q,V)} \right).$$

Note that the cost function is convex and the loss function is convex. That is, $C'(q,V) > 0, C''(q,V) > 0, L'(q) < 0,$ and $L''(q) > 0.$ Thus, $\frac{L'(q)}{C'(q,V)} < 0, \frac{L''(q)}{L'(q)} < 0,$ and $\frac{C''(q,V)}{C'(q,V)} > 0.$

Therefore, $\frac{\partial Y}{\partial q} > 0,$ that is, $\frac{L'(q)}{C'(q,V)}$ increases with q. Recall that

$$\frac{L'(q_{1F}^*)}{C'(q_{1F}^*,V)} = -\frac{1}{\underline{\theta}} \quad \text{and} \quad \frac{L'(q_{2F}^*)}{C'(q_{2F}^*,V)} = -1.$$

Since $\frac{L'(q)}{C'(q,V)}$ increases with q, $q_{1F}^* > q_{2F}^*$ if $\underline{\theta} > 1$. Otherwise $(\underline{\theta} < 1)$, $q_{1F}^* < q_{2F}^*$. ***QED.***

Proof of Proposition 2

Let $k = a/t$. Then $\frac{\partial q_{1P}^*}{\partial k} = \frac{\bar{\theta}^2}{8V} > 0$, $\frac{\partial q_{1P}^*}{\partial \theta} = \frac{a\bar{\theta}}{4tV} > 0$, and $\frac{\partial q_{1P}^*}{\partial V} = -\frac{a\bar{\theta}^2}{8tV^2} < 0$. Assume that $a < 2\sqrt{tV}$.

Then $\frac{\partial q_{2P}^*}{\partial \theta} = \frac{aV}{4tV^2 - a^2} > 0$ since $a^2 < 4tV^2$. Suppose that $a > 2\sqrt{tV}$. Then

$$\frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} > \frac{1}{2} \quad \Rightarrow \quad q_{2P}^* = 0$$
$$\frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} < \frac{1}{2} \quad \Rightarrow \quad q_{2P}^* = 1.$$

Case 1: High Marginal Willingness to Pay ($\bar{\theta} > \frac{a}{V}$)

Since $a^2 - 4tV^2 > 0$ and $a^2 - aV\bar{\theta} < 0, \frac{a^2 - aV\bar{\theta}}{a^2 - 4tV^2} < 0 < \frac{1}{2}$. Thus, $q_{2P}^* = 1$.

Case 2: Medium Marginal Willingness to Pay ($2\sqrt{tV} < \bar{\theta} < \frac{a}{V}$)

The conditions for boundary solutions can be arranged as follows:

$$a^2 - 2aV\bar{\theta} + 4tV^2 > 0 \;\Rightarrow\; q_{2P}^* = 0$$
$$a^2 - 2aV\bar{\theta} + 4tV^2 < 0 \;\Rightarrow\; q_{2P}^* = 1.$$

Consider the equation: $a^2 - 2aV\bar{\theta} + 4tV^2 = 0$. The solution to the equation is either $V(\bar{\theta} - \sqrt{\bar{\theta}^2 - 4t})$ or $V(\bar{\theta} - \sqrt{\bar{\theta}^2 - 4t})$. Since $a > V\bar{\theta}$ by assumption, $a > V(\bar{\theta} - \sqrt{\bar{\theta}^2 - 4t})$. Thus,

$$a > V(\bar{\theta} + \sqrt{\bar{\theta}^2 - 4t}) \;\Rightarrow\; q_{2P}^* = 0$$
$$V\bar{\theta} < a < V(\bar{\theta} + \sqrt{\bar{\theta}^2 - 4t}) \;\Rightarrow\; q_{2P}^* = 1.$$

Case 3: Low Marginal Willingness to Pay ($\bar{\theta} < 2\sqrt{tV}$)

Note that $\bar{\theta}^2 - 4t < 0$. Thus, $a^2 - 2aV\bar{\theta} + 4tV^2 > 0$. Therefore $q_{2P}^* = 0$. ***QED.***

Proof of Proposition 3

In section 3.3.2., we show that the full market condition without liability is $\bar{\theta} > 2$. Suppose that $a < 2\sqrt{tV}$. Then the optimal quality is an interior solution. Note that full market coverage obtains when $\frac{(\bar{\theta}V - 2V)(a^2 - 4tV^2) - a^2V\bar{\theta} + 4atV^2}{a^2 - 4tV^2} \geq 0$.

Case 1: High Marginal Willingness to Pay ($\bar{\theta} > 2 + \frac{t}{2}$)

Note that $a^2 - 4tV^2 < 0$. The full market condition can be written as

$$(\bar{\theta}V - 2V)(a^2 - 4tV^2) - a^2V\bar{\theta} + 4atV^2 \leq 0 \Leftrightarrow a^2 - 2atV + 2tV^2(\bar{\theta} - 2) \geq 0.$$

Consider the equation: $a^2 - 2atV + 2tV^2(\bar{\theta} - 2) = 0.$ The solutions are

$$a_- = tV\left(1 - \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right) \text{ and } a_+ = tV\left(1 + \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right).$$

Note that $\bar{\theta} > 2 + \frac{t}{2} \Leftrightarrow 1 - \frac{2}{t}\bar{\theta} + \frac{4}{t} < 0.$ Thus, a_- and a_+ do not exist, meaning that the full market condition always holds: $a^2 - 2atV + 2tV^2(\bar{\theta} - 2) > 0.$

Case 2: Medium Marginal Willingness to Pay ($2 < \bar{\theta} < 2 + \frac{t}{2}$)

Note that $2 < \bar{\theta} < 2 + \frac{t}{2}.$ Thus, $0 < \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}} < 1 \Leftrightarrow 0 < a_- = tV\left(1 - \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right) < tV.$

Then $a^2 - 4tV^2 < 0.$ The full market condition is $a^2 - 2atV + 2tV^2(\bar{\theta} - 2) \geq 0.$ The monopolist covers the full market when $a \leq a_-$ or $a \geq a_+.$ Thus, the full market conditions are:

$$a_+ > 2\sqrt{t}V, \quad a \in (0, \text{Min}\{2\sqrt{t}V, a_-\})$$
$$a_+ < 2\sqrt{t}V, \quad a \in (0, a_-) \text{ or } a \in (a_+, 2\sqrt{t}V).$$

Case 3: Low Willingness to Pay ($1 < \bar{\theta} < 2$)

Since $1 < \bar{\theta} < 2,\ 1 < \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}} < \sqrt{1 + \frac{2}{t}}.$ Then we have $tV\left(1 - \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right) < 0 \Leftrightarrow a_- < 0.$

Note that $a^2 - 4tV^2 < 0.$ The full market condition is $a^2 - 2atV + 2tV^2(\bar{\theta} - 2) \geq 0.$ The monopolist covers the full market when $a_+ < 2\sqrt{t}V$ and $a \in (a_+, 2\sqrt{t}V).$ Therefore, the monopolist has an incentive to cover the full market for any level of marginal willingness to pay. ***QED.***

Proof of Proposition 4

Note that $q_{1F}^* = \frac{a(\bar{\theta} - 1)}{2tV}$ and $q_{2F}^* = \frac{a}{2tV}.$ Let $k = a/t.$ Then

$$\frac{\partial q_{1F}^*}{\partial k}=\frac{(\bar{\theta}-1)}{2V}>0,\quad \frac{\partial q_{1F}^*}{\partial V}=-\frac{a(\bar{\theta}-1)}{2tV^2}<0,\quad \frac{\partial q_{1F}^*}{\partial \bar{\theta}}=\frac{a}{2tV}>0,$$

and

$$\frac{\partial q_{2F}^*}{\partial k}=\frac{1}{2V}>0,\quad \frac{\partial q_{2F}^*}{\partial V}=-\frac{a}{2tV^2}<0.\ \textbf{\textit{QED.}}$$

Proof of Proposition 5

Note that $q_{1F}^*=\frac{a(\bar{\theta}-1)}{2tV}$ and $q_{2F}^*=\frac{a}{2tV}$. Thus, $q_{1F}^*=1$ if $\frac{a(\bar{\theta}-1)}{2tV}\geq 1$, and $q_{2F}^*=1$ if $\frac{a}{2tV}\geq 1$. No matter which mechanism is adopted in the market, the monopolist covers the full market when

$$\bar{\theta}>2+\frac{t}{2},\quad a\in(0,2\sqrt{t}V)$$
$$2<\bar{\theta}<2+\frac{t}{2},\quad a_+>2\sqrt{t}V,\quad a\in(0,\text{Min}\{2\sqrt{t}V,a_-\})$$
$$2<\bar{\theta}<2+\frac{t}{2},\quad a_+<2\sqrt{t}V,\quad a\in(0,a_-)\ \text{ or }\ a\in(a_+2\sqrt{t}V)$$

where $a_-=tV\left(1-\sqrt{1-\frac{2}{t}\bar{\theta}+\frac{4}{t}}\right)$ and $a_+=tV\left(1+\sqrt{1-\frac{2}{t}\bar{\theta}+\frac{4}{t}}\right)$. Since $\bar{\theta}\geq 2$,

$$1<\frac{a}{2tV}\ \Rightarrow\ q_{1F}^*=q_{2F}^*=1$$
$$\frac{a}{2tV}\leq 1\leq\frac{a(\bar{\theta}-1)}{2tV}\ \Rightarrow\ q_{1F}^*=1\geq\frac{a}{2tV}=q_{2F}^*$$
$$\frac{a(\bar{\theta}-1)}{2tV}<1\ \Rightarrow\ q_{1F}^*=\frac{a(\bar{\theta}-1)}{2tV}\geq\frac{a}{2tV}=q_{2F}^*.$$

Therefore, $q_{1F}^*\geq q_{2F}^*$. Also, note that $p_{1F}^*=(\bar{\theta}-1)(V-a(1-q_{1F}^*))$ and $p_{1F}^*=(\bar{\theta}-1)V$. Then $p_{2F}^*-p_{1F}^*=a(1-q_{1F}^*)$. Since q_{1F}^* is bounded by 1, $a(1-q_{1F}^*)\geq 0$. Therefore, $p_{1F}^*\leq p_{2F}^*$.

Recall that $E\pi_{1F}^{*} = (\bar{\theta} - 1)(V - a) + \frac{a^2(\bar{\theta} - 1)}{4tV}$ and $E\pi_{2F}^{*} = (\bar{\theta} - 1)V - a + \frac{a^2}{4tV}$. Thus, $E\pi_{2F}^{*} - E\pi_{1F}^{*} = a(\bar{\theta} - 2)(\frac{a}{4tV}\bar{\theta} - 1)$. Therefore, $E\pi_{1F}^{*} < E\pi_{2F}^{*}$ if $2 < \bar{\theta} < \frac{4tV}{a}$. Otherwise $(\bar{\theta} > \frac{4tV}{a})$, $E\pi_{1F}^{*} > E\pi_{2F}^{*}$. ***QED.***

Proof of Proposition 6

Recall that the monopolist covers the partial market without liability when $1 < \bar{\theta} < 2$. With liability, full market obtains for any $1 < \bar{\theta} < 2$, when the following conditions hold:

$$a_{+} < 2\sqrt{tV} \text{ and } a \in (a_{+}, 2\sqrt{tV}) \text{ where } a_{+} = tV\left(1 + \sqrt{1 - \frac{2}{t}\bar{\theta} + \frac{4}{t}}\right).$$

Suppose that $1 < \bar{\theta} < 2$, and that he full market condition holds for the case with liability. Note that $q_{1P}^{*} = \frac{a\bar{\theta}^2}{8tV}$ and $q_{2F}^{*} = \frac{a}{2tV}$. Since $1 < \bar{\theta} < 2$, $q_{1P}^{*} = \frac{a\bar{\theta}^2}{8tV} < \frac{a}{2tV} = q_{2F}^{*}$. ***QED.***

Proof of Proposition 7

The monopolist covers the full market no matter whether liability is imposed or not if the following conditions hold:

$$\bar{\theta} > 2 + \frac{t}{2}, \quad a \in (0, 2\sqrt{tV})$$

$$2 < \bar{\theta} < 2 + \frac{t}{2}, \quad a_{+} > 2\sqrt{tV}, \quad a \in (0, \text{Min}\{2\sqrt{tV}, a_{-}\})$$

$$2 < \bar{\theta} < 2 + \frac{t}{2}, \quad a_{+} < 2\sqrt{tV}, \quad a \in (0, a_{-}) \text{ or } a \in (a_{+} 2\sqrt{tV})$$

where $a_- = tV\left(1-\sqrt{1-\frac{2}{t}\bar{\theta}+\frac{4}{t}}\right)$ and $a_+ = tV\left(1+\sqrt{1-\frac{2}{t}\bar{\theta}+\frac{4}{t}}\right)$. Note that $\bar{\theta} > 2$. Thus,

$$q^*_{1SP} = \frac{a(\bar{\theta}-\frac{1}{2})}{2tV} > \frac{a}{2tV} = q^*_{2SP}.$$

Then the following completes the proof: $q^*_{1F} = \frac{a(\bar{\theta}-1)}{2tV} < \frac{a(\bar{\theta}-\frac{1}{2})}{2tV} = q^*_{1SP}$ and $q^*_{2F} = \frac{a}{2tV} = q^*_{2SP}$.

QED.

Proof of Proposition 8

Note that

$$\begin{aligned}
&\frac{a(\bar{\theta}-1)}{2tV} \geq 1 \;\Rightarrow\; CS^*_{1F} = \frac{V}{2} = CS^*_{2F} \\
&\frac{a(\bar{\theta}-1)}{2tV} < 1 \;\Rightarrow\; CS^*_{1F} - CS^*_{2F} = \frac{a}{2}\left(\frac{a(\bar{\theta}-1)}{2tV}-1\right) < 0 \\
&\qquad\qquad \Leftrightarrow\; CS^*_{1F} < CS^*_{2F}.
\end{aligned}$$

Thus, $CS^*_{1F} \leq CS^*_{2F}$. In Proposition 5, we show that $E\pi^*_{1F} < E\pi^*_{2F}$ if $2 < \bar{\theta} < \frac{4tV}{a}$. Therefore, $S^*_{1F} < S^*_{2F}$ if $2 < \bar{\theta} < \frac{4tV}{a}$. ***QED.***

Proof of Proposition 9

Note that $q^*_{2F} - q^*_{1F} = \frac{1}{2tV}(a - r(\bar{\theta}-1))$. Thus, $q^*_{1F} < q^*_{2F}$ when $r < \frac{a}{\bar{\theta}-1}$. ***QED.***

References

Armour, J., W. S. Humphrey. 1993. Software Product Liability. Technical Report CMU/SEI-93-TR-13. Software Engineering Institute. Carnegie Mellon University.

Arora, A., J. Caulkins, R. Telang. 2006. Sell First, Fix Later: Impact of Patching on Software Quality. *Management Science* 52(3) 465–471.

Ernst & Young. 2004. Global Information Security Survey 2004. http://www.ey.com.

Fisher, D. 2002. Contracts Getting Tough on Security. eWeek (April 15).

Fisk, M. 2002. Causes & Remedies for Social Acceptance of Network Insecurity. *1st Workshop on Economics and Information Security*, Berkeley, CA.

Harmon, A. 2003. Digital Vandalism Spurs a Call for Oversight. *New York Times* (September 1).

Heckman, C. 2003. Two Views on Security Software Liability: Using the Right Legal Tools. *IEEE Security & Privacy* 1(1) 73–75.

Mussa, M. and S. Rosen. 1978. Monopoly and Product Quality. *Journal of Economic Theory* 18 301–317.

Ronnen, U. 1991. Minimum Quality Standards, Fixed Costs, and Competition. *RAND Journal of Economics* 22 490–504.

Ryan, D. J. 2003. Two Views on Security Software Liability: Let the Legal System Decide. *IEEE Security & Privacy* 1(1) 70–72.

Sager, I., J. Green. 2002. The Best Way to Make Software Secure: Liability. *Business Week* (March 18) 61.

Schneier, B. 2004. Information Security: How Liable Should Vendors Be? *Computerworld* (Octover 28).

Spence, M. 1975. Monopoly, Quality, and Regulation. *Bell Journal of Economics* 6 417–429.

Spence, M. 1977. Consumer Misperceptions, Product Failure and Producer Liability. *The Review of Economic Studies* 44(3) 561-572.

Yurcik, W., D. Doss. 2002. Cyberinsurance: A Market Solution to the Internet Security Market Failure. *1st Workshop on Economics and Information Security*, Berkeley, CA.

Varian, H. R. 2000. Managing Online Security Risks. *New York Times* (June 1).

Verton, D. 2003. Software Failure Cited in Blackout Investigation. *Computerworld* (November 24).

Zetter, K. 2001. Three Minutes With Security Expert Bruce Schneier. *PCWorld* (September 28).

4 Pricing Open Source Software

4.1 Introduction

Open Source Software (OSS) is software for which the source code is available to the public, enabling anyone to copy, modify and redistribute the source code (Varian and Shapiro 2003). Commercial software, by contrast, is software that is distributed under commercial license agreements, usually for a fee. Open source was a service mark of the Open Source Initiative (OSI), a non-profit organization that continues to provide an official "Open Source Definition". According to OSI definition, open source software is software whose source code can be freely modified and redistributed.[2] The redistribution rights do not preclude a company selling such software for profit.

Recently, open source software is getting more attention from not only software developers but also users including governments and large enterprises. According to the Economist (2003), many countries in European Union and Asia are funding open source software initiatives outright. For example, Japan announced that it would collaborate with China and South Korea to develop open source alternatives to Microsoft's software, and it has already allocated nine million US dollars to the project. According to BBC, the Brazilian government announced that it would adopt open source software. As a result, Brazil's government ministries and state-run enterprises are abandoning Windows in favor of open source software, like Linux (Kingstone 2005). Open source adoption is rising in the industry sector as well. A recent survey by Forrester Research shows that 46 percent of the 140 North American firms were using open source software and 14 percent had a plan to use it in 2004. In 2005, open source usage was even higher. Among 128

[2] The Open Source Definition. http://opensource.org/docs/definition.php

respondents, 56% were using open source software and 19% were planning (Gloude 2005). Especially, Linux is identified as the most widely used open source software and other widely adopted open source software includes Apache, MySQL, and JBoss.

Why are firms using open source software? The Free/Libre and Open Source Software (FLOSS) Survey 2002 found that enterprises are adopting open source software on the grounds of higher stability, cost savings, and flexibility due to the open and modifiable source code (Wichmann 2002). Independence from pricing and licensing policies of big software companies and better availability of IT specialists are also considered to be motivations for open source software usage. Another survey by the International Data Corporation (IDC) shows that firms are using open source software because of its quality and flexibility, rather than merely considering it "good enough" because it is inexpensive (Broersma 2005). Such firms consider the customizability of open source software to be important since they believe that vendors of pre-packaged, proprietary software routinely downplay the customizability of open source software, arguing customers are not interested in extending software functions themselves. On the other hand, a user survey by Forrester Research finds that the top barriers to open source adoption by firms include the lack of support and the availability of applications (Reid 2004). Companies are also concerned about not having skilled workers to support or use open source applications, licensing costs, and security. Another survey by Actuate, conducted in 2006, reflects opinions from 141 respondents in the financial services sector. It shows that the majority of respondents cited the availability of long term support (58.2%) as the main barrier to adopting open source software and the availability of long-term maintenance was a close second-place barrier (44.7%). The benefits include cost savings (55%), vendor independence (49.3%) and flexibility (47.1%) (McCarthy 2006).

Lately a new movement in the open source software industry has been observed. As some open source software products become popular and their market shares approach critical mass, the vendors of such software seek to capitalize on the publicity and popularity of their software products. Two approaches have been successful in commercializing open source software: the dual-licensing model and the support model. With the dual-licensing model, a software vendor offers the very same software under two different licenses. Open source license allows the licensees to modify, distribute, and use the software for free, but it requires the release of any modifications under the same open source license. The commercial license permits using and developing the software under standard commercial terms. MySQL and Openoffice are the examples of the dual-licensing model. A second approach is the support model, which allows users to get the software for free, but the software vendor charges users for support. Although the viability of this model is still questioned, Red Hat and JBoss have been successful commercializing their products by adopting the support model. There is a lively debate going on among industry experts about the viability and the scalability of the commercial model of the open source software (Moczar 2005, Vaughan-Nichols 2005).

To our best knowledge, no academic research has examined the issue of open source software pricing from an economic perspective. In this paper, we examine the optimal pricing strategies of both the commercial and the open source software vendors and find the conditions under which a for-profit open source pricing model is viable. We consider only the businesses as the customers in this paper since most of the profitable customers of for-profit open source software are not individuals but firms. Our model captures two sources of customer heterogeneity in their valuation of software: taste and tech-savviness. Customers value the same software differently. For example, a company in an industry using information technology

heavily may have higher valuation of the software than other firms since a significant portion of their business may depend on information systems powered by the software. Taste parameter captures this heterogeneity in customer taste. We consider two types of customers in terms of their tech-savviness. Along with cost savings, benefits from customizability and flexibility are considered as the top reasons for open source adoption. When an in-house IT management team with skilled workers is available, the firm can enjoy such benefits since it has the capability of customizing and managing the open source software. When a firm does not have such capability of managing technology, open source software incurs significant support cost, which is cited as the main barrier to open source adoption. We consider the firms with a strong in-house IT management team to be high-type customers and the others to be low-type customers. Reflecting reality, we assume that high-type customers enjoy benefit from flexibility and customizability whereas low-type customers suffer from support cost when they use open source software.

We compare three different pricing mechanisms: commercial software, open source dual-licensing, and open source support. We first examine the optimal pricing strategies for each pricing scheme under monopoly setting. We find that the monopoly price for open source software under dual-licensing is higher than the commercial software price, hence higher profit, if the total flexibility benefit for the high-type customers is higher than total support cost for the low-type customers. Under the support model, the open source vendor charges for support at higher price than the commercial software price, but it makes lower profit than the commercial software vendor. Then we investigate whether the open source software vendor prefers a certain pricing mechanism in the monopoly market. Our result shows that the dual-licensing model is more profitable when the difference between flexibility benefit and support cost exceeds a certain threshold. Otherwise, the open source vendor prefers the support model. The optimal

choice between the dual-licensing model and the support model also depends on other factors such as marginal cost for support and proportion of the high-type and low-type customers. Our paper can give guidelines to the open source software vendors that may have difficulty in making pricing decision in the current environment.

We extend our model to a duopoly competition setting. The first scenario models competition between a commercial software vendor and an open source software vendor with a dual-licensing model. We analyze the optimal pricing strategies of each software vendor. We find the conditions under which an equilibrium exists. At equilibrium, the commercial and the open source software vendors split the market. The open source software vendor with a dual-licensing model serves high-type customers only, charging higher price than the commercial software vendor that covers the low-type segment. Our result proves the viability of the dual-licensing model of the open source software in the presence of competition with commercial software. Next, we study the duopoly software market when there is competition between a commercial software vendor and an open source software vendor with a support model. The profitability of the support model has been questioned since all the customers can get the software for free and the vendor can sell its support only to the customers in need, who are low-type customers in our model. We find that the commercial software vendor competing against the open source support model charges price equal to the open source vendor's marginal cost for support. With this pricing strategy of the commercial software vendor, the support model brings zero profit to the open source software vendor.

Then, we examine the case of quality asymmetry. Interestingly, we find that the open source support model is viable in the presence of quality asymmetry no matter whether the quality of the open source software is higher or lower than the commercial software quality. The case of the

open source software with higher quality is reasonable. The opposite case is interesting in the sense that the open source software with lower quality is profitable while the open source software with the same quality as the commercial software is not. This can be explained as market expansion for the commercial software. When both vendors offer the same quality software the high-type customers have no incentive to buy the commercial software since the open source software is free and they do not need any support. However, when the commercial software is superior to the open source software, some high-type customers may buy the commercial software, implying that the commercial software vendor can target such high-type customers as well.

We generalize our model to reflect what happens in the real-world database and operating systems markets. We model the different sources of the benefit from different open source software. For example, customers may enjoy customizability of open source application while they may benefit independence from an established commercial software platform by adopting open source operating systems. Motivated by real-world phenomenon, for example, Google, which is a tech savvy customer of Red Hat's support model, we assume that support cost may incur to both types of customers. We set up a two-period game to study the role of switching cost that customers face when they switch from established commercial software to open source software. We find that neither the dual-licensing nor the support model is viable when the switching cost is large. When the switching cost is lower than a certain threshold, software vendors split the market in the second period in a way that the commercial software serves low-type customers and the open source software vendor covers high-type segment. Interestingly, our results show that the open source software with support may dominate the market in the second

period when the vendor independence benefit exceeds switching cost. Our study identifies the conditions under which each open source pricing model is viable.

Finally, we discuss viability of open source support model under threat from large software vendor's entering the support service market. Inspired by the Linux support service market where Oracle recently started selling support service for Linux distributed by Red Hat, we model the open source support service market in order to investigate whether an established software vendor's move hurts an open source software service provider by sharing the same customer base or helps it by expanding the market and making the open source software an industry standard. We find that Red Hat's business model is hurt by Oracle move when the intensity of network externality is low. In this case, Red Hat's competitor, Novell can be better off or worse off with Oracle depending on conditions. When customers value the size of the network, our results show that Oracle move can allow Red Hat to make more profit in the oligopoly market with Oracle than it makes in the duopoly market without Oracle. In the presence of large network externality, Oracle move hurts Novell while it may help Red Hat. It is interesting in the sense that Oracle's targeting Red Hat makes Red Hat Linux an industry standard and hurts Novell's business when customers enjoy relatively high benefit from a large network.

The rest of the article is organized as follows. We discuss the related literature in Section 4.2. We present our model in Section 4.3 and examine each of the three pricing mechanisms under monopoly in Section 4.4. In Section 4.5, we extend our model to a duopoly competition between a commercial software vendor and an open source software vendor with each of the open source pricing models. We generalize our model and study the role of switching cost in Section 4.6. In Section 4.7 we discuss the viability of open source support model under threat from an established software vendor. Section 4.8 concludes the paper.

4.2 Literature Review

This paper is relevant to two streams of research: (1) information goods pricing, and (2) the economics of the open source software. The issue of information goods pricing been extensively studied in the domain of information systems. Dewan and Mendelson (1990) examine the optimal pricing policy and capacity investment strategy in the context of ASP in the presence of nonlinear delay costs. Mendelson and Whang (1990) study the issue of pricing computer services under asymmetric information for queues. While the expected mean delay is identified as a critical factor that affect consumers' utility (Dewan and Mendelson 1990; Mendelson and Whang 1990), Mackie-Mason and Varian (1995) argue that what consumers cares about is whether their packets are eventually transmitted, not the delay. Westland (1992) examines the optimal pricing of information systems in the presence of network externalities. Gupta et al. (1997) develops a model which integrates priority-based pricing and congestion-based pricing. Pricing strategies for information goods or IT-enabled pricing strategies have been widely examined by researchers in the domain of Information Systems. Such issues include versioning (Bhargava and Choudhary 2001; Sundararajan 2004), bundling (Bakos and Brynjolfsson 1999; Chuang and Sirbu 1999; Hitt and Chen 2005), price discrimination (Choudhary et al. 2005; Dewan et al. 2003), and price dispersion (Clemons et al. 2002). Chen and Png (2003) examine information goods pricing in the context of digital rights management. This paper aims to make contributions to the literature of information goods pricing by first examining pricing issue of open source software, which has not been studied yet.

Prior literature on open source software focuses heavily on understanding the motivations of the individual developers to participate in and contribute to open source software projects (Fitzgerald and Feller 2001; Shah 2006). von Hippel and von Krogh (2003) consider two models

of innovation, private and collective, and argue that contributors to OSS projects get intangible private benefits which are not present for free riders, such as personal satisfaction and learning. Franke and von Hippel (2003) examine the motivations of the Apache project participants and find that users' desire to satisfy their own needs gives incentive for participation. Raymond (1999) suggests that the reputation and the status motivate developers' participation while Ghosh (1998) argues that enjoyment and creativity matter. More recently, empirical studies with survey data find that there are a variety of reasons for OSS participation (Ghosh et al. 2002, Hertel et al. 2003). Rossi (2004) classifies various motivations into either intrinsic or extrinsic. Others examine the issue of open source participation through an economic lens and suggest that existing economic theory can partially explain OSS participation (Learner and Tirole 2001, 2002). Hann et al. (2002) argue that OSS participation is driven by career concerns, learning, and reputation. Roberts et al. (2006) develop a theoretical model to examine the system of interrelationship between motivations, participation, and performance. We view the motivation for open source software development in a different way from the existing literature. That is, some open source software vendors may share the same incentive with commercial software vendors who pursue financial gain. Our paper examines the incentive of open source software vendors from an economic perspective and identifies the conditions under which they can make money from open source software.

Modeling competition between proprietary software and open source software is an emerging issue among researchers who study open source software from an economic perspective. Raghunathan et al. (2005) examine the quality debate in open source software by setting up an analytical model and show that open source software quality is not necessarily lower than commercial software quality. Casadesus-Masanell and Ghemawat (2006) analyze a

dynamic mixed duopoly in which a for-profit proprietary software vendor interacts with an open source software vendor that prices at zero in the presence of demand-side learning effects. Economides and Katsamakas (2006) analyze the optimal two-sided pricing strategy of a platform firm and compare industry structures based on a proprietary platform such as Microsoft Windows with those based on an open source platform such as Linux. From a legal perspective, some research discusses the issue of the for-profit open source software, with which the vendor makes profit (Gomulkiewicz 2004, Välimäki 2003). The existing literature assumes that open source software is free of charge and does not consider the commercial open source software from which the vendors make profit yet.

Our paper contributes to the literature in the following ways. First, this paper is the first study that examines the issue of pricing for-profit open source software from an economic perspective. To our best knowledge, no research has examined the issue of open source software pricing through an economic lens despite the growing interest in the commercial open source software among industry experts and jurists. We find the conditions under which each model is viable by identifying the factors that affect the viability of the pricing models. Second, our result can give pricing guidelines to the open source software vendors, which is not clear in the current state. Finally, we model the motivation for and the barrier to open source adoption, reflecting reality. Our findings may provide a better picture of the open source software market.

4.3 Model

We analyze the optimal pricing decisions of an open source software vendor and investigate whether the emerging pricing schemes for open source software are viable in both monopoly and duopoly cases where the open source software vendor competes against a commercial software

vendor. We consider two models of open source software pricing: the dual-licensing model and the support model. The dual-licensing model provides the same open source software under two different licenses: open source license and commercial license. Users who want to donate their source code to the Open Source community can license their software under the General Public License (GPL), which is the most popular free software license. Under the open source license, the licensees can freely modify, distribute, and use the software at no charge. On the other hand, any users who want to use the open source software for profit seeking purpose must purchase a commercial license. MySQL and Openoffice are following the dual-licensing model. Under the support model, users can get the software for free, but the software vendor charges the users for support. Red Hat and JBoss are examples of successful open source support model.

4.3.1 Customers

Customers in our model are firms, not individual software users. Potential customers divide into two segments. A proportion μ of customers are technically savvy, and we call them high-type customers. These customers have a capable in-house IT management team. We assume that the high-type customers can take advantage of the flexibility of the open source software by customizing the source code and that their in-house IT management team can handle maintenance so that they do not need any further support from a vendor or a third party. The high-type customers may adopt the open source software not only because open source software has cost advantage but also it gives flexibility. Customers in the remaining fraction $(1 - \mu)$ are low-type customers that may incur support cost if they adopt the open source software. The low-type customers' incentive for open source adoption may be cost savings only. No matter which segment a customer is in, the customer does not enjoy any flexibility benefit or suffer from

support cost when it uses commercial software. Within each segment, potential customers are characterized by their value, *v*, from using the software. We assume that *v* is uniformly distributed on [0, 1], which leads to a linear demand curve. We also assume risk neutrality of the customers. We consider two sources of customer utility from open source software.

Our model assumes correlation between the value of open source software value customer's technology savviness in a way that any additional benefit that technology savvy customers enjoy due to openness of source code is captured by flexibility benefit. In other words, technology savvy customers' capability of managing and modifying source code adds value to the software, which is flexibility benefit in our model. The correlation between software value and technology savviness will lead to marketability of open source software under competition against dominant commercial software since the open source software can be appealing to some customers who have an in-house IT management team with skillful workers. This view is realistic since most business customers of commercialized open source software are technology savvy firms. For example, Google, Yahoo, and NASA are the customers of MySQL, which is open source database.

4.3.2 Software Pricing Schemes

We examine a commercial software model and two open source pricing schemes: dual-licensing model and support model. In this section, we present our model and derive customer utility and vendor profit under each of the three software pricing mechanisms.

Commercial Software

When a customer buys the commercial software, her benefit would be the same no matter which type the customer is. In other words, commercial software incurs neither flexibility benefit nor

support cost to the customer beyond the value of v that the customer derives from using the software. In reality, the commercial software may incur some level of flexibility benefit and/or support cost, which is not significant compared to the case of the open source software. In the open source pricing models, we consider the relative benefit and cost from the open source software. C in the subscript means commercial software. Let p_C be the price of the commercial software. The customer's utility would be

$$u_C = v - p_C.$$

Let q_C be the demand for the commercial software. Then the profit for the commercial software vendor would be $\pi_C = q_C p_C$.

Open Source Software: Dual-Licensing Model

We label the high-type customers H and the low-type customers L. The proportion of the high-type customers is μ and the remaining proportion $(1 - \mu)$ is the low-type segment. D in the subscript represents the dual-licensing model of open source software. When adopting open source software, a high-type customer enjoys the flexibility benefit whereas a low-type customer suffers from the support cost. Let f be the flexibility benefit and s be the support cost. p_D denotes the price of the open source software with a commercial license. The net benefits for a high-type customer and a low-type customer are as follows:

$$u_{DH} = v - p_D + f$$
$$u_{DL} = v - p_D - s.$$

Let q_{DH} and q_{DL} be the demand for the open source software under dual-licensing pricing scheme from the high-type and the low-type segments, respectively. The profit for the open source software vendor adopting the dual-licensing model would be

$$\pi_D = \{\mu q_{DH} + (1-\mu) q_{DL}\} p_D.$$

Open Source Software: Support Model

Let *S* in the subscript represent the support model for open source software. Under this pricing scheme, the open source software vendor provides the software at no charge. The vendor makes profit by selling its support service to the users in need. Let p_S be the price for the support. The net benefits for a high-type and a low-type customer are

$$u_{SH} = v + f$$
$$u_{SL} = v - p_S \quad \text{given} \quad p_S \le s.$$

Note that p_S is bounded by the support cost *s*. If p_S exceeds *s*, no customer would buy support service. Since the high-type customers use the software for free and would never ask for any support, the vendor makes profit from the low-type customers only. Let q_S be the demand for the support. We assume that support and service incur marginal cost, c_S. The profit for the open source software vendor adopting the support model would be $\pi_S = (1-\mu)\, q_S\, (p_S - c_S)$. Table 1 summarizes utility that a customer enjoys from software under each of the three mechanisms.

Table 1. Utility from Software

	High-Type	Low-Type
Commercial	$v - p_C$	$v - p_C$
OSS: Dual-Licensing	$v - p_D + f$	$v - p_D - s$
OSS: Support	$v + f$	$v - p_S$

4.4 Monopoly

In this section, we examine the optimal pricing strategies for each pricing scheme under monopoly. We compare prices and profits of the three pricing models: commercial, open source dual-licensing and open source support. We investigate whether each mechanism is viable when the software vendor monopolizes the market. Since a monopoly case is not uninteresting for software (e.g. Microsoft), we start with the monopoly case although it may not be realistic for most for-profit open source software currently available in the market.

4.4.1 Price

Commercial Software

Customers who get positive utility would buy the software. The demand for the commercial software would be

$$q_C = \int_{p_C}^{1} dv = 1 - p_C .$$

The profit for the commercial software vendor is

$$\pi_C = q_C p = (1 - p_C) p_C .$$

The first-order condition for π_C is

$$\frac{\partial \pi_C}{\partial p_C} = 1 - 2p_C = 0,$$

resulting in the profit-maximizing price as

$$p_C^* = \frac{1}{2}.$$

Open Source Software: Dual-Licensing Model

The demand for the open source software from the high-type and the low-type segments are

$$q_{DH} = \int_{p_D - f}^{1} dv = 1 - p_D + f$$
$$q_{DL} = \int_{p_D + s}^{1} dv = 1 - p_D - s.$$

The open source software vendor adopting a dual-licensing gets the following profit:

$$\begin{aligned}\pi_D &= \{\mu q_{DH} + (1-\mu) q_{DL}\} p_D \\ &= \{\mu (1 - p_D + f) + (1-\mu)(1 - p_D - s)\} p_D.\end{aligned}$$

The first-order condition for π_D is

$$\frac{\partial \pi_D}{\partial p_D} = \mu f - (1-\mu)s + 1 - 2p_D = 0.$$

Thus, the optimal price under the dual-licensing model is

$$p_D^* = \frac{\mu f - (1-\mu)s + 1}{2}.$$

Open Source Software: Support Model

Under the support model, the open source software vendor makes profit out of low-type customers only. The demand for the support model would be

$$q_S = (1-\mu)\int_{p_S}^{1} dv = (1-\mu)(1-p_S).$$

The profit for the open source software is then

$$\pi_S = q_S (p_S - c_S) = (1-\mu)(1-p_S)(p_S - c_S)$$

where c_S is the marginal cost for support. The first-order condition is

$$\frac{\partial \pi_S}{\partial p_S} = (1-\mu)(1 - 2p_S + c_S) = 0.$$

Thus, the optimal price is

$$p_S^* = \frac{1 + c_S}{2}.$$

4.4.2 Profit

In this section, we compare the profits of the three different software pricing models and investigate how much profit the open source software pricing scheme brings compared to commercial software. We examine the viability of the open source software pricing models in a monopolized software market. Note that the commercial software vendor makes the following profit at the optimal price:

$$\pi_C^* = (1 - p_C^*) p_C^* = \frac{1}{4}.$$

The profit for the open source software vendor with a dual licensing model at the optimal price would be

$$\begin{aligned} \pi_D^* &= \{\mu(1 - p_D^* + f) + (1-\mu)(1 - p_D^* - s)\} p_D^* \\ &= \{\mu f - (1-\mu)s + 1 - p_D^*\} p_D^* \\ &= \frac{1}{4}\{\mu f - (1-\mu)s + 1\}^2. \end{aligned}$$

Finally, with a support model, the open source software vendor makes its profit as follows:

$$\begin{aligned} \pi_S^* &= (1-\mu)(1 - p_S^*)(p_S^* - c_S) \\ &= (1-\mu)\left(1 - \frac{1+c_S}{2}\right)\left(\frac{1+c_S}{2} - c_S\right) \\ &= \frac{1}{4}(1-\mu)(1-c_S)^2. \end{aligned}$$

Comparison of the optimal prices leads to the following Proposition.

Proposition 1: *Under monopoly, the open source software with a dual-licensing model charges higher price than the commercial software vendor* $(p_D^* > p_C^*)$ *and makes higher profit* $(\pi_D^* > \pi_C^*)$ *if the total flexibility benefit for the high-type customers is higher than the total support cost for the low-type customers* $(\mu f > (1-\mu)s)$. *Otherwise if* $(\mu f < (1-\mu)s)$ *, the*

commercial software is more expensive $(p_D^* < p_C^*)$ *and more profitable* $(\pi_D^* < \pi_C^*)$ *than the open source software with a dual-licensing model. With a support model, the open source software vendor provides support at higher price than commercial software price* $(p_S^* > p_C^*)$ *and makes less profit* $(\pi_S^* < \pi_C^*)$.

Proof. See the Appendix.

When an open source software vendor with a dual-licensing model monopolizes the market, the open source software price depends on flexibility benefit, support cost and proportion of high-type customers. If the total flexibility benefit for the high-type customers exceeds the total support cost for the low-type customers, open source software can be more expensive than commercial software. Otherwise, open source software charges a price lower than commercial software. Interestingly, with a support model, an open source vendor sells its support at higher price than the commercial software price. Since the profitable customers for the open source support model is only the low-type segment, the open source vendor's optimal strategy is charging high price for support. This may contradict the results from the survey on motivation for open source adoption, which cites cost advantage on the top of the list. In reality, monopoly may not be the case for most for-profit open source software. Proposition 1 shows the optimal pricing strategies of the open source software as a benchmark. We also find that both pricing models for open source software are viable when the software vendor monopolizes the market. Then which pricing scheme is better for the open source software vendor that monopolizes the market? Dual-licensing model or support model? We next examine the strategic incentive of the open source software vendor for each pricing scheme and find the conditions under which one mechanism is better than the other.

Proposition 2: *Under monopoly, a dual-licensing model is more profitable than a support*

model $(\pi_D^* > \pi_S^*)$ *when* $\mu f - (1-\mu)s > \sqrt{1-\mu}(1-c_S) - 1$. *The open source software vendor makes more profit with a support model than with a dual licensing model* $(\pi_D^* < \pi_S^*)$ *when* $\mu f - (1-\mu)s < \sqrt{1-\mu}(1-c_S) - 1$.

Proof. See the Appendix.

Proposition 2 shows the conditions under which the monopolist open source vendor makes a choice between the dual-licensing model and the support model. The result implies that the dual-licensing model is better than the support model if the total net benefit from open source, which is the difference between the total flexibility benefit and total support cost, is greater than a certain threshold. If the net benefit does not reach the threshold, the support model is more profitable than the dual-licensing model.

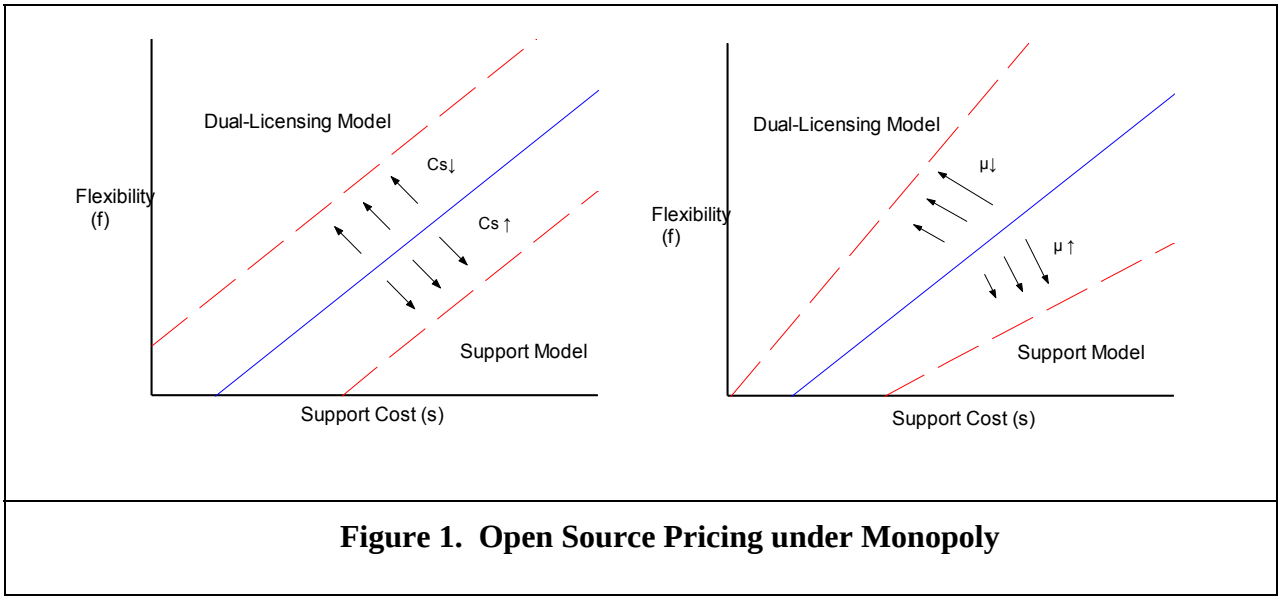


Figure 1. Open Source Pricing under Monopoly

Figure 1 illustrates the role of the marginal cost for support and the proportion of the high-type customers to the framework. It is shown that the dual-licensing model is preferred in the upper-left region while the support model is chosen in the lower-right region. As the marginal cost for support increases, the region for the dual-licensing model becomes wider. This is

reasonable in the sense that the support model is less viable when the marginal cost for support is higher. The proportion of the high-type customers tilts the line while the marginal cost shifts it. As the high-type segment becomes bigger, the dual-licensing model becomes more preferable. This makes sense because the high-type customers are profitable only to the vendor with the dual-licensing model. Thus, an increase in the proportion of the high-type customers makes the dual-licensing model more attractive to the open source software vendor.

4.5 Competition

In the real world, there are successful examples of for-profit open source software competing with commercial software. With a dual-licensing model, MySQL competes against Microsoft SQL server. Red Hat Enterprise Linux, which is a commercial version with support and service, competes against Microsoft Windows. In this section, we study the optimal pricing strategies of the commercial and the open source software vendors in more realistic cases. We extend our model to a duopoly competition setting. First, we examine the competition between a commercial software vendor and an open source software vendor with a dual-licensing model. Then we study the case where an open source software vendor with a support model competes against a commercial software vendor.

4.5.1 Open Source Software with Dual-Licensing versus Commercial Software

Recall that a customer enjoys the following utility by adopting commercial software:

$$u_C = v - p_C.$$

The open source software vendor with a dual-licensing model provides the following benefits to a high-type customer and a low-type customer, respectively.

$$u_{DH} = v - p_D + f$$
$$u_{DL} = v - p_D - s.$$

A further analysis of the customer benefits leads to Proposition 3.

Proposition 3: *In a duopoly market with a commercial software vendor and an open source software vendor with a dual-licensing model, the commercial software vendor serves low-type customers while the open source software vendor serves high-type customers. An equilibrium exists in the presence of sufficient flexibility benefit and support cost* ($f \geq \sqrt{\mu}$ *and* $s \geq \frac{1}{2}$). *When flexibility benefit is high and support cost is low* ($f \geq \sqrt{\mu}$ *and* $0 < s < \frac{1}{2}$), *the condition for an equilibrium to exist is as follows:*

$$\mu(1+f)^2 \geq (2\mu f + 2\mu s + 1)(1-2s).$$

The optimal prices are

$$p_C^* = \frac{1}{2} \quad \textit{and} \quad p_D^* = \frac{1+f}{2}.$$

Proof. See the Appendix.

We find that a commercial software vendor and an open source software vendor with a dual-licensing model target different segments of the customers. When they compete against each other, a commercial software vendor serves the low-type customers while an open source software vendor serves the high-type customers. This is consistent with the literature on product discrimination and versioning in the sense that the high-type customers choose high-value products, open source software in this case, since it provides flexibility benefit in addition to the original value of the product to the high-type customers. Our findings support viability of the dual-licensing model for open source software.

4.5.2 Open Source Software with Support versus Commercial Software

We examine a duopoly software market where an open source software vendor with a support model competes with a commercial software vendor. We investigate whether the support model is viable in this market and examine the impact of competition on each vendor's profit. Recall that net benefit from commercial software would be

$$u_C = v - p_C.$$

On the other hand, open source software with a support model gives different levels of benefit to the customers in different types:

$$u_{SH} = v + f$$
$$u_{SL} = v - p_S \quad \text{given} \quad p_S \le s.$$

Proposition 4: *In a duopoly market with a commercial software vendor and an open source software vendor with a support model, the commercial software vendor charges price at the open source vendor's marginal cost for support* ($p_C^* = c_S$). *The commercial software vendor makes a positive profit* ($\pi_C = (1-\mu)(1-c_S)c_S$) *while the open source software vendor makes zero profit. When the marginal cost for support is large* $\left(\frac{1}{2} < c_S < 1\right)$, *the commercial software vendor charges its monopoly price.*

Proof. See the Appendix.

Unlike the dual-licensing model, the open source support model is not viable in a duopoly market. The commercial software vendor's best strategy is not to allow the open source vendor to make any positive profit by aggressive pricing. Since there is an asymmetry in marginal cost, the open source software vendor is not able to survive the competition. Our result shows that the commercial software vendor provides its software at the open source software vendor's marginal

cost for support. Consequently, the open source vendor makes zero profit while the commercial software vendor is able to make positive profit.

Then we examine the different quality case and study how asymmetry in quality affects the viability of the open source software support model when both vendors decide their prices simultaneously. Let α be the quality difference parameter, which is a proportion of the value of the commercial software to the value of the open source software. We assume that quality is the only factor that affects the value of software. We use software value and software quality interchangeably. Then the customer's utility from the commercial software would be

$$u_C = \alpha v - p_C .$$

Recall that the net benefit from the open source software with a support model is

$$u_{SH} = v + f$$
$$u_{SL} = v - p_S \quad \text{given} \quad p_S \le s.$$

There are two possible cases of asymmetric quality. The commercial software offers higher quality than the open source software ($\alpha > 1$) and the open source software quality is higher than the commercial software ($0 < \alpha < 1$). We start with the former case.

Case 1: Better Commercial Software ($\alpha > 1$)

<u>High-Type Customers</u>

Note that $u_C - u_{SH} = \alpha v - p_C - v - f$. The customer in the high-type segment would buy the commercial software $u_C - u_{SH} > 0$, i.e. $(\alpha - 1)v > p_C + f$. Since $\alpha > 1$,

$$\frac{p_C + f}{\alpha - 1} < v < 1 \quad \Rightarrow \quad \text{Buy commercial software}$$
$$0 < v < \frac{p_C + f}{\alpha - 1} \quad \Rightarrow \quad \text{Get open source software for free}$$

Thus, the demand for the commercial and the open source software would be

$$q_{CH} = \int_{\frac{p_C+f}{\alpha-1}}^{1} dv = 1 - \frac{p_C + f}{\alpha - 1} \quad \text{and} \quad q_{SH} = \int_{0}^{\frac{p_C+f}{\alpha-1}} dv = \frac{p_C + f}{\alpha - 1}.$$

Low-Type Customers

The customer in the low-type segment would buy the commercial software $u_C - u_{SL} > 0,$ i.e.

$(\alpha - 1)v > p_C - p_S$.

$$\begin{aligned}
&\frac{p_C - p_S}{\alpha - 1} < v < 1 && \Rightarrow \quad \text{Buy commercial software} \\
&p_S < v < \frac{p_C - p_S}{\alpha - 1} && \Rightarrow \quad \text{Get open source software and buy support} \\
&0 < v < p_S && \Rightarrow \quad \text{Not buy any software}
\end{aligned}$$

Thus, the demand for the commercial and the open source software would be

$$q_{CL} = \int_{\frac{p_C-p_S}{\alpha-1}}^{1} dv = 1 - \frac{p_C - p_S}{\alpha - 1} \quad \text{and} \quad q_{SL} = \int_{p_S}^{\frac{p_C-p_S}{\alpha-1}} dv = \frac{p_C - p_S}{\alpha - 1} - p_S.$$

The profit for the commercial vendor is

$$\begin{aligned}
\pi_C &= \{\mu q_{CH} + (1-\mu) q_{CL}\} p_C \\
&= \left\{ \mu \left(1 - \frac{p_C + f}{\alpha - 1} \right) + (1-\mu) \left(1 - \frac{p_C - p_S}{\alpha - 1} \right) \right\} p_C \\
&= \frac{1}{\alpha - 1} \{(1-\mu) p_S - p_C + \alpha - \mu f - 1\} p_C .
\end{aligned}$$

The first-order condition is

$$\frac{\partial \pi_C}{\partial p_C} = \frac{1}{\alpha - 1} \{(1-\mu) p_S - 2 p_C + \alpha - \mu f - 1\} = 0.$$

The profit for the open source software vendor comes from only low-type customers. Thus,

$$\begin{aligned}\pi_S &= (1-\mu)q_{SL}(p_S - c_S) \\ &= (1-\mu)\left(\frac{p_C - p_S}{\alpha - 1} - p_S\right)(p_S - c_S) \\ &= \frac{1}{\alpha - 1}(1-\mu)(p_C - \alpha p_S)(p_S - c_S).\end{aligned}$$

The first-order condition is

$$\frac{\partial \pi_C}{\partial p_C} = \frac{1}{\alpha - 1}(1-\mu)(p_C - 2\alpha p_S + \alpha c_S) = 0.$$

Solving for the following equations give optimal prices.

$$\begin{aligned}(1-\mu)p_S - 2p_C + \alpha - \mu f - 1 &= 0 \\ p_C - 2\alpha p_S + \alpha c_S &= 0.\end{aligned}$$

The optimal prices are

$$p_C^* = \frac{\alpha\{2\alpha + (1-\mu)c_S - 2(1+\mu f)\}}{4\alpha + \mu - 1} \quad \text{and} \quad p_S^* = \frac{\alpha(1+2c_S) - (1+\mu f)}{4\alpha + \mu - 1}.$$

Case 2: Better Open Source Software ($0 < \alpha < 1$**)**

<u>High-Type Customers</u>

Note that $u_C - u_{SH} = \alpha v - p_C - v - f = -(1-\alpha)v - p_c - f < 0.$ Thus, high-type customers always choose the open source software. When the open source software provides higher quality than the commercial software, both software vendors target the low-type customers only.

<u>Low-Type Customers</u>

The customer in a low-type segment would buy open source software $u_{SL} - u_C > 0,$ i.e.

$(1-\alpha)v > p_S - p_C.$

$$\frac{p_S - p_C}{1-\alpha} < v < 1 \quad \Rightarrow \quad \text{Get open source software and buy support}$$

$$p_C < v < \frac{p_S - p_C}{1-\alpha} \quad \Rightarrow \quad \text{Buy commercial software}$$

$$0 < v < p_C \quad \Rightarrow \quad \text{Not buy any software}$$

Thus, the demand for the commercial and the open source software would be

$$q_{CL} = \int_{p_C}^{\frac{p_S - p_C}{1-\alpha}} dv = \frac{p_S - p_C}{1-\alpha} - p_C = \frac{p_S - p_C(2-\alpha)}{1-\alpha}$$

$$q_{SL} = \int_{\frac{p_S - p_C}{1-\alpha}}^{1} dv = 1 - \frac{p_S - p_C}{1-\alpha}.$$

The profit for the commercial software vendor is

$$\pi_C = (1-\mu) q_{CL} p_C = \frac{1-\mu}{1-\alpha}\{p_S - p_C(2-\alpha)\} p_C.$$

The first-order condition is

$$\frac{\partial \pi_C}{\partial p_C} = \frac{1-\mu}{1-\alpha}\{p_S - 2p_C(2-\alpha)\} = 0.$$

The profit for the open source software vendor is

$$\pi_S = (1-\mu) q_{SL}(p_S - c_S) = \frac{1-\mu}{1-\alpha}(1-\alpha - p_S + p_C)(p_S - c_S).$$

The first-order condition is

$$\frac{\partial \pi_S}{\partial p_S} = \frac{1-\mu}{1-\alpha}(1-\alpha - 2p_S + p_C + c_S) = 0.$$

Solving for the following equations give optimal prices.

$$p_S - 2p_C(2-\alpha) = 0$$
$$1-\alpha - 2p_S + p_C + c_S = 0.$$

The optimal prices are

$$p_C^* = \frac{c_S + 1 - \alpha}{7 - 4\alpha} \quad \text{and} \quad p_S^* = \frac{2(c_S + 1 - \alpha)(2-\alpha)}{7 - 4\alpha}.$$

A further analysis of the prices leads to the Proposition 5.

Proposition 5:

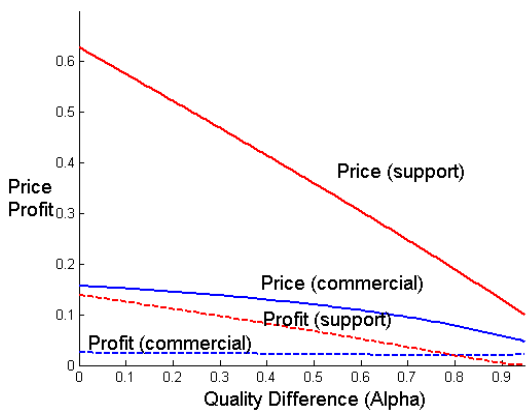
(1) In a software market where the commercial software vendor provides higher quality than the open source software with a support model $(\alpha > 1)$*, the commercial software vendor charges more for the software* $(p_C \uparrow)$ *and the open source software vendor charges more for the support* $(p_S \uparrow)$ *as the quality difference between the commercial software and the open source software increases* $(\alpha \uparrow)$*. The price difference* $(p_C - p_S)$ *increases with the quality difference.*

(2) When the open source software quality is higher than the commercial software $(0 < \alpha < 1)$*, the open source software vendor charges more on support* $(p_S \uparrow)$ *as the quality difference increases* $(\alpha \downarrow)$*. When the marginal cost for support is low* $(0 < c_S < \frac{3}{4})$*, commercial software price increases* $(p_C \uparrow)$ *with quality difference* $(\alpha \downarrow)$*. Otherwise* $(\frac{3}{4} < c_S < 1)$*, commercial software price decreases* $(p_C \downarrow)$ *with quality difference* $(\alpha \downarrow)$*.*

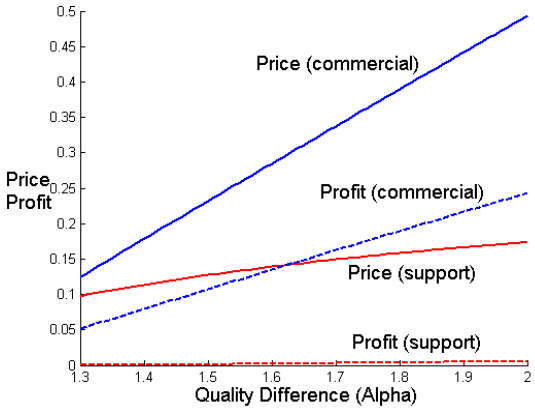
Proof. See the Appendix.

Proposition 5 shows that the open source support model is viable in the presence of quality asymmetry. No matter whether the quality of the open source software is higher or lower than the commercial software, the open source software vendor makes profit with a support model in a duopoly market. When the open source software has higher quality, it is reasonable that the open source software is profitable. The opposite case is interesting in the sense that the open source software with lower quality is also profitable although we find that the open source software with same quality as the commercial software is not. This can be explained as the

market expansion of the commercial software vendor. When the quality is identical for both vendors, the high-type customers have no incentive to buy the commercial software, since the open source software is free and they do not need any support. However, when the commercial software offers higher quality than the open source software, some high-type customers will be willing to buy the commercial software, therefore, the market for the commercial software is expanded.



(a) Better Open Source Software ($\alpha < 1$)



(b) Better Commercial Software ($\alpha > 1$)

Figure 2. Software Price and Profit

Figure 2 illustrates the optimal prices and profits with respect to the quality difference. Unlike the case of symmetric quality, the open source software vendor with a support model can make positive profit in the presence of quality asymmetry. Even when the open source software provides worse quality, the open source software vendor makes a thin but positive profit. As mentioned earlier, this can be a result of market expansion. This quality difference can be one possible way to explain the successful open source support model by some open source software vendors such as Red Hat and JBoss. Our findings support the viability of the open source software support model.

4.6 Competition in the Presence of Switching Cost

Commercializing open source software is a fairly new idea which is not prevalent in a real-world software market yet. Unless an open source software vendor creates a market by introducing totally new software, it cannot help facing competition against an existing commercial software vendor which already dominates the market. In that case, the potential customers who consider buying commercialized open source software need to consider the extra cost that incurs when they switch from the commercial software that they are familiar with to the open source software, as well as the price they pay for the software. Economists call such extra cost switching cost. In order to reflect reality, taking switching cost into account is important. This is first issue we bring in this section to enrich our model and examine what happens in reality. Second, we consider two different cases which are representative of the real-world open source software market. We choose two scenarios so that they can fully cover the open source software market in terms of both pricing models and software characteristics. As discussed earlier, two pricing mechanisms to sell open source software have been available: dual-licensing model and support model. MySQL is a successful example of commercializing open source software under dual-licensing mechanism and Red Hat Linux is a case of support model. Software characteristics wise, we cover two different categories. One is application software, and the other is operating systems software. The source of benefit is different in that customers are likely to enjoy modifiability of open source application software whereas open source operating systems software like Linux allow flexibility and vendor independence benefit.

In this section, we examine two scenarios which can serve as representative cases of the real world open source software markets. The first case is inspired by the competition in the database management systems software market where open source MySQL competes with Microsoft SQL

Server. This scenario reflects a market with open source application software under dual-licensing model. The second case models operating systems software market where Red Had Linux is threatening Microsoft Windows, which captures a market with open source operating systems software under support model. In both scenarios, we examine the role of switching cost in the competition between open source software and commercial software. We aim to have enough insights from these two scenarios since they can represent the real-world software markets with competition between profit-seeking open source software vendor and dominating commercial software vendor such as Microsoft. By considering the factors that affect viability of commercial open source software such as switching cost and sources of benefits to open source customers, our analysis can provide managerial insights to open source software vendors who may want to pursue profit, commercial software vendors who may face a threat from open source software, and the customers who want to make a right decision between open source and commercial software.

In this section, we generalize our model and examine the case of two-period competition in the presence of switching cost. In Section 6.1, we model the competition between a commercial software vendor and an open source software vendor with a dual licensing model in the presence of switching cost (e.g. competition between Microsoft SQL server and MySQL). In Section 6.2, motivated by the competition between Microsoft's Windows and Red Hat Linux, we study the competition between commercial software and open source software with a support model. Our generalized model reflects reality better than the basic model in the following ways. First, we identify and model the different sources of OSS benefit. Customers enjoy different benefit from different open source software. For example, MySQL may provide customizability while Linux may allow platform independence. Second, we examine the role of switching cost by setting up a

two-period game. In reality, a for-profit open source software vendor often enters the market that has been dominated by a monopolistic commercial software vendor. In an electronic market such as a software market, vendors often play a strategy to lock in their customers by creating switching cost (Chen and Hitt 2002). Third, in contrast to the basic model where only low-type customers incur support costs, we assume that open source software may incur both flexibility benefit and support cost to both high-type and low-type customers to reflect the reality better. For example, Red Hat announced that Google operates its search engine and all of its computing functions on a cluster of more than 4,000 PCs running Red Hat and that Google is one of the large customers of the Red Hat Linux support model. This implies that technically savvy firms such as Google, may suffer from support cost as well.

4.6.1 Microsoft SQL Server versus MySQL

We model the competition between commercial and open source software with a dual licensing model. Figure 3 visualizes the structure of the game.

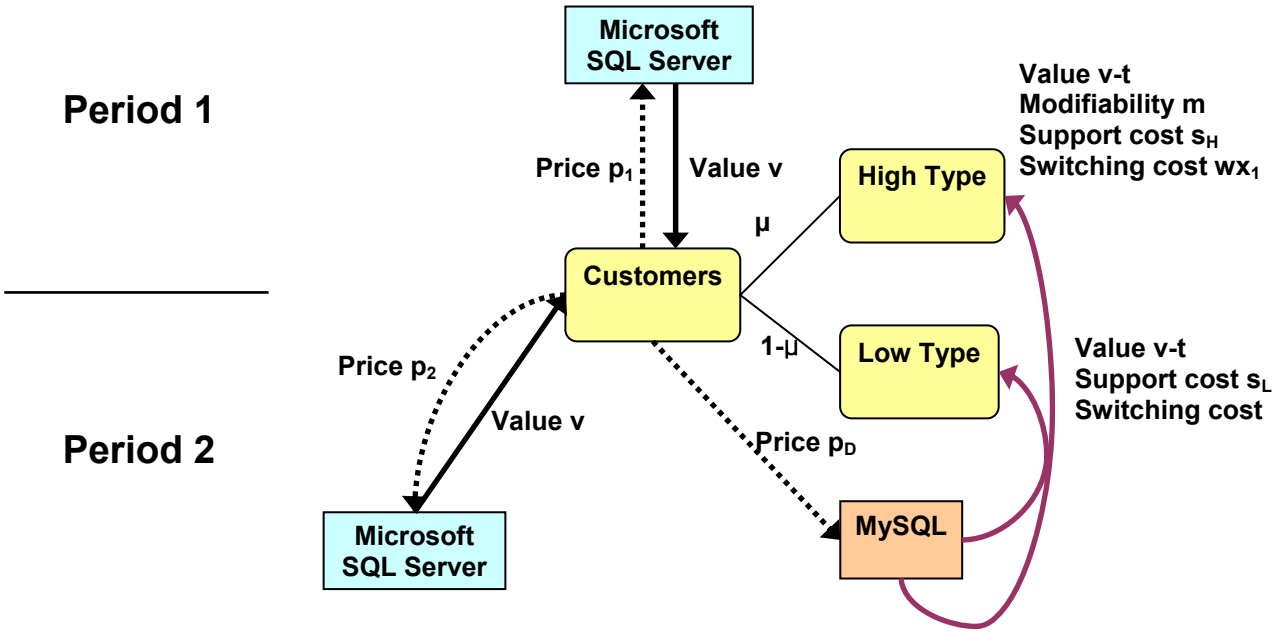


Figure 3. Microsoft SQL Server versus MySQL

Period 1: Commercial Software Only

- Commercial vendor introduces software at price p_1 for period 1 licensing.
- Customers decide whether or not to buy the software.
- The demand for commercial software (x_1) is determined.

Period 2: Commercial versus Open Source

- Period 2 game is based on the customers who purchased commercial software in period 1.
- An open source software vendor provides software at price p_D for period 2 licensing.
- A commercial software vendor observes the OSS price and then offers software at price p_2 for period 2 licensing.
- Customers choose either of the following strategies.
  - Stay with the commercial software and pay p_2.
  - Switch to open source software and incur switching cost wx_1.

    (Switching cost depends on the scale parameter w and the market size x_1.)
  - Exit, i.e., do not buy any software.

In this game, a customer has four strategic choices. A customer can buy commercial software in both period 1 and 2 (CC), or buy commercial software in period 1 and switch to open source software in period 2 (CO). It is also possible that the customer who buys the commercial software in period 1 chooses not to buy any in period 2 (CX). Finally, a customer can choose not to buy any software in any period (XX). Industry experts argue that Microsoft SQL server comes with more features than MySQL, such as compatibility with SAP3. Our model capture this functional deficiency of MySQL by having parameter *t.* MySQL allows high-type customers to enjoy modifiability *m*. But they suffer from support cost s_H. If a low-type customer buys open

source software it incurs support cost s_L. We assume that $0 < m - t - s_H < 1 + w$, meaning that net benefit to high type customer does not exceed sum of maximum possible product value and maximum possible switching cost. Note that p_1, p_2 and p_D represent commercial vendor's period 1 price, period 2 price, and open source software vendor's price, respectively. Then high-type and low-type customers enjoy the following utility from each choice.

	High - Type (μ)	Low - Type ($1 - \mu$)
CC	$v - p_1 + v - p_2$	$v - p_1 + v - p_2$
CO	$v - p_1 + v - p_D + m - t - s_H - wx_1$	$v - p_1 + v - p_D - s_L - wx_1$
CX	$v - p_1$	$v - p_1$
XX	0	0

Note that a customer who chooses CC enjoys positive utility: $v - p_1 \geq 0$ and $v - p_2 \geq 0$. For such customers, XX and CS are dominated by CC. Comparison of CC and CO is not trivial. A high-type customer prefers the commercial software to the open source software if

$$v - p_1 + v - p_2 > v - p_1 + v - p_D + m - t - s_H - wx_1.$$

Note that x_1 is the period 1 demand for the commercial software, which is, $1 - p_1$. Thus, the above condition can be written as $wp_1 + p_2 < w + p_D - m + t + s_H$. A low-type customer stays with the commercial software if

$$v - p_1 + v - p_2 > v - p_1 + v - p_D - s_L - wx_1$$

which can be rewritten as, $wp_1 + p_2 < w + p_D + s_L$. Denote $d_H = m - t - s_H$. A further analysis leads to Proposition 6.

Proposition 6: *In the presence of high switching cost ($w > d_H$), a commercial software vendor sets up its prices to prevent positive profit for an open source software vendor with a dual licensing model. The optimal first and second period prices for the commercial software are as follows:*

$$p_1^* = \frac{1}{3w^2}\left(3w^2 - 2wd_H - w + 1 - \sqrt{w^2({d_H}^2 + d_H + 4) - 2w(2d_H + 1) + 1}\right)$$
$$p_2^* = \frac{1}{3}\left(1 - d_H - \frac{1}{w} + \frac{1}{w}\sqrt{w^2({d_H}^2 + d_H + 4) - 2w(2d_H + 1) + 1}\right).$$

When the switching cost is low ($w < d_H$), a commercial software vendor serves low-type customers and an open source software vendor serves high-type customers by charging following prices:

$$p_1^* = \frac{1}{3w^2}\left(w^2 + 2w(d_H - s_L) - w + \frac{1}{1-\mu} - \sqrt{\left(w^2 + 2w(d_H - s_L) - w + \frac{1}{1-\mu}\right)^2 - 3w^2\left((d_H - s_L)^2 + 2w(d_H - s_L) - (d_H - s_L) - w + \frac{1}{(1-\mu)}\right)}\right)$$

$$p_2^* = d_H - s_L - \frac{1}{3w}\left(w^2 + 2w(d_H - s_L) - w + \frac{1}{1-\mu} - \sqrt{\left(w^2 + 2w(d_H - s_L) - w + \frac{1}{1-\mu}\right)^2 - 3w^2\left((d_H - s_L)^2 + 2w(d_H - s_L) - (d_H - s_L) - w + \frac{1}{(1-\mu)}\right)}\right)$$

$$p_D^* = d_H - w.$$

Proof. See the Appendix.

Proposition 6 shows that the dual-licensing model is not viable when commercial software is dominating the market and customers face high switching cost. The commercial software vendor sets up the prices not to allow the open source software vendor to make any positive profit. Consequently, the open source software vendor does not have any incentive to enter the market. In the presence of low switching cost, the open source software vendor provides its software at price that attracts high-type customers who are the users of commercial software in period 1. As a result, a commercial software vendor and an open source software vendor split the market in that a commercial software vendor serves low-type customers and an open source software

vendor serves high-type customers. Our results prove that the dual-licensing model is viable under competition against established commercial software if switching from commercial software to open source software does not incur large cost to the customers. Figure 4 and 5 illustrate the optimal prices and profits at equilibria.

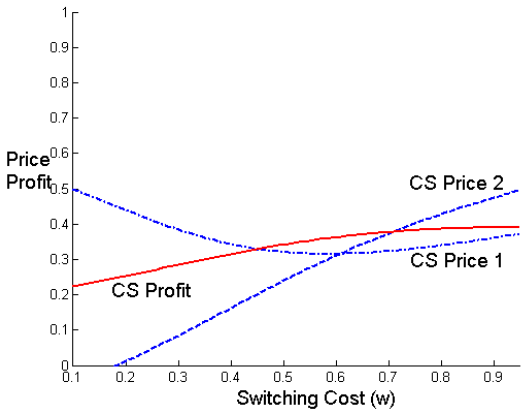


Figure 4. Price, Profit and Switching Cost (High Switching Cost)

Figure 4 shows the optimal first and second period prices and the profit for the commercial software. When the scale parameter for the switching cost is sufficiently large, the commercial software vendor charges low price in the first period and increases its price in the second so that it can have a large installed base in the first period and lock in the customers in the second. When the scale parameter is relatively small, the commercial vendor cannot increase switching cost in the second period efficiently by having a large installed base in the first period. Thus, the optimal strategy is to charge higher price in the first period than in the second to maximize its profit. We also find that the profit for Microsoft increases with the switching cost parameter.

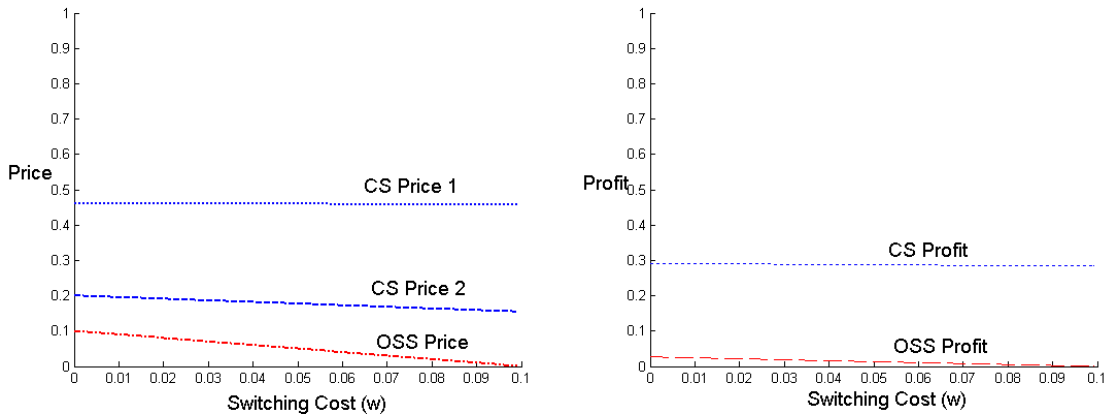


Figure 5. Price, Profit and Switching Cost (Low Switching Cost)

The optimal prices and profits in the presence of low switching cost are shown in Figure 5. Since customers face low switching cost when they make a decision in the second period, the open source software vendor sets up the price that attracts high-type customers. The commercial software vendor plays a pricing strategy to maximize its first period profit by charging higher price first and lowering it in the second. The commercial software vendor makes higher profit than the open source software vendor since it sells its software for both periods. But, unlike the case of high switching cost, the open source software vendor makes positive profit with a dual-licensing model.

4.6.2 Microsoft Windows versus Red Hat Linux

In this section, we examine the operating systems market where an open source software vendor competes against an established commercial software vendor. Red Hat Linux is one of the few open source software vendors that have been successful in commercializing its open source software by adopting a support model. Industry experts have been doubtful about its viability since customers get the software for free and the vendor only targets the customers who need

support. Red Hat proves viability of the support model. However, there is no academic research that identifies the conditions that affect viability and scalability of the open source support model. Figure 6 illustrates the model structure.

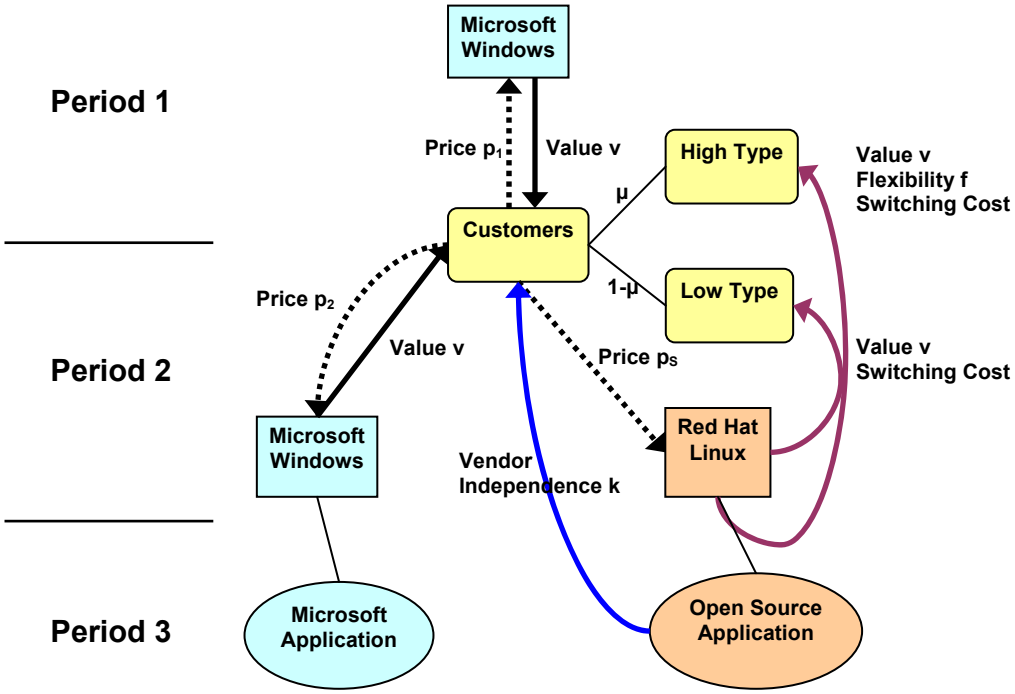


Figure 6. Microsoft Windows versus Red Hat Linux

The structure of the game is similar to the game in Section 4.6.1. The changes made are as follows. First, the sources of flexibility benefit are different in this game. Unlike customizability that may be the biggest part of the flexibility of application software such as database management systems, the open source operating systems allow users to enjoy vendor independence, which is identified as a major benefit of using Linux by surveys. For example, Microsoft Windows customers need to buy costly Microsoft-compatible applications such as Microsoft Office. But, the Linux customers may enjoy low-cost or free software such as StarOffice. In our model, Linux adopters realize its vendor independence benefit in period 3. Parameter *k* captures the benefit due to independence of commercial platform. It incurs to both

types of customers. Second, we generalize our setting in a way that open source software may incur support cost to high-type customers as well as low-type customers. Goolge, known as a firm with advanced technology is one of the biggest customers of Red Hat's support model. Unlike the open source application software which may allow technology savvy firms to modify the software, Linux may give such high-type firms flexibility which may exceed support cost. In this scenario, our model still captures support costs that Linux brings to the customers. By definition of high-type and low-type customers, high-type customers enjoy benefits more than low-type customers. Third, we normalize the support cost to low-type customers to be zero in our model. Thus, flexibility benefit (f) represents the additional benefit that high-type customers have compared to the low-type customers. Recall that p_1 , p_2 and p_S represent commercial vendor's period 1 price, period 2 price, and open source software vendor's price for support, respectively. Then high-type and low-type customers enjoy the following utility from each of possible choices.

	High - Type (μ)	Low - Type (1 - μ)
CC	$v - p_1 + v - p_2$	$v - p_1 + v - p_2$
CO	$v - p_1 + v + f - p_S - wx_1 + k$	$v - p_1 + v - p_S - wx_1 + k$
CX	$v - p_1$	$v - p_1$
XX	0	0

Solving for sub-game perfect equilibria leads to the following Proposition.

Proposition 7: *When the switching cost is high ($w > f + k - c_S$), a commercial software vendor plays a pricing strategy to prevent positive profit for an open source software vendor with a support model. The optimal first and second period prices for the commercial software are*

$$p_1^* = \frac{1}{3w^2}\left(3w^2 - 2w(f+k-c_S) - w + 1 - \sqrt{w^2\{(f+k-c_S)^2 + (f+k-c_S) + 4\} - 2w\{2(f+k-c_S)+1\} + 1}\right)$$

$$p_2^* = \frac{1}{3}\left(1 - (f+k-c_S) - \frac{1}{w} + \frac{1}{w}\sqrt{w^2\{(f+k-c_S)^2 + (f+k-c_S) + 4\} - 2w\{2(f+k-c_S)+1\} + 1}\right).$$

In the presence of medium switching cost ($k - c_S < w < f + k - c_S$), a commercial software vendor serves low-type customers and an open source software vendor serves high-type customers by charging following prices:

$$p_1^* = \frac{1}{3w^2}\left(w^2 - w(1-2f) + \frac{1}{1-\mu} - \sqrt{\left(w^2 - w(1-2f) + \frac{1}{1-\mu}\right)^2 - 3w^2\left(f^2 - f - w(1-2f) + \frac{1}{1-\mu}\right)}\right)$$

$$p_2^* = f - \frac{1}{3w}\left(w^2 - w(1-2f) + \frac{1}{1-\mu} - \sqrt{\left(w^2 - w(1-2f) + \frac{1}{1-\mu}\right)^2 - 3w^2\left(f^2 - f - w(1-2f) + \frac{1}{1-\mu}\right)}\right).$$

$$p_S^* = f + k - w$$

When the switching cost is low ($w < k - c_S$), both high-type and low-type customers switch from commercial software to open source software. The commercial software vendor sets up monopoly price in the first period ($p_1^* = \frac{1}{2}$), and the open source software vendor prices at $p_S^* = k - w$.

In the presence of high switching cost, the results are consistent with the case of the dual-licensing model. The open source software vendor cannot make any positive profit with a support model either. When the customers face medium switching cost, a commercial software vendor and an open source software vendor split the market. The commercial software vendor serves low-type customers and the open source software vendor serves high-type customers. When the switching cost is low, interestingly, open source software dominates the market in the second period. All the customers switch from commercial software to open source software. The commercial software vendor maximizes its first period profit by setting up monopoly price. This

is interesting in the sense that the open source support model may win the competition against established commercial software when switching cost is low enough compared to benefits from the open source software. Our results can explain the success of Red Hat Linux in the real-world operating systems market.

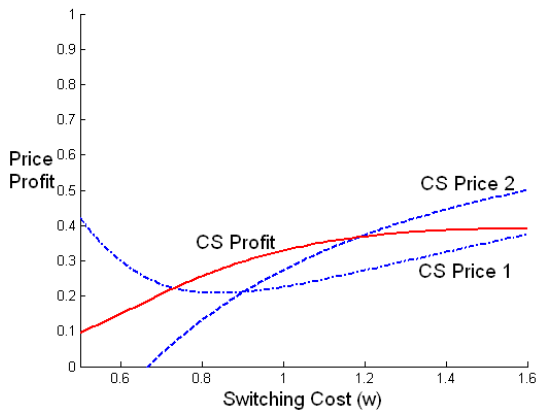


Figure 7. Price, Profit and Switching Cost (High Switching Cost)

Figure 7 illustrates the optimal first and second period prices and the profit for the commercial software. The open source support model is not viable in the presence of large switching cost. Consistent with the pricing strategy under competition against open source with dual-licensing, the commercial software vendor charges low price in the first period and increases its price in the second to maximize its installed base in the first period.

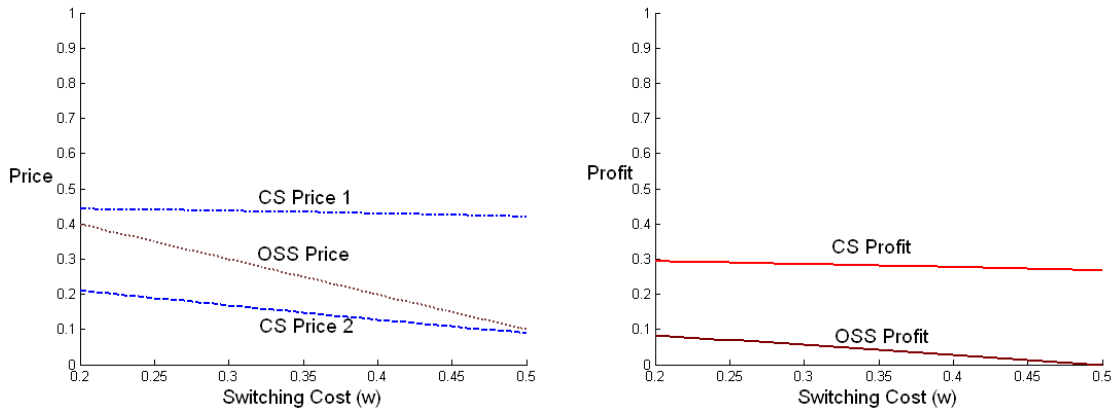


Figure 8. Price, Profit and Switching Cost (Medium Switching Cost)

Figure 8 shows the optimal prices and profits in the presence of medium switching cost. In this case, the software vendors split the market in the second period in a way that the commercial software vendor serves the low-type customers and the open source software vendor covers the high-type segment. Interestingly, as switching cost increases, prices and profits decrease for both vendors. When the switching cost is in a medium range, large switching cost may even hurt the commercial software vendor as well as the open source software vendor. As switching cost increases, both the commercial and the open source software vendors lower their prices in order to make up for decreased customers' utility, resulting in decreased profits.

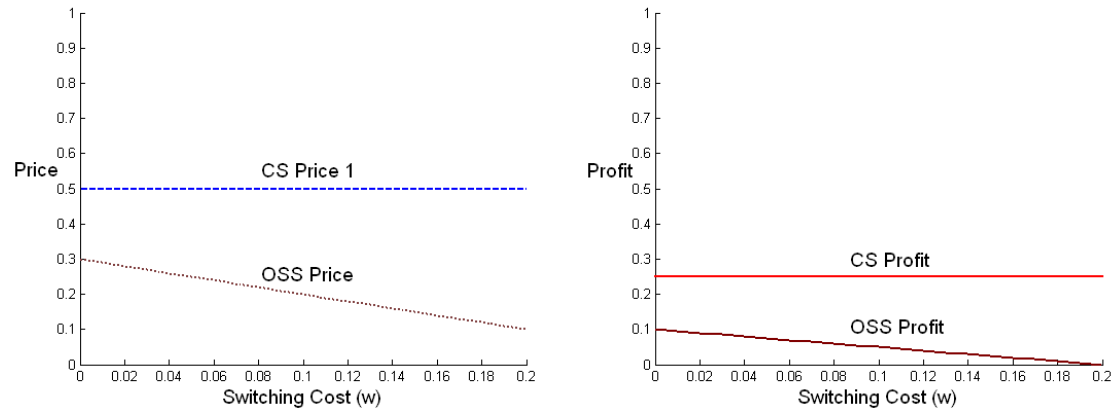


Figure 9. Price, Profit and Switching Cost (Low Switching Cost)

The optimal price and profit in the presence of low switching cost are shown in Figure 9. When the vendor independence benefit exceeds the switching cost, the open source software support model dominates the market in the second period. Our results imply that the open source software support model is viable when open source operating systems adoption brings vendor independence benefit that exceeds switching cost, and incurs support cost to the customers in both segments. The commercial software vendor maximizes its profit in the first period by charging monopoly price and does not care about keeping its customers in the second period. The open source software vendor lowers its price for support as switching cost increases.

4.7 Discussion

In October 2006, Oracle announced that it would provide technical support for Linux software distributed by Red Hat and that its support would be at about half the price of Red Hat's service (Flynn and Lohr 2006). In after-hours trading, Red Hat's stock plunged 16 percent, or $3.16, to $19.51. Since Red Hat makes profit on technical support, the Oracle move was seen as a hostile act which might eventually kill Red Hat's business. This view was shared by many industry experts such as Charles di Bona, an analyst at Sanford C. Bernstein & Company. According to Oracle press release, Oracle was able to offer its Unbreakable Linux program for substantially less than Red Had charges for its best support on behalf of its breadth and depth of technical expertise, advanced support technologies, and global reach including 7000 support staff in 17 global support centers (Shores 2006). "We believe that better support and lower support prices will speed the adoption of Linux, and we are working closely with our partners to make that happen", said Oracle CEO Larry Ellison.

The Oracle's involvement in the Linux support service market targeting Red Hat Linux has

been a popular discussion topic among industry experts who view this phenomenon as a possible risk to any businesses making money from open source software by selling support service. In the current state of the software industry, selling support service is the only possibility to make money from open source software when the copyright to the source code is not owned by any entity such as Linux. As mentioned earlier, this support service model has been considered to be a promising business model which brings profit to open software vendors. However, the problem is that anyone could sell support service to the customers of the same open source since nobody owns the copyright to Linux source code. As a result, anyone can sit and wait till the viability of a support service model for certain open source software is proven, and then enter the market selling the support service for the same open source software. When it is the large software vendor such as Oracle who may already have a large installed base for selling support service and a group of experience employees, it can become a serious threat to the businesses that have been successful with its support service model. This argument may explain the incentives of the established giant software vendors who have been getting involved in open source software industry as investors or observers.

Interestingly, some industry experts do not agree that Oracle Linux will kill Red Hat. Kerner (2006) argues that Oracle may not be able to match the quality of support service with Red Hat's service because Oracle has specific needs from Red Hat to better control and configure the software provided by Red Hat. Especially, Red Hat is on the verge of releasing RHEL 5, which includes a long list of new features that only Red Hat truly understands and is able to support. For example, a Xen virtualization manager and improved SELinux support are now appearing in Fedora Core 6. Although Oracle may reduce cost and offer support service at lower price, the customers may need to sacrifice quality of service. As a result, Oracle Linux may not hurt Red

Hat as much as expected. Rather, industry experts argue that it may hurt Red Hat's competitors such as Novell far worse since Oracle is basically making Red Hat Linux as an industry standard (Kerner 2006). This view is very interesting in the sense that Red Hat may be better off having a serious competitor in the market such as Oracle. If the benefit from an expanded network exceeds the loss from price-cut, it may be true that Red Hat will be better off having Oracle in the market. In this section, we examine the viability of the open source software support service model under competition in the presence of a large software vendor threatening its business model.

The potential threat from commercial software vendors who already have expertise in software support is an important factor to consider when open source software vendors plan to commercialize their software by selling support service. Due to the open nature of the software, it is likely that some vendors can exploit the opportunity to make profit by selling the support service to open source software whose support model is proven to be viable. Thus, if the commercial vendor's interruption would kill the open source software vendor's support business, then there would be no motivation for such business model. Thus, it is important to examine the viability of open source software support model in the presence of commercial vendor's potential threat. The analysis in this section is inspired by the real-world Linux support service market where Red Hat and Novell compete against each other, and Oracle enters with expertise in software support, which allows Oracle to bring lower marginal cost for support than both open source vendors. We investigate whether open source software support model is viable under competition against commercial software support expert. We also examine how the interruption to a certain open source support model affects the competition between existing open source software vendors.

We compare profits for the open source software vendors from two different markets: duopoly without Oracle and oligopoly with Oracle. We first examine the duopoly market where Red Hat and Novell are under Cournot competition, implying that they compete for the sales level. Then, we examine an equilibrium oligopoly model with a sequential entry in which three software vendors compete against each other to sell the support service for open source software. In the real-world Linux support service market, Red Hat and Novell compete against each other charging almost same prices for their support services. Our scenario is inspired by what has been happening in the Linux support service market. *R*, *N*, and *O* in the subscript denote Red Hat, Novell, and Oracle respectively. This oligopoly model is grounded on the Stackelberg model where Red Hat and Novell choose their sales levels first, then Oracle observes the sales levels by Red Hat and Novell, enters the market, and decides the sales level of the support service for Red Hat Linux. Finally, customers choose whether or not to buy support service, and from which provider. Our primary focus is Red Hat's profit at equilibrium in both markets. We investigate whether Oracle move is positive or negative on the Red Hat and Novell in order to verify industry experts' argument on the viability of the Red Hat's business model.

Duopoly Market without Oracle

We start with examining the duopoly market where Red Hat and Novell compete against each other. Let x_i denote the sales level, that is, the number of customers that a customer expects from vendor *i* to have and p_i represent the price that vendor *i* sets up for its Linux support service. Recall that *v* is the value of the Linux software that characterize and we assume that *v* is uniformly distributed on [0, 1]. α denotes the intensity of network externality, implying that high α means high level of customer appreciation of network size. In the software market,

customers often enjoy additional benefit from large networks. For example, more application software is available for the operating system with more users since application software vendors have higher motivation. Our model is grounded on the Cournot model. In the first stage, both vendors decide the sales level, and in the second, customers make their purchase decision: Red Hat or Novell or neither. The expected utility from each support service model is as follows:

$$u_R = v - p_R + \alpha x_R$$
$$u_N = v - p_N + \alpha x_N.$$

Since the value of the Linux software is assumed to be identical, both the vendors will have positive sales only when the following condition holds:

$$p_R - \alpha x_R = p_N - \alpha x_N.$$

The above condition says that the prices adjusted for the network size must be identical in order for both vendors to have positive sales. Let ϕ be the common value of the adjusted prices, that is,

$$\phi = p_R - \alpha x_R = p_N - \alpha x_N.$$

Recall that v is uniformly distributed on [0, 1]. Thus, the total market demand for Linux support service is $1-\phi$, which equals the total sales of both vendors, implying that, at equilibrium, the prices must be set such that $1-\phi = x_R + x_N$. The vendors offer the prices at equilibrium as follows:

$$p_R = 1 + \alpha x_R - (x_R + x_N)$$
$$p_N = 1 + \alpha x_N - (x_R + x_N).$$

In reality, the marginal cost for support service may be different across the providers. Providing support service to an additional customer may incur positive cost to the vendor. Unlike marginal cost to produce additional copy of software which is often negligible, it is reasonable to

assume positive marginal cost for support service. Let c_i denote the marginal cost for support service for vendor *i*. Given the demand, price, and the marginal cost, the profit for each vendor is as follows:

$$\pi_R = x_R(p_R - c_R) = x_R(1 + \alpha x_R - (x_R + x_N) - c_R)$$
$$\pi_N = x_N(p_N - c_N) = x_N(1 + \alpha x_N - (x_R + x_N) - c_N).$$

Note that both Red Hat and Novell want to set up the sales level at which they maximize their expected profit given their expectation about each other's sales level. Their profit maximization can be characterized by the first-order conditions:

$$\frac{\partial \pi_R}{\partial x_R} = 2(\alpha - 1)x_R - x_N - c_R + 1 = 0$$
$$\frac{\partial \pi_N}{\partial x_N} = 2(\alpha - 1)x_N - x_R - c_N + 1 = 0.$$

Solving the first-order conditions leads to the condition which the equilibrium sales levels (x_R^*, x_N^*, x_O^*) must satisfy as follows:

$$x_R^* = \frac{1}{2(1-\alpha)}(1 - x_N - c_R)$$
$$x_N^* = \frac{1}{2(1-\alpha)}(1 - x_R - c_N).$$

Let *D* in the subscript denote the duopoly case. The equilibrium sales levels by Red Hat and Novell are:

$$x_R^{D^*} = \frac{1}{4(1-\alpha)^2 - 1}(2(1-\alpha)(1-c_R) - (1-c_N))$$
$$x_N^{D^*} = \frac{1}{4(1-\alpha)^2 - 1}(2(1-\alpha)(1-c_N) - (1-c_R)).$$

At equilibrium in the duopoly market, Red Hat and Novell make profit as follows:

$$\pi_R^{D^*} = x_R^{D^*}(1 + \alpha x_R^{D^*} - (x_R^{D^*} + x_N^{D^*}) - c_R)$$
$$\pi_N^{D^*} = x_N^{D^*}(1 + \alpha x_N^{D^*} - (x_R^{D^*} + x_N^{D^*}) - c_N).$$

Oligopoly Market with Oracle

In this scenario, we aim to model the real-world Linux market where Red Hat and Oracle sell the support service for Red Hat Linux and Novell provides the service for its own Linux. We examine the viability of the existing support service model in the presence of a threat of an established software vendor who is able to cut cost and lower price. We study the role of network externality. Recall that parameter α denotes the strength of network externality in the Linux market. Our model reflects the view of software experts who argue that Oracle support is not as good as Red Hat support for Red Hat Linux by having parameter s which represents the quality deficiency of Oracle support service compared to Red Hat. This oligopoly model is grounded on the Stackelberg model where vendors make decision on the sales level sequentially. The structure of the game is as follows. In the first stage, Red Hat and Novell choose their sales levels which are observed by Oracle who enters the market, and decides the sales level of the support service for Red Hat Linux in the second stage. In the final stage, customers make their purchase decision. Customers expect utility from each vendor as follows:

$$u_R = v - p_R + \alpha(x_R + x_O)$$
$$u_N = v - p_N + \alpha x_N$$
$$u_O = v - s - p_O + \alpha(x_R + x_O).$$

Note that Red Hat and Oracle shares the same network by providing support service for the identical software Red Hat Linux while Novell manages its own network. Given the identical value of the Linux software, all the vendors will have positive sales only when the following condition holds:

$$p_R - \alpha(x_R + x_O) = p_N - \alpha x_N = p_O - \alpha(x_R + x_O) + s.$$

The above condition implies that the prices adjusted for the network size and quality deficiency must be same in order for all the three vendors to have positive sales. Recall that ϕ be

the common value of the adjusted prices:

$$\phi = p_R - \alpha(x_R + x_O) = p_N - \alpha x_N = p_O - \alpha(x_R + x_O) + s.$$

The total market demand for Linux support service is $1-\phi$, which equals the total sales of the three vendors. Thus, the prices must be set such that $1-\phi = x_R + x_N + x_O$. At equilibrium, the vendors offer the following prices:

$$\begin{aligned} p_R &= 1 + \alpha(x_R + x_O) - (x_R + x_N + x_O) \\ p_N &= 1 + \alpha x_N - (x_R + x_N + x_O) \\ p_O &= 1 + \alpha(x_R + x_O) - s - (x_R + x_N + x_O). \end{aligned}$$

Thus, the expected profit for each vendor is as follows:

$$\begin{aligned} \pi_R &= x_R(p_R - c_R) = x_R(1 + \alpha(x_R + x_O) - (x_R + x_N + x_O) - c_R) \\ \pi_N &= x_N(p_N - c_N) = x_N(1 + \alpha x_N - (x_R + x_N + x_O) - c_N) \\ \pi_O &= x_O(p_O - c_O) = x_O(1 + \alpha(x_R + x_O) - s - (x_R + x_N + x_O) - c_O). \end{aligned}$$

We examine the optimal choice of the output level by Oracle who makes a decision in the second stage after observing the choices made by Red Hat and Novell in the first stage. The first-order condition is

$$\frac{\partial \pi_O}{\partial x_O} = 1 + \alpha(x_R + x_O) - s - (x_R + x_N + x_O) - c_O + \alpha x_O - x_O = 0.$$

The optimal level of sales by Oracle is then $x_O^* = \frac{1}{2(1-\alpha)}(1 - (1-\alpha)x_R - x_N - s - c_O)$.

In the first stage, both Red Hat and Novell make their sales level choices simultaneously to maximize profits. Red Hat's expected profit at the Oracle's sales level choice in the second stage becomes

$$\begin{aligned} \pi_R &= x_R(1 + \alpha(x_R + x_O^*) - (x_R + x_N + x_O^*) - c_R) \\ &= \frac{1}{2} x_R(-(1-\alpha)x_R - x_N + 1 + s + c_O - 2c_R). \end{aligned}$$

The first-order condition is

$$\frac{\partial \pi_R}{\partial x_R} = \frac{1}{2}(-(1-\alpha)x_R - x_N + 1 + s + c_O - 2c_R - (1-\alpha)x_R) = 0.$$

Solving the first-order condition leads to the Red Hat's optimal sales level equal to

$$x_R^* = \frac{1}{2(1-\alpha)}(-x_N + 1 + s + c_O - 2c_R).$$

Novell's expected profit at the Oracle's sales level equals:

$$\begin{aligned}\pi_N &= x_N(1 + \alpha x_N - (x_R + x_N + x_O^*) - c_N) \\ &= \frac{1}{2(1-\alpha)} x_N(-(1-\alpha)x_R - (2(1-\alpha)^2 - 1)x_N + 2(1-\alpha)(1-c_N) - 1 + s + c_O).\end{aligned}$$

The first-order condition is then

$$\frac{\partial \pi_N}{\partial x_N} = \frac{1}{2(1-\alpha)} x_N(-(1-\alpha)x_R - (2(1-\alpha)^2 - 1)x_N + 2(1-\alpha)(1-c_N) - 1 + s + c_O - (2(1-\alpha)^2 - 1)x_N) = 0.$$

The optimal sales level equals

$$x_N^* = \frac{1}{4(1-\alpha)^2 - 2}(-(1-\alpha)x_R + 2(1-\alpha)(1-c_N) - 1 + s + c_O).$$

Thus, the equilibrium sales levels (x_R^*, x_N^*, x_O^*) must satisfy

$$\begin{aligned}x_R^* &= \frac{1}{2(1-\alpha)}(-x_N^* + 1 + s + c_O - 2c_R) \\ x_N^* &= \frac{1}{4(1-\alpha)^2 - 2}(-(1-\alpha)x_R^* + 2(1-\alpha)(1-c_N) - 1 + s + c_O) \\ x_O^* &= \frac{1}{2(1-\alpha)}(1 - (1-\alpha)x_R^* - x_N^* - s - c_O).\end{aligned}$$

O in the superscript denotes the oligopoly case. Solving for the above equations leads to the equilibrium sales levels by the three vendors in the market as follows:

$$x_R^{O*} = \frac{1}{(1-\alpha)(8(1-\alpha)^2-5)}(-2(1-\alpha)(1-c_N)+2(2(1-\alpha)^2-1)(1+s+c_O-2c_R)+1-s-c_O)$$

$$x_N^{O*} = \frac{1}{8(1-\alpha)^2-5}(s+2c_R+4(1-\alpha)(1-c_N)+c_O-3)$$

$$x_O^{O*} = \frac{1}{2(1-\alpha)}\left(1-\frac{2}{8(1-\alpha)^2-5}((1-\alpha)(1-c_N)+(2(1-\alpha)^2-1)(1+s+c_O-2c_R)-1+c_R)-s-c_O\right).$$

At equilibrium, the vendors make profits equal to

$$\pi_R^{O*} = x_R^{O*}(1+\alpha(x_R^{O*}+x_O^{O*})-(x_R^{O*}+x_N^{O*}+x_O^{O*})-c_R)$$
$$\pi_N^{O*} = x_N^{O*}(1+\alpha x_N-(x_R^{O*}+x_N^{O*}+x_O^{O*})-c_N)$$
$$\pi_O^{O*} = x_O^{O*}(1+\alpha(x_R^{O*}+x_O^{O*})-s-(x_R^{O*}+x_N^{O*}+x_O^{O*})-c_O).$$

We examine the viability of the Linux support model provided by Red Hat and Novell under the threat imposed by Oracle by comparing duopoly profit with oligopoly profit. We visualize our findings in Figure 10 and 11. Both figures illustrate how each vendor's profit varies with the intensity of network externality. Figure 10 shows the case of small network externality and Figure 11 visualizes the case of large network externality. We include duopoly profit for Red Hat and Novell in both figures in order to use it as a baseline for comparison with oligopoly profits. Both figures are based on the case where Oracle can cut the marginal cost for support about half of what Red Hat and Novell spend but the quality of Oracle's support is not as good as Red Hat and Novell.

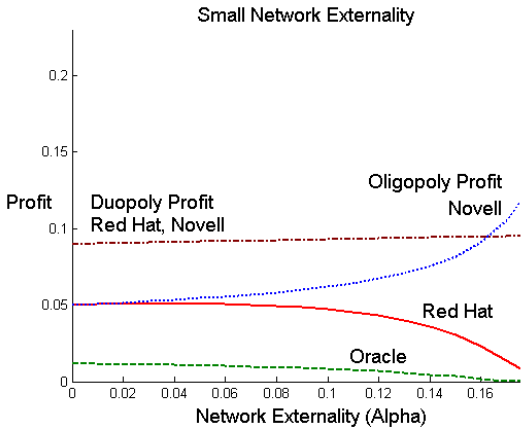


Figure 10. Profit and Network Externality (Small Network Externality)

Figure 10 illustrates duopoly profit for Red Hat and Novell and oligopoly profit for the three vendors including Oracle when the intensity of network externality is low, implying that customers do not much appreciate the network size. This result is consistent with the view that Oracle move will badly hurt Red Hat's business, explaining Red Hat stock plunge in hours after Oracle's announcement. We find that Red Hat's profit decreases with Oracle in the market, who sells the support service for the Red Hat Linux and that Oracle makes thin but positive profit margin, which explains Oracle's motivation to join the Linux support service market. Oracle's impact on Novell's business is not straightforward. When the intensity of network externality is low, Novell is also hurt by Oracle move, implying that Novell does not make as much as it does in the duopoly market without Oracle. Interestingly, when network externality is relatively strong, Novell can be better off with Oracle in the market. The very right side of figure 11 shows that Novell's oligopoly profit may exceed its duopoly profit.

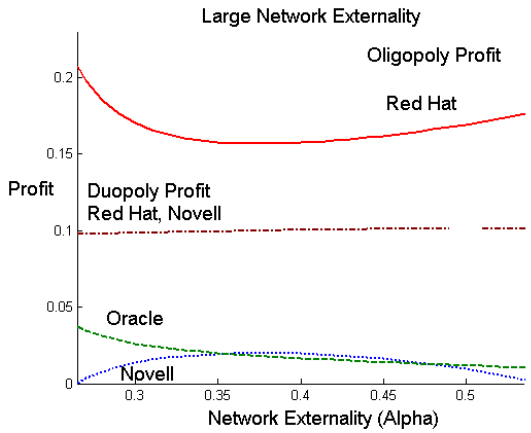


Figure 11. Profit and Network Externality (Large Network Externality)

Our findings for the case of large network externality are visualized in Figure 11. Interestingly, our results support industry experts' argument that Oracle's selling support for Red Hat Linux may hurt Red Hat's competitor Novell far worse by making Red Hat Linux as an industry standard. Figure 11 shows that Red Hat may make even higher profit with Oracle in the market than without Oracle when customers enjoy much benefit from a large network. When customers appreciate network size, it is valuable to have a large number of customers and to become an industry standard. Thus, what Red Hat gains from being such an industry standard may exceed what it loses from having another competitor, Oracle as long as Red Hat maintains its superiority in terms of service quality for its own software in spite of higher marginal cost than Oracle. Unlike in the case of small network externality, Oracle move hurt Novell's business badly in the presence of large network externality. Since customers enjoy much benefit from a network, they may choose Red Hat or Oracle over Novell who manages its own independent and incompatible network. Among those customers who choose Red Hat–Oracle network, customers

who value quality more than price will choose Red Hat while others who are concerned more about price than quality will buy from Oracle. Our findings are very interesting in the sense that an open source software vendor who makes money with selling support service may become more successful when it has a strong competitor who has cost advantage. Our results imply that the start-up companies who think about making money with selling support service for open source software whose source code is publicly available do not need to be discouraged by the potential threat of the large established software vendors who may want to target its own business model. As long as technical superiority is maintained and customers value the size of the network, the support service model is viable despite the threat of large software vendors who are able to cut the cost greatly.

4.8 Conclusion

In this paper, we compare three different software pricing mechanisms: commercial software, open source dual-licensing, and open source software support. We start with examining the optimal pricing strategies of a software vendor with each pricing scheme under monopoly. We find that the monopoly price for open source software under dual-licensing is higher than the commercial software price, hence higher profit, if total flexibility benefit for the high-type customers exceeds total support cost for the low-type customers. With a support model, the open source vendor charges for support at higher price than the commercial software price, but it makes a lower profit than the commercial software vendor. Then we investigate whether the open source software vendor prefers one pricing mechanism over the other in the monopoly market. Our result shows that the dual-licensing model is more profitable when the difference

between flexibility benefit and support cost is high. Otherwise, the support model is preferred. We identify the factors that affect the open source software vendor's decision.

We extend our model to a duopoly competition setting. In a market with competition between a commercial software vendor and an open source software vendor with a dual-licensing model, the open source software vendor serves high-type customers only by charging more than the commercial software vendor that covers low-type segment only. Our result supports the viability of the dual-licensing model for the open source software in the presence of competition. When there is a competition between a commercial software vendor and an open source software vendor with a support model, the commercial software vendor competing against the open source support model charges a price equal to the open source vendor's marginal cost of support. With this pricing strategy of the commercial software vendor, the support model brings zero profit to the open source software vendor.

We examine the viability of the open source support model when the open source software quality is different from commercial software quality. Interestingly, we find that the open source software support model is viable in the presence of quality asymmetry no matter whether the quality of the open source software is higher or lower than the commercial software. Even when the open source software provides lower quality, the support model is profitable. This can be explained as market expansion of the commercial software vendor. When both vendors offer the same quality software the high-type customers have no incentive to buy the commercial software since the open source software is free and they do not need any support. However, when the commercial software is superior to the open source software, some high-type customers may buy the commercial software, implying that the commercial software vendor can target such high-type customers as well.

With a generalized model, we further study competition between commercial and open source software, motivated by what happens in the real-world database and operating systems markets. We model the different sources of the benefit from different open source software and assume that support cost may incur to both types of customers. We set up a two-period game to study the role of switching cost that customers face when they switch from established commercial software to open source software. We find that neither the dual-licensing nor the support model is profitable in the presence of large switching cost that customers must incur to switch away from commercial software. When the switching cost is lower than a certain threshold, software vendors split the market in the second period in a way that the commercial software serves low-type customers and the open source software vendor covers high-type segment. Interestingly, we find that the open source software with support may dominate the market in the second period when the vendor independence benefit exceeds switching cost.

Finally, we examine the open source software support service market where the copyright to the source code is publicly owned and anyone can sell the support service for the same open source software. Inspired by the real-world Linux support service market where Oracle sells support service for Red Hat Linux, we investigate whether an established software vendor's move like Oracle hurts or helps an open source software service provider like Red Hat. We find that Red Hat's business model is hurt by Oracle move in the presence of small network externality. We also find that Red Hat may be better off having Oracle in the market in the presence of large network externality. When customers' valuation of the network size is relatively high, Oracle move hurts Novell while it may help Red Hat. Our results suggest that the relatively new software vendor who has been successful selling the support service for the open source software do not have to be discouraged by the threat from large software vendors who

may enter the market and sell the same support service as long as superiority in terms of support quality is maintained.

Our paper contributes to the literature in the following ways. First, this paper is the first study that examines the issue of pricing open source software through an economic lens. In spite of the growing interest in the commercial open source software among industry experts and jurists, no academic study has shown the viability of the pricing models of for-profit open source software. We identify the factors that affect the viability of the pricing models for open source software and find the conditions under which each model can be successful. Second, our result can give pricing guidelines to the open source software vendors, which is not clear in the current state. Finally, we model the motivation for and the barrier to open source adoption, which provides a better picture of the open source software market. Considering such factors that are consistent with survey statistics, may allow us to better understand the issue of open source software.

Appendix

Proof of Proposition 1

Note that $p_D^* - p_C^* = \frac{\mu f - (1-\mu)s + 1}{2} - \frac{1}{2} = \frac{1}{2}\{\mu f - (1-\mu)s\}$. Thus,

$$p_D^* > p_C^* \quad \text{if} \quad \mu f > (1-\mu)s$$
$$p_D^* < p_C^* \quad \text{if} \quad \mu f < (1-\mu)s.$$

The profit difference is $\pi_D^* - \pi_C^* = \frac{1}{4}\{\mu f - (1-\mu)s + 1\}^2 - \frac{1}{4}$. Since $p_D^* = \frac{\mu f - (1-\mu)s + 1}{2} \geq 0$,

$$\mu f - (1-\mu)s + 1 \geq 0.$$

Thus, $\pi_D^* - \pi_C^* > 0 \quad \text{if} \quad \mu f - (1-\mu)s + 1 - 1 > 0$. Therefore,

$$\pi_D^* > \pi_C^* \quad \text{if} \quad \mu f > (1-\mu)s$$
$$\pi_D^* < \pi_C^* \quad \text{if} \quad \mu f < (1-\mu)s.$$

The price difference between open source software support and commercial software is

$$p_S^* - p_C^* = \frac{1 + c_S}{2} - \frac{1}{2} = \frac{c_S}{2} > 0.$$

Therefore, $p_S^* > p_C^*$. Note that $\pi_S^* - \pi_C^* = \frac{1}{4}(1-\mu)(1-c_S)^2 - \frac{1}{4}$. The marginal cost is bounded

by price, i.e. $c_S \leq p_S^* = \frac{1 + c_S}{2}$. Therefore, $0 < c_S \leq 1$. Since $0 \leq \mu \leq 1$, we have

$0 \leq (1-\mu)(1-c_S)^2 < 1$. Thus, $\pi_S^* - \pi_C^* = \frac{1}{4}(1-\mu)(1-c_S)^2 - \frac{1}{4} < 0$. Therefore, $\pi_S^* < \pi_C^*$. ***QED.***

Proof of Proposition 2

Note that

$$\pi_D^* - \pi_S^* = \frac{1}{4}\{\mu f - (1-\mu)s + 1\}^2 - \frac{1}{4}(1-\mu)(1-c_S)^2.$$

Since $\mu f - (1-\mu)s + 1 > 0$ and $c_S \leq 1$,

$$\pi_D^* > \pi_S^* \quad \text{if} \quad \mu f - (1-\mu)s > \sqrt{1-\mu}(1-c_S) - 1$$

$$\pi_D^* < \pi_S^* \quad \text{if} \quad \mu f - (1-\mu)s < \sqrt{1-\mu}(1-c_S) - 1.$$

QED.

Proof of Proposition 3

Recall that a customer's utility from commercial software would be $u_C = v - p_C$. By using open source software sold under a dual-licensing model, a high-type and a low-type customer enjoys the following benefits:

$$u_{DH} = v - p_D + f$$
$$u_{DL} = v - p_D - s.$$

Thus, the demands for commercial and open source software from high-type and low-type customers are as follows:

<u>High-Type Customers</u>

$$q_{CH}(p_C, p_D) = \begin{cases} q_H(p_C) & \text{if} \quad p_C < p_D - f \\ \frac{1}{2} q_H(p_C) & \text{if} \quad p_C = p_D - f \\ 0 & \text{if} \quad p_C > p_D - f \end{cases} \quad q_{DH}(p_D, p_C) = \begin{cases} q_H(p_D) & \text{if} \quad p_D - f < p_C \\ \frac{1}{2} q_H(p_D) & \text{if} \quad p_D - f = p_C \\ 0 & \text{if} \quad p_D - f > p_C. \end{cases}$$

<u>Low-Type Customers</u>

$$q_{CL}(p_C, p_D) = \begin{cases} q_L(p_C) & \text{if} \quad p_C < p_D + s \\ \frac{1}{2} q_L(p_C) & \text{if} \quad p_C = p_D + s \\ 0 & \text{if} \quad p_C > p_D + s \end{cases} \quad q_{DL}(p_D, p_C) = \begin{cases} q_L(p_D) & \text{if} \quad p_D + s < p_C \\ \frac{1}{2} q_L(p_D) & \text{if} \quad p_D + s = p_C \\ 0 & \text{if} \quad p_D + s > p_C. \end{cases}$$

Note that the commercial software vendor loses all the demand if it charges price p_C above $p_D + s$. Thus, $p_C \in (p_D + s, 1)$ is a dominated strategy. The commercial software vendor would charge price at $p_C \in (0, p_D + s)$ to guarantee the demand from the customers in the low-type segment. From the open source software vendor's perspective, charging price at p_D such that $p_D - f > p_C$ is a dominated strategy since such pricing would lead to zero demand for the open source software vendor. Thus, the open source software vendor would charge price at $p_D \in (0, p_C + f)$ and serve the high-type customers. If an equilibrium exists, at equilibrium, the commercial software vendor uses a pricing strategy to serve the low-type customer segment while the open source software vendor serves the high-type customer segment. Consider the profit for the commercial software vendor:

$$\pi_C = (1-\mu)(1-p_C)p_C.$$

The optimization problem for the commercial software vendor is

$$\underset{p_C}{\text{Max}}\, \pi_C \quad \text{subject to } p_C \le p_D + \text{s}.$$

The first-order condition for π_C is $\dfrac{\partial \pi_C}{\partial p_C} = (1-\mu)(1-2p_C) = 0,$ resulting in the profit-maximizing price as

$$p_C^* = \begin{cases} \dfrac{1}{2} & \text{if} \quad \dfrac{1}{2} \le p_D + \text{s} \\ p_D + \text{s} & \text{if} \quad \dfrac{1}{2} > p_D + \text{s}. \end{cases}$$

The open source software vendor's profit would be $\pi_D = \mu(1-p_D+f)p_D$. The optimization problem for the commercial software vendor is

$$\underset{p_D}{\text{Max}}\, \pi_D \quad \text{subject to } p_D \le p_C + f.$$

The first-order condition for π_D is $\frac{\partial \pi_D}{\partial p_D} = \mu(1-2p_D+f) = 0,$ leading to the optimal price as

$$p_D^* = \begin{cases} \frac{1+f}{2} & \text{if} \quad \frac{1+f}{2} \le p_C + f \\ p_C + f & \text{if} \quad \frac{1+f}{2} > p_C + f. \end{cases}$$

Now, we investigate whether $(p_C^*, p_D^*) = (\frac{1}{2}, \frac{1+f}{2})$ is a Nash equilibrium. Consider the case when the open source software vendor offers price at $p_D^* = \frac{1+f}{2}$. We investigate whether the commercial software vendor has incentive to deviate from $p_C^* = \frac{1}{2}$. Since $f > 0$ and $s > 0$,

$$p_D^* - f = \frac{1-f}{2} < p_C^* = \frac{1}{2} < p_D^* + s = \frac{1+f}{2} + s.$$

Suppose that $f \ge 1$. Then $p_D^* - f = \frac{1-f}{2} \le 0$. Thus, the commercial software vendor cannot offer any price below $p_D^* - f$, meaning that the commercial software vendor cannot serve any customer in the high-type segment. Since $p_C^* = \frac{1}{2}$ is the price that maximizes profit from serving low-type segment, the commercial software vendor has no incentive to deviate from $p_C^* = \frac{1}{2}$. Thus, $p_C^* = \frac{1}{2}$ is a best response to $p_D^* = \frac{1+f}{2}$. Now, suppose that $0 < f < 1$. Then $p_D^* - f = \frac{1-f}{2} > 0$. In this case, the commercial software vendor has two choices as below:

Strategy 1: Stay at $p_C^* = \frac{1}{2}$ and serve low-type customers only.

Strategy 2: Deviate to $p_C^* = \frac{1-f}{2}$ and serve both segments.

Denote π_C^1 and π_C^2 be the profit for the commercial software vendor playing strategy 1 and 2, respectively. Then

$$\pi_C^1 = (1-\mu)(1-p_C^*)p_C^* = (1-\mu)(1-\frac{1}{2})\frac{1}{2} = \frac{1-\mu}{4}$$
$$\pi_C^2 = (1-p_C^*)p_C^* = (1-\frac{1-f}{2})\frac{1-f}{2} = \frac{1-f^2}{4}.$$

If $\pi_c^1 \geq \pi_c^2$, the commercial software vendor has no incentive to deviate. Note that $\pi_C^1 \geq \pi_C^2 \Leftrightarrow f \geq \sqrt{\mu}$. Therefore, $p_C^* = \frac{1}{2}$ is a best response to $p_D^* = \frac{1+f}{2}$ if $f \geq \sqrt{\mu}$.

Now, we investigate $p_D^* = \frac{1+f}{2}$ is a best response of the open source software vendor to $p_C^* = \frac{1}{2}$ offered by the commercial software. Suppose that $s \geq \frac{1}{2}$. The open source software vendor cannot charge any price p_D such that $p_D + s < p_C^* = \frac{1}{2}$. Thus, the open source software vendor cannot serve any customer in the low-type segment. Since $p_D^* = \frac{1+f}{2}$ is the optimal price when serving low-type customers only, the open source software vendor has no incentive to deviate from $p_D^* = \frac{1+f}{2}$. Thus, $p_D^* = \frac{1+f}{2}$ is a best response to $p_C^* = \frac{1}{2}$. Now, suppose that $0 < s < \frac{1}{2}$. Then $p_C^* - s = \frac{1}{2} - s > 0$. Thus, the open source software vendor has two choices as follows:

Strategy 1: Stay at $p_D^* = \frac{1+f}{2}$ and serve high-type customers only

Strategy 2: Deviate to $p_D^* = \frac{1}{2} - s$ and serve both segments.

Denote π_D^1 and π_D^2 be the profit for the commercial software vendor playing strategy 1 and 2, respectively. Then

$$\pi_D^1 = \mu(1-p_D^* + f)p_D^* = \mu(1-\frac{1+f}{2}+f)\frac{1+f}{2} = \frac{\mu(1+f)^2}{4}$$

$$\pi_D^2 = \{\mu(1-p_D^* + f) + (1-\mu)(1-p_D^* - s)\}p_D^* = \{\mu(1-\frac{1}{2}+s+f) + (1-\mu)(1-\frac{1}{2}+s-s)\}(\frac{1}{2}-s)$$

$$= \frac{1}{4}(2\mu f + 2\mu s + 1)(1-2s).$$

Note that $\pi_C^1 \geq \pi_C^2 \Leftrightarrow \mu(1+f)^2 \geq (2\mu f + 2\mu s + 1)(1-2s)$. Thus, $p_D^* = \frac{1+f}{2}$ is a best response to $p_C^* = \frac{1}{2}$ if the following conditions hold:

$$s \geq \frac{1}{2} \quad \text{or} \quad \left\{0 < s < \frac{1}{2} \quad \text{and} \quad \mu(1+f)^2 \geq (2\mu f + 2\mu s + 1)(1-2s)\right\}.$$

Therefore, $(p_C^*, p_D^*) = (\frac{1}{2}, \frac{1+f}{2})$ is a Nash equilibrium when

$$\left\{f \geq \sqrt{\mu} \quad \text{and} \quad s \geq \frac{1}{2}\right\} \quad \text{or} \quad \left\{f \geq \sqrt{\mu}, \quad 0 < s < \frac{1}{2} \quad \text{and} \quad \mu(1+f)^2 \geq (2\mu f + 2\mu s + 1)(1-2s)\right\}.$$

QED.

Proof of Proposition 4

Consider the high-type customers. Since $u_C = v - p_C < v + f = u_{SH}$, high-type customers always choose the open source software. Thus, both vendors compete for the low-type segment only. The demand for the commercial and the open source software would be

$$q_{CL}(p_C, p_S) = \begin{cases} q_L(p_C) & \text{if} \quad p_C < p_S \\ \frac{1}{2}q_L(p_C) & \text{if} \quad p_C = p_S \\ 0 & \text{if} \quad p_C > p_S \end{cases} \quad \text{and} \quad q_{SL}(p_S, p_C) = \begin{cases} q_L(p_S) & \text{if} \quad p_S < p_C \\ \frac{1}{2}q_L(p_S) & \text{if} \quad p_S = p_C \\ 0 & \text{if} \quad p_S > p_C. \end{cases}$$

Note that

$$\pi_C = (1-\mu) q_{CL} p_C$$
$$\pi_S = (1-\mu) q_{SL} (p_S - c_S).$$

First, no vendor charges less than marginal cost to avoid making negative profit. That is,

$$p_C \geq 0 \quad \text{and} \quad p_S \geq c_S.$$

Second, there cannot be an equilibrium in which both p_C and p_S are strictly above c_S. Consider the case where $p_C > p_S > c_S$. Then the commercial software vendor can obtain the entire demand and increases its profit by charging at $p_S - \varepsilon$ where ε is a small and positive number. If $p_S > p_C > c_S$, the open source software vendor has an incentive to lower its support charge to $p_C - \varepsilon$. Suppose that $p_C = p_S > c_S$. Both vendors still want to lower their prices by ε so that they can increase profit. Finally, the commercial software vendor can obtain the entire demand by charging $c_S - \varepsilon$. The equilibrium prices can be obtained as

$$p_C^* = p_S^* = c_S.$$

The commercial software has the whole market and the open source software vendor makes zero profit. The profit for the commercial software vendor would be

$$\pi_C^* = (1-\mu) q_{CL}^* p_C^* = (1-\mu)(1-c_S) c_S.$$

When $\frac{1}{2} < c_S < 1$, the commercial vendor charges its monopoly price $p_C^{Monopoly} = \frac{1}{2}$ and

$$\pi_C^{Duopoly} = \frac{1}{4}(1-\mu) < \frac{1}{4} = \pi_C^{Monopoly}.$$

QED.

Proof of Proposition 5

Suppose that $\alpha > 1$. Recall that

$$p_C^* = \frac{\alpha\{2\alpha + (1-\mu)c_S - 2(1+\mu f)\}}{4\alpha + \mu - 1} \quad \text{and} \quad p_S^* = \frac{\alpha(1+2c_S) - (1+\mu f)}{4\alpha + \mu - 1}.$$

Note that

$$\begin{aligned}\frac{\partial p_C^*}{\partial \alpha} &= \frac{1}{(4\alpha + \mu - 1)^2}[(4\alpha + \mu - 1)\{4\alpha + (1-\mu)c_S - 2(1+\mu f)\} - 8\alpha^2 - 4\alpha\{(1-\mu)c_S - 2(1+\mu f)\}] \\ &= \frac{1}{(4\alpha + \mu - 1)^2}[4\alpha(2\alpha + \mu - 1) - (1-\mu)\{(1-\mu)c_S - 2(1+\mu f)\}]\end{aligned}$$

Since $\alpha > 1$ and $0 \le \mu \le 1$, $4\alpha(2\alpha + \mu - 1) > 0$. Note that $c_S \le 1$ and $f \ge 0$. Thus,

$(1-\mu)c_S - 2(1+\mu f) < 0$. Therefore, $\frac{\partial p_C^*}{\partial \alpha} > 0$.

Note that

$$\begin{aligned}\frac{\partial p_S^*}{\partial \alpha} &= \frac{1}{(4\alpha + \mu - 1)^2}\{(4\alpha + \mu - 1)(1+2c_S) - 4\alpha(1+2c_S) + 4(1+\mu f)\} \\ &= \frac{1}{(4\alpha + \mu - 1)^2}\{\mu(4f + 1 + 2c_S) + 3 - 2c_S\}.\end{aligned}$$

Since $\mu > 0$, $f > 0$, and $0 < c_S < 1$, $4f + 1 + 2c_S > 0$ and $3 - 2c_S > 0$. Therefore, $\frac{\partial p_S^*}{\partial \alpha} > 0$.

Recall that the first-order condition for the commercial software vendor's profit is

$$\frac{\partial \pi_C}{\partial p_C} = \frac{1}{\alpha - 1}(1-\mu)(p_C - 2\alpha p_S + \alpha c_S) = 0.$$

Thus, $p_C^* = \alpha(2p_S^* - c_S)$. Let δ be the difference between the commercial software price and the support charge for open source, i.e., $\delta = p_C^* - p_S^* = (2\alpha - 1)p_S^* - \alpha c_S$. Then

$$\frac{\partial \delta}{\partial \alpha} = 2p_S^* + (2\alpha - 1)\frac{\partial p_S^*}{\partial \alpha} - c_S.$$

Since $c_S \le p_S < 2p_S$ and $\alpha > 1$, $2p_S^* - c_S > 0$ and $2\alpha - 1 > 0$. We show that $\frac{\partial p_S^*}{\partial \alpha} > 0$. Therefore, $\frac{\partial \delta}{\partial \alpha} > 0$. Now suppose that $0 < \alpha < 1$. Note that

$$p_C^* = \frac{c_S + 1 - \alpha}{7 - 4\alpha} \quad \text{and} \quad p_S^* = \frac{2(c_S + 1 - \alpha)(2 - \alpha)}{7 - 4\alpha}.$$

Thus, $\frac{\partial p_C^*}{\partial \alpha} = \frac{1}{(7-4\alpha)^2}\{-(7-4\alpha) + 4(c_S + 1 - \alpha)\} = \frac{4c_S - 3}{(7-4\alpha)^2}$. Therefore, $\frac{\partial p_C^*}{\partial \alpha} > 0$ if $\frac{3}{4} < c_S < 1$ and $\frac{\partial p_C^*}{\partial \alpha} < 0$ if $0 < c_S < \frac{3}{4}$. Note that

$$\begin{aligned}\frac{\partial p_S^*}{\partial \alpha} &= \frac{2}{(7-4\alpha)^2}[(7-4\alpha)(2\alpha - c_S - 3) + 4\{\alpha^2 - \alpha(c_S + 3) + 2(c_S + 1)\}] \\ &= \frac{2}{(7-4\alpha)^2}\{-2(2\alpha - 5)(\alpha - 1) + c_S - 3\}.\end{aligned}$$

Since $2\alpha - 5 < 0$, $\alpha - 1 < 0$, and $c_S - 3 < 0$, $\frac{\partial p_S^*}{\partial \alpha} < 0$. Finally, $p_S^* = 2(2-\alpha)p_C^* > 2p_C^* > p_C^*$.

QED.

Proof of Proposition 6

Case 1: High Switching Cost ($w > m - t - s_H$)

Assume that $w > m - t - s_H$. Consider the commercial vendor's pricing decision given the open source software vendor's price p_D. Note that $w + p_D - m + t + s_H > 0$ for any p_D since $w > m - t - s_H$. Thus, the commercial vendor can serve both high-type and low-type customers and dominate the market by charging p_1 and p_2 such that $wp_1 + p_2 < w + p_D - m + t + s_H$. By charging prices slightly above, the commercial software vendor looses the entire high-type segment. Thus, for any price that the open source software

vendor charges, the commercial software vendor's best response is to offer prices satisfying $wp_1 + p_2 < w + p_D - m + t + s_H$. Since the open source software vendor knows what the commercial software vendor will play given its price, the open source software vendor is indifferent among any prices in a defined range. When $w > m - t - s_H$, the open source software vendor makes zero profit. This game is similar to sequential matching pennies game where the first mover is indifferent between two choices. This result has economic implications. First, consider market entry of the open source software vendor. Suppose that entering the market incurs positive cost to the open source software vendor. Then, the open source software vendor does not have any incentive to enter the market. When the switching cost parameter is high $w > m - t - s_H$ and the entry is the choice of the open source software vendor, the commercial software vendor plays pricing strategy to set up an entry barrier so that it can enjoy monopoly profit in both period 1 and period 2. Now, suppose that entry is not the open source software vendor's choice. Then an infinite number of equilibria exist. Consider one among those equilibria, which is,

$$p_D^* = 0 \text{ and } (p_1^*, p_2^*) \text{ such that } wp_1 + p_2 < w + p_D - m + t + s_H .$$

This is actually the unique equilibrium of the simultaneous game where both the commercial and the open source software vendors make their pricing decisions in period 2 at the same time. The commercial software vendor's profit is

$$\begin{aligned} \pi_C &= q_1 p_1 + q_2 p_2 \\ &= (1 - p_1) p_1 + (1 - p_1)(1 - p_2) p_2 . \end{aligned}$$

Let $d_H = m - t - s_H$. The optimization problem is then

$$\underset{p_1, p_2}{\text{Max}} \, \pi_C \quad \text{subject to } wp_1 + p_2 \le w - d_H .$$

The Lagrangean function can be defined as

$$L(p_1, p_2, \lambda) = (1-p_1)p_1 + (1-p_1)(1-p_2)p_2 - \lambda(wp_1 + p_2 - w + d_H).$$

The Kuhn-Tucker conditions are

$$\frac{\partial L}{\partial p_1} = 1 - 2p_1 - (1-p_2)p_2 - w\lambda = 0$$
$$\frac{\partial L}{\partial p_2} = (1-p_1)(1-2p_2) - \lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = wp_1 + p_2 - w + d_H = 0.$$

From $\frac{\partial L}{\partial \lambda}$, we have $p_2 = w - d_H - wp_1$. From $\frac{\partial L}{\partial p_1}$ and $\frac{\partial L}{\partial p_2}$,

$$1 - 2p_1 - (1-p_2)p_2 = w(1-p_1)(1-2p_2)$$
$$\Leftrightarrow p_2^2 + p_2(2w - 2wp_1 - 1) + p_1(w-2) + 1 - w = 0.$$

Replacing p_2 in the above equation with $w - d_H - wp_1$ in the above equation leads to

$$(w - d_H - wp_1)^2 + (w - d_H - wp_1)(2w - 2wp_1 - 1) + p_1(w-2) + 1 - w = 0$$
$$\Leftrightarrow 3w^2 p_1^2 - 2p_1(3w^2 - 2wd_H - w + 1) + (3w^2 - 4wd_H - 2w + {d_H}^2 + d_H + 1) = 0.$$

Solving the above equation gives the equilibrium prices as follows:

$$p_1^* = \frac{1}{3w^2}\left(3w^2 - 2wd_H - w + 1 - \sqrt{w^2({d_H}^2 + d_H + 4) - 2w(2d_H + 1) + 1}\right)$$
$$p_2^* = \frac{1}{3}\left(1 - d_H - \frac{1}{w} + \frac{1}{w}\sqrt{w^2({d_H}^2 + d_H + 4) - 2w(2d_H + 1) + 1}\right)$$

Case 2: Low Switching Cost ($w < m - t - s_H$)

Assume that $w < m - t - s_H$. Then the open source software vendor charges

$$p_D \le m - t - s_H - w = d_H - w$$

so that it guarantees to serve high-type customers and maximizes its profit. Note that the OSS vendor's profit is

$$\begin{aligned}\pi_D &= \mu q_D p_D \\ &= \mu(1-p_1)(1-p_D+d_H-w(1-p_1))p_D\end{aligned}$$

Then profit-maximizing price is $p_D^{Max} = \frac{1}{2}(1+d_H-w(1-p_1))$.

Note that

$$p_D^{Max}-(d_H-w) = \frac{1}{2}(1+d_H-w(1-p_1))-(d_H-w) = \frac{1}{2}(1+w+wp_1-d_H).$$

Recall that net benefit to high type customer does not exceed sum of maximum possible product value and maximum possible switching cost. That is $d_H < 1+w$. Therefore, $p_D^{Max} > (d_H-w)$. The OSS vendor's optimal price is then

$$p_D^* = f_H - s_H - w = d_H - w.$$

Then, the commercial software vendor offers prices such that

$$\underset{p_1,p_2}{\text{Max}}\,\pi_C \quad \text{subject to } wp_1+p_2 \le d_H - w - s_L + w = d_H - s_L.$$

Let $d_H = f_H - s_H$. Solving for the subgame perfect equilibrium leads to the following optimal prices:

$$p_1^* = \frac{1}{3w^2}\left(w^2+2w(d_H-s_L)-w+\frac{1}{1-\mu} - \sqrt{\left(w^2+2w(d_H-s_L)-w+\frac{1}{1-\mu}\right)^2 - 3w^2\left((d_H-s_L)^2+2w(d_H-s_L)-(d_H-s_L)-w+\frac{1}{(1-\mu)}\right)}\right)$$

$$p_2^* = d_H - s_L - \frac{1}{3w}\left(w^2+2w(d_H-s_L)-w+\frac{1}{1-\mu} - \sqrt{\left(w^2+2w(d_H-s_L)-w+\frac{1}{1-\mu}\right)^2 - 3w^2\left((d_H-s_L)^2+2w(d_H-s_L)-(d_H-s_L)-w+\frac{1}{(1-\mu)}\right)}\right)$$

$p_D^* = m - t - s_H - w = d_H - w$. ***QED.***

Proof of Proposition 7

Note that a customer will play a strategy CC if the customer enjoys positive utility: $v - p_1 > 0$ and $v - p_2 > 0$. Comparing CC and CO is not trivial. A high-type customer stays with commercial software in the second period if

$$v - p_1 + v - p_2 > v - p_1 + v + f - p_S - wx_1 + k .$$

Rearranging the above condition leads to

$$wp_1 + p_2 < w + p_S - f - k.$$

A low-type customer prefers commercial software to open source software in the second period if

$$v - p_1 + v - p_2 > v - p_1 + v - p_S - wx_1 + k$$

which can be rewritten as

$$wp_1 + p_2 < w + p_S - k.$$

Case 1: High Switching Cost ($w > f + k$)

Suppose that $w > f + k$. Then $w - f - k > 0$. The commercial software vendor can dominate the market in period 2 if it plays a pricing strategy such that $wp_1 + p_2 < p_S + w - f - k$ where p_S is the price for open source support. The open source software vendor cannot charge any price below marginal cost, so the optimal price is $p_S^* = c_S$. After observing the open source software vendor's price for support, the commercial software vendor charges prices such that

$$wp_1 + p_2 < c_S + w - f - k .$$

The commercial software vendor's profit is

$$\begin{aligned} \pi_C &= q_1 p_1 + q_2 p_2 \\ &= (1 - p_1) p_1 + (1 - p_1)(1 - p_2) p_2 . \end{aligned}$$

The optimization problem is then

$$\underset{p_1,p_2}{\text{Max}}\,\pi_C \quad \text{subject to } wp_1 + p_2 \le c_S + w - f - k.$$

The Lagrangean function can be defined as

$$L(p_1, p_2, \lambda) = (1-p_1)p_1 + (1-p_1)(1-p_2)p_2 - \lambda(wp_1 + p_2 - c_S - w + f + k).$$

The Kuhn-Tucker conditions are

$$\frac{\partial L}{\partial p_1} = 1 - 2p_1 - (1-p_2)p_2 - w\lambda = 0$$
$$\frac{\partial L}{\partial p_2} = (1-p_1)(1-2p_2) - \lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = wp_1 + p_2 - c_S - w + f + k = 0.$$

From $\frac{\partial L}{\partial \lambda}$, we have $p_2 = c_S + w - f - k - wp_1$. From $\frac{\partial L}{\partial p_1}$ and $\frac{\partial L}{\partial p_2}$,

$$1 - 2p_1 - (1-p_2)p_2 = w(1-p_1)(1-2p_2)$$
$$\Leftrightarrow p_2^2 + p_2(2w - 2wp_1 - 1) + p_1(w-2) + 1 - w = 0.$$

Replacing p_2 in the above equation with $w + c_S - f - k - wp_1$ leads to

$$(c_S + w - f - k - wp_1)^2 + (c_S + w - f - k - wp_1)(2w - 2wp_1 - 1) + p_1(w-2) + 1 - w = 0$$
$$\Leftrightarrow 3w^2 p_1^2 - 2p_1(3w^2 + 2wc_S - 2wf - 2wk - w + 1)$$
$$+ (3w^2 - 4wf - 4wk + 4wc_S - 2w - 2c_S f - 2c_S k + 2fk + c_S^2 + f^2 + k^2 - c_S + f + k + 1) = 0.$$

Solving the above equation gives the equilibrium prices as follows:

$$p_1^* = \frac{1}{3w^2}\left(3w^2 - 2w(f + k - c_S) - w + 1 - \sqrt{w^2\{(f + k - c_S)^2 + (f + k - c_S) + 4\} - 2w\{2(f + k - c_S) + 1\} + 1}\right)$$
$$p_2^* = \frac{1}{3}\left(1 - (f + k - c_S) - \frac{1}{w} + \frac{1}{w}\sqrt{w^2\{(f + k - c_S)^2 + (f + k - c_S) + 4\} - 2w\{2(f + k - c_S) + 1\} + 1}\right).$$

Case 2: Medium Switching Cost ($k < w < f + k$)

Assume that $k < w < f + k$. Then the open source software vendor charges price p_S such that $p_S < f + k - w$, which guarantees the demand from high-type customers no matter what prices

the commercial software vendor charges. Suppose that $c_S > f + k - w$. Then the open source software vendor cannot play a pricing strategy to attract the customers. Thus, the equilibrium is the same as in the high switching cost case. Now, suppose that $c_S < f + k - w$. Then the optimal price for open source support is $p_S^* = f + k - w$. After observing the open source support price, the commercial software vendor offers prices such that

$$wp_1 + p_2 < w + p_S^* - k \Leftrightarrow wp_1 + p_2 < f.$$

Such pricing strategy allows the commercial software vendor to serve the customers in the low-type segment. Note that the commercial software vendor's profit is

$$\pi_C = (1 - p_1)p_1 + (1 - \mu)(1 - p_1)(1 - p_2)p_2.$$

The optimization problem is then

$$\underset{p_1, p_2}{\text{Max}}\, \pi_C \quad \text{subject to } wp_1 + p_2 \le f.$$

The Lagrangean function can be defined as

$$L(p_1, p_2, \lambda) = (1 - p_1)p_1 + (1 - \mu)(1 - p_1)(1 - p_2)p_2 - \lambda(wp_1 + p_2 - f).$$

The Kuhn-Tucker conditions are

$$\frac{\partial L}{\partial p_1} = 1 - 2p_1 - (1 - \mu)(1 - p_2)p_2 - w\lambda = 0$$
$$\frac{\partial L}{\partial p_2} = (1 - \mu)(1 - p_1)(1 - 2p_2) - \lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = wp_1 + p_2 - f = 0.$$

From $\frac{\partial L}{\partial \lambda}$, we have $p_2 = f - wp_1$. From $\frac{\partial L}{\partial p_1}$ and $\frac{\partial L}{\partial p_2}$,

$$p_2^2 + p_2(2w - 2wp_1 - 1) + p_1\left(w - \frac{2}{1 - \mu}\right) + \frac{1}{1 - \mu} - w = 0..$$

Replacing p_2 in the above equation with $f - wp_1$ leads to

$$3w^2 p_1^2 + 2p_1(w - w^2 - 2wf - \frac{1}{1-\mu}) + (f^2 - f + 2wf + \frac{1}{1-\mu} - w) = 0.$$

Solving the above equation gives the equilibrium prices as follows:

$$p_1^* = \frac{1}{3w^2}\left(w^2 - w(1-2f) + \frac{1}{1-\mu} - \sqrt{\left(w^2 - w(1-2f) + \frac{1}{1-\mu}\right)^2 - 3w^2\left(f^2 - f - w(1-2f) + \frac{1}{1-\mu}\right)}\right)$$

$$p_2^* = f - \frac{1}{3w}\left(w^2 - w(1-2f) + \frac{1}{1-\mu} - \sqrt{\left(w^2 - w(1-2f) + \frac{1}{1-\mu}\right)^2 - 3w^2\left(f^2 - f - w(1-2f) + \frac{1}{1-\mu}\right)}\right).$$

Case 3: Low Switching Cost ($w < k$)

Suppose that the scale parameter for switching cost, i.e., maximum possible switching cost is lower than vendor independence benefit ($w < k$). Suppose that $c_S > f + k - w.$ Then it is trivial to show that the equilibrium is the same as in high switching cost case. Now, assume that $k - w < c_S < f + k - w.$ Then the open source software vendor can dominate the market for high-type customers by charging price at $p_S^* = f + k - w.$ Thus, the result in medium switching cost case holds. Finally, suppose that $c_S < k - w.$ In this case, the open source software vendor dominates the entire market in period 2 by charging price $p_S^* = k - w$. Since the commercial vendor observes the OSS price, it does not have any incentive to participate market in period 2. Consequently, the commercial software vendor maximizes its profit by charging monopoly price ($p_1^* = \frac{1}{2}$) in the first period. This completes proof of Proposition 7. ***QED.***

References

Bakos, Y., and Brynjolfsson, E. “Bundling Information Goods: Pricing, Profits and Efficiency”, *Management Science* 45(12), 1999, pp. 1613 – 1630.

Bhargava, H., and Choudhary, V. “Information Goods and Vertical Differentiation”, *Journal of Management Information Systems* 18(2), 2001, pp. 85 – 102.

Broersma, M. “Companies Buy Open Source Because It’s Better, Not Cheaper”, *Techworld*, April 21, 2005.

Casadesus-Masanell, R., and Ghemawat, P. “Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows”, *Management Science* 52(7), 2006, pp. 1072 – 1084.

Chen, P., and Hitt, L. “Measuring Switching Costs and Their Determinants in Internet Enabled Businesses: A Study of the Online Brokerage Industry,” *Information Systems Research* 13(3), 2002, pp. 255-276.

Chen, Y., and Png, I. “Information Goods Pricing and Copyright Enforcement: Welfare Analysis.”, *Information Systems Research* 14(1), 2003, pp. 107–123.

Choudhary, V., Ghose, A., Mukhopadhyay, T., and Rajan, U. “Personalized Pricing and Quality Differentiation”, *Management Science* 51(7), 2005, pp. 1120 – 1130.

Chuang, J., and Sirbu, M. “Optimal Bundling Strategy for Digital Information Goods: Network Delivery of Articles and Subscriptions”, *Information Economics and Policy* 11, 1999, pp. 147 – 176.

Clemons, E., Hann, I., and Hitt, L., “Price Dispersion and Differentiation in Online Travel: An Empirical Investigation”, *Management Science* 48(4), 2002, pp. 534 – 549.

Dewan, R., Jing, B., and Seidmann, A. “Product Customization and Price Competition on the Internet”, *Management Science* 49(8), 2003, pp. 1055 – 1070.

Dewan, S., and Mendelson, H. “User Delay Costs and Internal Pricing for a Service Facility.” *Management Science* 36(12), 1990, pp. 1502–1517.

Economides, N., and Katsamakas, E. “Two-Sided Competition of Proprietary vs. Open Implications for the Software Industry”, *Management Science* 52(7), 2006, pp. 1057 – 1071.

Economist “Microsoft at the Power Point”, September 11, 2003.

Fitzgerald, B., and Feller, J. “Open Source Software: Investigating the Software Engineering, Psychosocial and Economic Issues,” *Inform. Systems Journal* 11(4), 2001, pp. 273 – 276.

Flynn L., and Lohr, S. “2 Giants in a Deal over Linux”, *New York Times*, November 2, 2006.

Franke, N., and von Hippel, E. “Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software”, Research Policy 32, 2003, pp. 1199 – 1215.

Ghosh, R. A. “Interview with Linus Torvalds: What Motivated Free Software Developers?”, *First Monday* 3(3), 1998.

Ghosh, R. A., Glott, R., Krieger, B., and Robles, G. “Free/Libre and Open Source Software: Survey and Study”, International Institute of Infonomics, University of Maastricht, The Netherlands, Retrieved March 30, 2002.

Gloude, M. “Open Source Usage Is Up, But Concerns Linger”, *Forrester Research*, June 23, 2005.

Gomulkiewicz, R. W. “Entrepreneurial Open Source Software Hackers: MySQL and Its Dual Licensing”, C*omputer Law Review and Technology Journal* (9), 2004, pp. 203 – 212.

Gupta A., Staul, D., and Whinston A. “A Stochastic Equilibrium Model of Internet Pricing”, *Journal of Economic Dynamics and Control* 21, 1997, pp.697 – 722.

Hann, I., Roberts, J., and Slaughter, S., and Fielding, R. “An Empirical Analysis of Economic Returns to Open Source Participation”, Working Paper 2006-E5, Tepper School of Business,

Carnegie Mellon University, Pittsburgh, PA, and Marshal School of Business, University of Southern California, Los Angeles, CA.

Hertel, G., Niedner S., and Herrmann S. "Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel", *Research Policy* 32(7), 2003, pp. 1159 – 1177.

Hitt, L., and Chen, P. "Bundling with Customer Self-Selection: A Simple Approach to Bundling Low Marginal Cost Goods", M*anagement Science* 51(10), 2005, pp. 1481 – 1493.

Kerner S. M. "Will Oracle Linux Kill Red Hat?", *internetnews.com*, October 26, 2006.

Kingstone, S. "Brazil Adopts Open Source Software", June 2, 2005.

Learner, J., and Tirole, J. "The Open Source Movement: Key Research Questions", *European Economic Review* 45 (4 – 6), 2001, pp. 819 – 826.

Lerner, J. and J. Tirole "Some Simple Economics of Open Source", *Journal of Industrial Economics* 50(2), 2002, pp. 197 - 234.

Mackie-Mason J., and Varian H. "Pricing Congestible Network Resources", *IEEE Journal of Selected Areas in Communications* 13(7), 1995, pp. 1141-1149.

McCarthy, V. "Survey: Financial IT Execs Say Jury Still Out Open Source", *OpenEnterpriseTrends.com*, March 24, 2006.

Mendelson, H., and Whang, S. "Optimal Incentive-Compatible Priority Pricing for the *M/M/*1 Queue." *Operations Research* 38(5), 1990, pp. 870–883.

Moczar, L. "The Economics of Commercial Open Source", *Galatea Information Strategies*, 2005.

Raghunathan, S., Prasad, A., Mishra, B., and Chang, H. “Open Source Versus Closed Source: Software Quality in Monopoly and Competitive Markets”, *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* 35(6), 2005, pp. 903 – 918.

Raymond, E. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O’Reilly & Associates, Sebastopol, CA, 1999.

Reid R. “Forrester: More Firms Using Open Source”, *ITWorldCanada.com*, March 17, 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), 2006, pp. 984 – 999.

Rossi, M. A. “Decoding the Free/Open Source Puzzle. A Survey of Theoretical and Empirical Contributions”, Working Paper, Dipartimento Di Economia Politica**,** Università degli Studi di Siena., Siena, Italy.

Shah, S. “Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development”, *Management Science* 52(7), 2006, pp. 1000 – 1014.

Shores, R. “Oracle Announced the Same Enterprise Class Support for Linux as for Its Database”, Oracle Press Release, October 25, 2006.

Sundararajan, A. “Nonlinear Pricing of Information Goods.”, *Management Science* 50(12), 2004, pp. 1660-1673.

Välimäki, M. “Dual Licensing in Open Source Software Industry”, *Systemes d´Information et Management* 8(1), 2003, pp. 63-75.

Varian, H., and Shapiro, C. “Linux Adoption in the Public Sector: An Economic Analysis”, *Working Paper*, 2003.

Vaughan-Nichols S. J. “Making Money from Free Software”, *eWeek*, July 13, 2005.

von Hippel, E., and von Krogh, G. “Open Source Software and the “Private-Collective” Innovation Model: Issues for Organization Science”, *Organization Science* 32(2), 2003, pp. 209 – 233.

Westland, C. “Congestion and Network Externalities in the Short Run Pricing of Information Systems Services.”, *Management Science* 38(7), 1992, pp. 992–1099.

Wichmann, T. “Free/Libre and Open Source Software: Survey and Study”, *Berlecon Research GmbH*, Berlin, Germany, 2002.