NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was scanned as received.

x - xii

This reproduction is the best copy available.

UMI[®]

Open Source Software Projects, Market Offers, and Competitive Advantage

by

Richard Alam

A thesis submitted to the Faculty of Graduate Studies and Research in
partial fulfillment of the requirements for the degree of Master of
Engineering in Telecommunications Technology Management

Department of Systems and Computer Engineering

Carleton University

Ottawa, Canada K1S 5B6

January 2006

© Copyright 2006 Richard Alam

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-16483-9

Our file Notre référence

ISBN: 978-0-494-16483-9

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

This thesis uses a grounded theory approach to study the relationship between the architecture of a software telecommunications product, the structure of its development organization, and the market for the product during the severe downturn in the telecommunications industry during 2001 and 2002. The research was motivated by Conway's Law which maintains that the architecture of a product mirrors the organization that builds it.

Although the development organization for the product studied was severely downsized during the study period, its market remained very robust. This meant that the development organization used the architecture to develop many new product features with fewer resources. The research premise was that as the development organization shrank and new features were delivered, the product architecture would degrade correspondingly. Detailed data was collected on the product architecture, the development organization and the market.

Contrary to expectations, the product architecture did not degrade during the study period. On the other hand, post deployment error rates did increase. The major conclusion of the research is that unlike error creation and elimination, software architecture integrity is not a manpower issue. In small,

well-integrated development organizations, software architecture need not degrade even when those organizations are severely stressed by having to do more with less.

Based on this research result, several managerial recommendations are made on how to maintain software product architecture integrity during organizational stress.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	xi
LIST OF TABLES.....	xii
1 INTRODUCTION	1
1.1 Deliverables.....	2
1.2 Contribution	3
1.3 Relevance	4
1.4 Organization	5
2 LITERATURE REVIEW.....	6
2.1 Review of OSS Literature	6
2.1.1 OSS Project.....	6
2.1.2 Motivations of Individuals to Participate in OSS Projects	11
2.1.3 Motivations of Companies to Participate in OSS Projects	12
2.2 Market Offer	19
2.3 Complementary Product Integration.....	23
2.4 Competitive Environment	25
2.5 Commercialization Framework.....	29
2.6 Strategic Alliances, Freeware, and Standards.....	33
2.6.1 Strategic Alliances.....	33
2.6.2 Freeware	35
2.6.3 Standards.....	36
2.7 Lessons Learned from the Literature Review	37
3 RESEARCH METHOD	41
3.1 Unit of Analysis, Study Period, and Sample Selection.....	42

3.2	Data Collection	44
3.3	Data Analysis	45
3.3.1	How OSS is Integrated into Market Offers	45
3.3.2	Insights using Teece's Commercialization Framework.....	47
3.3.3	How OSS Improve Competitive Environment.....	48
3.3.4	Benefits and Risks of Establishing OSS Projects	48
3.3.5	Benchmark against Alliance, Freeware, and Standards.....	49
4	RESULTS.....	51
4.1	Companies in the Sample	51
4.2	Types of Market Offers	57
4.3	Ways OSS is Integrated into Market Offers.....	61
4.3.1	Minimal-No	62
4.3.2	Minimal-Yes.....	63
4.3.3	Comprehensive-No	68
4.3.4	Comprehensive-Yes.....	69
4.4	Insights Using Commercialization Framework.....	72
4.5	How OSS Improves the Companies Competitive Environment	75
4.5.1	Factor Conditions	75
4.5.2	Demand Conditions.....	77
4.5.3	Context for Strategy and Rivalry.....	79
4.5.4	Related and Supporting Industries	81
4.6	Benefits and Risks of Establishing OSS Project.....	81
4.6.1	Benefits	82
4.6.2	Risks	83
4.7	Benchmark against Strategic Alliance, Freeware, and Standards...84	
4.7.1	Strategic Alliances.....	84
4.7.2	Freeware	86
4.7.3	Standards.....	87

5	DISCUSSION OF RESULTS.....	88
5.1	Key Findings.....	88
5.1.1	OSS Based Solutions.....	88
5.1.2	OSS as an Alternative to Buy or Build.....	89
5.1.3	Lower Cost Solution to Achieve Strategic Objectives.....	90
5.1.4	Companies control the OSS projects and influence other projects	91
5.2	Issues with Teece's Commercialization Framework.....	91
6	CONCLUSIONS, LIMITATIONS, AND FUTURE RESEARCH	96
6.1	Conclusions.....	96
6.2	Limitations	98
6.3	Opportunities for Future Research	99
7	REFERENCES	111
APPENDIX	121	
	Appendix A List of Companies	121
	Appendix B How OSS is Integrated with Market Offers	125
	Appendix C Insights Using Commercialization Frameworks	133
	Appendix D How OSS Improves Competitive Environment	150
	Appendix E Benefits and Risks of Establishing OSS Projects.....	153

LIST OF FIGURES

Figure 1 Five Levels of a Product	21
Figure 2 Types of Complementary Product Integration	25
Figure 3 Diamond of Competitive Advantage (Porter, 1990)	27
Figure 4 Commercialization Framework	30
Figure 5 OSS Integration and Contribution to Market Offer	63
Figure 6 Pictorial Representation of OSS Integration with Market Offer	64

LIST OF TABLES

Table 1 Comparison of Open Source Licenses	9
Table 2 Existing Literature on License Models of OSS.....	10
Table 3 Motivations to Participate in OSS Projects.....	13
Table 4 Literature on Why Companies Participate in OSS Projects	18
Table 5 The Five Levels of a Product	22
Table 6 Companies and OSS Projects in the Sample	53
Table 7 Types of Market Offers	59
Table 8 Insights using Commercialization Frameworks	73
Table 9 How OSS Improves the Companies OSS Environment.....	76
Table 10 Companies Use OSS to Achieve Strategic Objectives	90
Table 11 Observations not Supported by Teece's Commercialization Framework.....	92

1 INTRODUCTION

This research uses data on twelve focal companies to identify how open source software (OSS) projects contribute to the companies' market offers and change their competitive environments.

Open source software (OSS) has attracted significant attention in recent years. Large, well known companies have legitimized the use of OSS for competitive purposes. Companies such as IBM, Red Hat, Novell, HP and others have invested significant resources in OSS. Linux, perhaps the most successful OSS project, has generated more than \$1 billion in annual revenues for both IBM and HP (Koenig, 2004). Novell, with a \$50 million investment from IBM, bought the open source distributor SuSE for \$210M.

Several studies are available on the ways companies use to profit from OSS (Raymond, 1999a; Koenig, 2004; Valimaki, 2003; Dahlander, 2004; Lerner and Tirole, 2000; Pal and Madanmohan, 2002) and the different ways companies use OSS to implement their strategies (Raymond, 1999a; Bonaccorsi and Rossi, 2003; Henkel, 2003; West and Gallagher, 2004).

What companies that establish OSS projects do with the software that is produced has not been examined in the literature. Research on how companies that establish open source software projects use the software in their market offers is lacking.

This thesis answers two research questions:

- how do companies integrate the software from the OSS projects that they establish into their market offers?
- how do companies use OSS to change their competitive environments?

For the purpose of this research, an OSS project refers to a project used to develop and release software under an open source license (Von Krogh and Von Hippel, 2003). A market offer refers to the generic product or service and the set of augmenting services, programs, and systems that a firm offers potential customers (Anderson et al., 2001). The competitive environment of a company includes: (i) factor conditions; (ii) demand conditions; (iii) context for strategy and rivalry; and (iv) related and supporting industries (Porter, 1990).

1.1 Deliverables

This research delivers six outcomes:

1. The types of market offers that use the software from the OSS projects that the companies have established.
2. The ways OSS is integrated into the market offers.
3. The insights gained from using the commercialization framework advanced by Teece (1986) to examine an OSS as a complementary asset.

4. The different ways OSS improves the companies' factor conditions, demand conditions, context for strategy and rivalry, and related and supporting industries.
5. The benefits and risks of establishing OSS projects.
6. How OSS differs from strategic alliance, freeware, and standards.

1.2 Contribution

Von Krogh and Von Hippel (2003) have organized the OSS literature into three streams: (i) studies on the motivation of programmers to contribute to OSS projects, (ii) studies on the process by which OSS is developed, and (iii) studies on the competitive dynamics of OSS. This research contributes to the third stream.

This research makes at least three contributions. First, it provides concrete examples of how companies that established OSS projects use the software to create market offers and change their competitive environments.

The second contribution this research makes is that it identifies what a company can do with OSS that it cannot do with strategic alliances, freeware, and standards.

The third contribution of this research is the insights gained from using Teece's framework (Teece, 1986) to examine the special case where companies can create complementary assets as public goods.

1.3 Relevance

There are at least three groups who may be interested in the contributions of this research. First, researchers in strategic management may be interested in how OSS projects change companies' competitive environments and are used to acquire external technology, develop markets, and establish standards.

Top management teams of companies that have not established OSS projects may also be interested in this research. They may be interested in the benefits and risks of setting up OSS projects.

The third group that may be interested in this research is comprised of the entrepreneurs who wish to create new businesses anchored around OSS. These entrepreneurs wish to assess how companies use OSS to create value.

1.4 Organization

This thesis is organized into six chapters. Chapter 1 is the introduction. Chapter 2 reviews the relevant literature. Chapter 3 presents the method used to undertake this research. Chapter 4 presents the results. Chapter 5 discusses the results.

Finally, Chapter 6 provides the conclusions, describes the limitations of the study, and identifies opportunities for future research.

2 LITERATURE REVIEW

This section reviews the literature. Section 2.1 reviews the different literature on OSS. Section 2.2 reviews the literature on market offers. Section 2.3 explains complementary product integration. Section 2.4 presents a literature on competitive environment while Section 2.5 reviews the literature on commercialization frameworks. Section 2.6 reviews the literature on strategic alliances, freeware, and standard. The last section, Section 2.7, presents the lessons learned from the literature review.

2.1 Review of OSS Literature

2.1.1 OSS Project

An OSS is governed by a license that permits anybody to access the source code, modify the software, and redistribute the software whether modified or unmodified (Open Source Initiative, 2005). OSS projects develop and release the software under an open source license (Von Krogh and Von Hippel, 2003).

OSS projects can be classified into community-founded and spun-out projects (West and O'Mahony, 2005). Community-founded projects are started by individuals or a small group of programmers (Raymond, 1999b). Founders of some community-founded projects start companies to profit from the project. Spun-out projects have software that initially started as proprietary inside the company but then released as open source software. Some companies continue to support the project by continuously providing resources.

Projects implement mechanisms to protect their work by enforcing copyrights, trademarks, and the license of the software (O'Mahony, 2003). Projects can choose from a number of approved open source licenses or create their own as long as it conforms to the Open Source Definition (Open Source Initiative, 2005). Lerner and Tirole (2002) classify open source licenses into three classes: highly restrictive, restrictive, and unrestrictive. There are a number of approved open source licenses (Open Source Initiative, 2005). However, three are most commonly used: GPL, LGPL, and BSD. The General Public License is an example of a highly restrictive; the GNU Lesser General Public License (LGPL) is an example of a restrictive license; while the Berkeley Software Distribution (BSD) license is considered an unrestrictive license.

The GPL is probably the most widely used license (Pearson, 2000; Bonaccorsi and Rossi, 2004). GPL uses copyright law to keep the software 'free'. Modified GPL software must be distributed with source code and available to the public granting others to use the modified software. The license causes any work that contains or derives from a GPL'd software must be released under the same license. This provision led others to call GPL as 'viral' and that mere interaction with GPL'd software will cause proprietary software to publish the source code. This is not the case. If the program is separate and independent from the GPL'd software, the license does not apply. Only in cases where the code is compiled

or statically linked with GPL'd software that the derived software has to be released as open source.

The LGPL is typically used to license a software "library" – a collection of pre-written software routines that other programs can use (Evans, Reddy, and Layne-Farrar, 2001). LGPL was developed to get around the 'viral' nature of GPL. If the software libraries were licensed as GPL, these libraries will 'infect' the software that uses them. Thus, these libraries would not be used by anyone wanting to develop proprietary software (Pearson, 2000). The LGPL allows greater flexibility with regards to allowing programs to link or use LGPL'd software without requiring the incorporating program to be open sourced (Lerner and Tirole, 2000).

The BSD license has very few provisions. Unlike GPL, the license permits the modified software to be licensed as proprietary when redistributed (Pearson, 2000). The software (whether modified or not) must retain the copyright notices in the program code but faces no other significant restrictions whenever it is redistributed. The BSD is the most widely used non-restrictive license because it allows anybody to do anything with the code (Bonaccorsi and Rossi, 2004).

Table 1 shows a comparison of some open source licenses (Nissila, 2002). The table compares the licenses based on permission to mix with non-free software,

take the modified software proprietary, re-license the software, and special privileges for original owners over modifications to the software.

There is a lack of empirical research on the use of open source licenses. Some studies have been done on open source projects choice of licenses (Lerner and Tirole, 2002; O'Mahony, 2003). They found that most projects use restrictive and highly restrictive licenses protecting the future of the project by taking away the incentive for others to take the software private.

Table 1 Comparison of Open Source Licenses				
Licenses	Can be mixed with non-free software	Modifications can be taken private and not returned to author	Can be re-licensed by anyone	Contains special privileges for the original copyright holder over user's modifications
Generalized Public License (GPL)	No	No	No	No
GNU Library General Purpose License	Yes	No	No	No
Berkeley System Distribution	Yes	Yes	No	No
Netscape Public License	Yes	Yes	No	Yes
Mozilla Public License	Yes	Yes	No	No
Public Domain	Yes	Yes	Yes	No

Source: Nissila (2002)

Valimaki and Oksanen (2005) examined the impact of licensing on the operating system market. They looked at the different licensing policies by Microsoft Windows, Apple's Mac OS X, GNU/Linux. Microsoft uses a proprietary license policy while Apple uses an open source license to its kernel and GNU/Linux uses an open source license for the whole system. They concluded that Apple and GNU/Linux has managed to gain stronger positions over Windows than previous years because of their adoption of open source licenses.

Bonaccorsi and Rossi (2004) investigated 146 companies that participate in OSS projects. They were interested to determine the different licenses of the open source software these companies' used. They found out that most companies use software from projects that issue GPL and LGPL licenses.

Table 2 presents a summary of literature on open source projects and companies choice of licenses.

Table 2 Existing Literature on License Models of OSS

• Impact of open source licenses on the operating systems market (Valimaki and Oksanen, 2005) • Projects choice of licenses (Lerner and Tirole, 2000) • Company's choice of licenses (Rossi and Bonaccorsi, 2004)

The success of an OSS project depends on two factors: developer motivation and user utility (Stewart, Ammeter, and Maruping, 2005). The project must be

able to attract motivated developers to create a vibrant community. One way to attract developers is to convince potential users about the usefulness of the software. This is important because all the developers are users of the software (Gacek and Arief, 2004). The developers must be convinced that they can get value out of participating in the project.

2.1.2 Motivations of Individuals to Participate in OSS Projects

Individual developers are motivated to participate for several reasons. They participate to solve their own programming problems – scratch an itch (Raymond, 1999b) or satisfy their own needs (Franke and von Hippel, 2003). While most of these volunteers do not get paid for their contributions, others are paid (Hars and Ou, 2002; Hertel, Niedner, and Herrmann, 2003; Lakhani et al., 2002). Other developers participate as a “signaling” method (Lerner and Tirole, 2001), that is, to gain reputation among peers and to showcase their skills for future employment opportunities. Some contributors participate for the learning opportunity and to sharpen their programming skills (Lakhani and Von Hippel, 2003).

In OSS projects, developers are also users (Raymond, 1999a; Gacek and Arief, 2004). They are involved not only in development of the software but also in testing and providing support to other users (Lakhani and von Hippel, 2003). As the project becomes popular and more users become involved, problems are quickly fixed (Raymond, 1999a). Since users have access to the source code, it

is easy to detect and fix bugs that they encounter (Franck and Jungwirth, 2003). Krishnamurthy (2002), though, found that most projects have limited number of developers and that only a small group of developers tend to contribute the most to the project.

Developers usually start out contributing by fixing bugs or contributing modules into the project (von Krogh, Spaeth, and Lakhani, 2003). They then become maintainers of those modules and then move on to be part of the core group that sets the direction of the project. These developers coordinate and communicate with each other through e-mail, newsgroups, instant messaging, and in the source code management system (von Hippel and von Krogh, 2003).

2.1.3 Motivations of Companies to Participate in OSS Projects

Some projects not only attract individual contributors but companies as well. But companies have different motivations from individual contributors. Companies participate for profit reasons. Companies want to better know the competition, incorporate open source ideas into their products, and hire talented programmers (Lerner and Tirole, 2001).

Bonnaccorsi and Rossi (2004) have classified the motivations, based on previous literature, of individuals and companies into economic, social, and technological areas as illustrated in Table 3.

Companies invest significant amount of resources in open source. IBM claims to have spent \$1 billion on Linux alone and is also active in several open source projects (Wilcox, 2000). Other companies also devote considerable resources to the development of open source software (Wichmann, 2002).

However, companies face some dilemmas by participating in open source. They lose intellectual property (IP) rights to their works and loss of revenue from licenses (Henkel, 2003). Companies have to continuously invest resources to maintain the software and to make sure that the code released does not infringe on IP of others. These companies have to make sure that the benefits outweigh the downside of participating in open source.

Table 3 Motivations to Participate in OSS Projects		
Motivation Area	Individuals	Companies
Economic	<ul style="list-style-type: none"> • Monetary rewards • Low opportunity costs • Gaining reputation among peers • Gaining future career benefits 	<ul style="list-style-type: none"> • Being independent of the price and license policies of the large software companies • Addressing the new model of software as a consumer-driven service (making money on complementary services) • Obtaining indirect revenues by selling related products • Affording innovation (by exploiting the R&D activity of the Open Source community) • Hiring good IT specialists
Social	<ul style="list-style-type: none"> • Fun to program (Loving to 	<ul style="list-style-type: none"> • Conforming to the values of the

	code) <ul style="list-style-type: none"> • Altruism (gift economy) • Sense of belonging to the community • Fight against proprietary software 	Open Source community (in order not to betray the trust of the Open Source developers) <ul style="list-style-type: none"> • Sharing code and knowledge with the community (reciprocating in order to sustain cooperation) • Thinking that software should not be a proprietary good (in order to reduce market power of large companies)
Technological	<ul style="list-style-type: none"> • Learning • Contributions and feedback from the community • Working with bleeding-edge technology • Scratching a personal itch 	<ul style="list-style-type: none"> • Exploiting feedback and contributions from developers of the Open Source community for cutting development costs and improve the software • Exploiting feedback and contributions from the user community for testing and improving software • Cutting hardware costs • Promoting standardization • Addressing security issues
Source: Bonnaccorsi and Rossi (2004)		

Companies lose revenue from software licenses when they participate in open source. Software can be thought of as having two parts: product and service. The product is the license sold to use the software while the service refers to the maintenance, consulting, training, and management of the software (Berlecon Research, 2002). Traditionally, companies make money from software by selling the license and the service. However, in OSS, the software is available to anyone for free as long as they abide by the license. But, some customers cannot or do

not want to abide by the open source license that they are willing to pay for a commercial license. Companies that have undisputed rights to the software are able to charge license fees (Valimaki, 2003; Dahlander, 2004; van Wendel de Joode, R., de Bruijn, J.A., and van Eeten, M.J.G., 2002). These companies institute two types of licenses to their software: an open source and commercial license.

Unfortunately, most companies do not own undisputed rights to the software. They have to come up with novel ways of profiting from OSS. Companies try to make money from providing complementary products and services (van Wendel de Joode et al., 2002; Raymond 1999a; Dahlander, 2004; Lerner and Tirole, 2000) not efficiently supplied by the open source community.

Companies add value by assembling and testing the software rigorously then selling support contracts, warranty, and integration services (Raymond, 1999b; Pal and Madanmohan, 2003). They build expertise on the open source software and charge customers for customization services tailoring the software to the customer's specific needs (van Wendel de Joode et al., 2002). Another way companies make money is through subscription services. Companies get maintenance, upgrades, and updates to the software as long as customers are subscribed to the service (Koenig, 2004).

Due to the chaotic nature of development of open source, other companies buffer their customers by creating delivery schedules that coincides with the customer's timelines (van Wendel de Joode et al., 2002). These companies make sure that the version the customers use is stable and well-tested. The company takes a snapshot of the development version maintained by the project. The company tests and fixes the problems found on this snapshot. The tested stable version plus add-on services is then offered to customers for a fee while the fixes are contributed back to the project.

Some companies participate in OSS projects for strategic reasons. Companies participate to gain a market position in a complementary segment, jumpstart an infant market, or break into an existing market with entrenched proprietary software (Raymond, 1999b; Bonaccorsi and Rossi, 2003). Bonnacorsi and Rossi (2003) found that Linux systems have managed to diffuse in environments dominated by Microsoft's Windows NT.

Another reason is to establish a standard or dominant design to preempt rivals from establishing a competing standard and dominant design (Raymond, 1999c; Henkel, 2003; Dahlander, 2004). Koenig (2004) provided IBM as an example of adopting the Apache web server while dropping its own web server to prevent Microsoft from monopolizing the web server market.

Other companies participate to commoditize complementary products (Koenig, 2004) and downstream suppliers (Raymond, 1999c). IBM is an example of supporting Linux to commoditize the x86-operating system, creating an opportunity by offering value higher up the software stack. The company makes money by offering clustering, availability, provisioning, and management software (Koenig, 2004).

Another reason companies participate is to lower their cost by taking advantage of pooled R&D especially for non-essential software (West and Gallagher, 2004). West and Gallagher (2004) gave examples such as Linux and Mozilla as software that benefited from pooled R&D. Companies donate R&D to the open source project while exploiting the skills of all contributors and for these companies to sell related products.

Table 4 presents a summary of literature on why companies participate in open source.

Table 4 Literature on Why Companies Participate in OSS Projects

- Make money through:
 - dual license (Valimaki, 2003; van Wendel de Joode et al., 2002; Dahlander, 2004)
 - selling of complements (van Wendel de Joode et al., 2002; Raymond, 1999; Dahlander, 2004; Lerner and Tirole, 2000)
 - providing consulting service, support, and warranty (Pal and Madanmohan, 2002; Raymond, 1999a; van Wendel de Joode et al., 2002; Koenig, 2004; Dahlander, 2004)
- Jumpstart an infant market, grow market, or enter market with entrenched proprietary players (Raymond, 1999b; Bonaccorsi and Rossi, 2003)
- Establish as standard or preempt a standard (Raymond, 1999c; Henkel, 2003; Dahlander, 2004)
- Commoditize complements and value chain (Raymond, 1999c; Koenig, 2004)
- Lower cost through pooled R&D (West and Gallagher, 2004)

A study by Dahlander and Magnusson (2005) examined Nordic OSS companies and came up with three approaches how companies participate in OSS projects. They argue that companies participate in a symbiotic, commensalistic, and parasitic approach to the project. In the symbiotic approach, the company is directly involved in the development of the project. They are invested in the project to thrive. The company gains when the project gains. The company gives something to the project, obey licenses, and respects norms and values of the project.

Commensalistic approach, on the other hand, the company gains from the project and the project does not expect anything in return from the company. The goal of the company is to co-exist with the project without causing it any harm.

Henkel (2004), in his study of embedded Linux companies, identified how firms protect their derived works by using the commensalistic approach. First, they only reveal code if their customers ask for it. Second, they delay and restrict the distribution of source code by demand only. Third, they delay the submission of code changes to the OSS project. Fourth, they make drivers as loadable modules so as not to be covered by the OSS license.

The parasitic approach is one where the company gains while the project loses. The company focuses on its own benefits without taking into account that its actions may harm the project. Some companies are not “good citizens” (Henkel, 2003) of the open source community. Rossi and Bonaccorsi (2004) in their survey of 146 companies on the motivation of participating in open source software found that the companies are merely code takers and that they give little importance to give back code to the community.

2.2 Market Offer

A product or market offer is anything that is offered to the market for attention, acquisition, use, or consumption that might satisfy a want or need (Kotler and Turner, 1995). The value of the market offer is proportion to the perceived ability to solve a problem or meet a need (Levitt, 1980).

A company that creates a product has to provide add-ons to the product in terms of services, programs, and systems to make it attractive (Anderson et al., 2001).

These add-ons might be in the form of packaging, customer services, advertising, financing, and other things that customers' value (Kotler and Turner, 1995). A market offer is the whole package of product and add-ons that creates value to the customer. Companies use privilege assets (Baghai, Coley, and White, 2000) to create attractive market offers. Examples of privilege assets are distribution networks, brands, reputations, customer information, infrastructure, and intellectual property.

To better understand market offers, it is best to examine the five levels of a product: core benefit, generic product, expected product, augmented product, and potential product (Kotler and Turner, 1995). Figure 1 show a diagram of the five product levels. The core benefit is the fundamental service or benefit that the customer buys. It is the fundamental performance of the product that solves a customer's basic problem (Anderson, et al., 2001).

The core benefit needs to be converted into a generic product. The generic product is the basic version of the product and is the minimum required for the company to sell to the market (Levitt, 1980). The generic product has no differentiation with the competitor's products.

Figure 1 Five Levels of a Product

Source: Kotler and Turner (1995)

To sell a product, it has to meet some minimal purchase conditions expected by the customer. The expected product has the attributes and conditions that customers expect and agree to when they buy the product (Kotler and Turner, 1995). Different customers have different expectations. Companies differentiate from competitors on which expectations they want to meet (Levitt, 1980).

The augmented product includes additional benefits that exceed customer expectations (Levitt, 1980; Kotler, 1995). The augmented product is what sets companies apart from competitors trying to meet similar customer expectations (Kotler, 1995). The augmented product meets a broader set of customer requirements and preferences to add value to what the customer is trying to accomplish (Anderson et al., 2001). Unfortunately, augmentation cost the

company money and as the product matures, these augmentations become part of the expected product (Kotler, 1995).

The potential product is everything that the product might undergo in the future. It is what holds and attract potential customers (Levitt, 1980). Some customers purchase products based on what the transformations the product might go through in the future (Kotler, 1995). As soon as these transformations are realized, they become an augmented product (Anderson et al., 2001).

Companies must be aware of these five product level because the competition has shifted from what comes out of the factories (generic product) to what attributes and benefits a company provides to meet and exceed customer expectations (Kotler, 1995). Companies mostly compete on the expected and augmented products. When companies compete in the generic product, they do so mostly through price.

Table 5 provides a summary of the five levels of a product.

Table 5 The Five Levels of a Product	
Core benefit	Fundamental service or benefit the customer buys
Generic product	Basic version of product
Expected product	Attributes and conditions the buyer normally expects
Augmented product	Distinguishes from others
Potential product	Augmentations and transformations that might undergo in the future

2.3 Complementary Product Integration

Nambisan (2002) investigated the relationship of complementary product integration and initial technology strategy of high-technology new ventures. A new venture's success is dependent on its ability to integrate its products to relevant complementary products as on the core functionality itself. Iansiti and West (1997) argued that competitive advantage now often goes to companies that are most adept at choosing among the vast number of technological options then integrate them and not necessarily to the companies that create them.

A complementary product is one that enhances the value of the core product when the two are used together (Sengupta, 1998). A complementary product is different from components or supplies of a product. A product cannot function without the components and supplies. A complementary product adds value to a fully-functional product.

There are four ways a company can develop complementary products. Companies can form co-development alliances, allow others access to the proprietary interface of their product, form co-marketing alliances, or OEM agreements from suppliers (Sengupta, 1998).

Nambisan (2002) identified three main type of complementary product integration based on the nature of coupling and the extent of coupling. Nature of coupling means whether the integration is achieved outside or inside the core product.

The extent of coupling means whether or not the integration is comprehensive and results in additional functionalities seen by the end-users.

Figure 2 show the three types of integration: value-added internal, add-on module, and data interface integration.

In value-added internal (VI) integration, the core product is *comprehensively internally integrated* with the complementary product in a seamless fashion. The result is the addition of features not available before integrating the two products. In add-on module (AM) integration, the core product is *comprehensively externally integrated* with the complementary product. The external integration is made through an external module or component separate from the core product. The external module facilitates sharing of data and functions between the two products resulting in additional features not available before integration. In data interface (DI) integration, the core product is *minimally externally integrated* with the complementary product. The resulting integration adds minimal functionality. This type of integration is usually used to facilitate transfer of data across the two products by defining a technical interface.

2.4 Competitive Environment

Porter's book *The Competitive Advantage of Nations* presented a method of analyzing how nations influence the competitive performance of companies in the international market (Grant, 1991). The nation is the "home base" that provides the competitive environment for companies to develop identity, character, and abilities to prepare the companies to successfully compete internationally.

The company's competitive environment is made up of four determinants that interact together in a diamond to create forces that provides pressures, incentives, and capabilities for firms to improve and innovate (Yetton, Craig, Davis, and Hilmer, 1992). The four determinants are: factor conditions, demand conditions, context for strategy and rivalry, and related and supporting industries.

They form the Diamond of Competitive Advantage (Porter, 1990) shown in Figure 3.

Factor conditions are, either, basic or advanced, generalized or specialized (Yetton et al., 1992). Examples of basic and generalized factors are natural resources, geographical location, or unskilled labor (Grant, 1991). Examples of advanced and specialized factors are skilled labor, capital, physical infrastructure such as roads or telecommunications, and administrative infrastructure (Porter and Kramer, 2002). Advanced factors are the result of investments by individuals, companies, and the government (Grant, 1991). The key to success is the rate at which these advanced and specialized factors are created and upgraded as well as the level of efficiency with which they are deployed (Hannah and Williamson, 1990).

Demand conditions refer to the presence of sophisticated and demanding customers, local market segments that are also present globally, and whose needs are the same as those outside the local markets (Porter and Kramer, 2002). These sophisticated customers provide companies with clearer and earlier insights into emerging customer needs. Companies become sensitive to their customers forcing them to respond to tough challenges to satisfy their customers. Demanding customers create pressure for companies to innovate, develop higher quality products quickly, and create better marketing.

Figure 3 Diamond of Competitive Advantage (Porter, 1990)

Context for strategy and rivalry refers to the presence of local policies and incentives, such as IP protection, that encourage investment and sustained upgrading (Porter and Kramer, 2002) of resources. It also refers to the presence of vigorous local competition. Nations must setup policies that encourage investments, protect intellectual property, open local market to trade, break-up or prevent monopolies, and reduce corruption. Rivalries create pressure for companies to cut costs, improve quality, and innovate (Grant, 1991). Presence of local rivals is a powerful stimulus to the creation and persistence of competitive advantage. "The more localized the rivalry, the more intense; and the more intense, the better" (Porter, 1990).

Related and supporting industries refer to the presence of capable and locally based suppliers and companies in related fields (Porter and Kramer, 2001).

Suppliers and end-users located near each other can take advantage of short lines of communication, quick and constant flow of information, and an ongoing exchange of ideas and innovation. Companies have the opportunity to influence their suppliers' technical efforts and can serve as test sites for R&D work, accelerating the pace of innovation (Porter, 1990). Also, an industry's investment in specialized and advanced factors is likely to have spillover benefits beyond the confines of that industry (Grant, 1991).

There are two forces that act on the firms: pressure and proximity (Yetton et al., 1992). Pressure is created by demanding customers and the presence of rivals. It is the key driver of innovation, and therefore, performance. Proximity, having demanding customers and tough rivals close by, further intensifies pressure on firms. Proximity also accelerates diffusion of innovation, facilitates investment in skills, and encourages the development of supporting industries (Grant, 1991). Proximity enhances responsiveness, exchange of information, and innovation, in addition to lowering transportation and inventory costs (Porter and Kramer, 2001).

Companies need to develop strategies in order to improve their Diamond of Competitive Advantage. Porter and Kramer (2001) argue that companies can use philanthropy as a way to improve their competitive environment.

2.5 Commercialization Framework

Existing commercialization frameworks (Teece, 1986; Gans and Stern, 2003) emphasize that to profit from innovation, a company must keep their knowledge secret or protect the innovation by legal means and must have access or control to key specialized complementary assets required to commercialize the innovation. Commercialization framework can best be explained through the diagram presented in Figure 4.

A firm has to examine three things in order to commercialize the innovation: (i) regimes of appropriability (including the nature of the technology and the efficacy of the legal protection mechanisms), (ii) dominant design, and (iii) access to complementary assets.

An appropriability regime involves the firm's ability to protect its innovation through legal mechanisms (i.e. IP protection) and the nature of the technology. Intellectual Property (IP) can be protected through patents, copyrights, and trade secrets. However, patents can be "invented around" and the cost of proving infringement is high. Trademarks are copyrighted words, symbols, and others used as the basis for brand identification. Strong brands give companies advantage over other imitations because of the brand's influence on the customers. Trade secrets provide protection when underlying technology or

processes are kept hidden even when the product becomes available to the public.

Figure 4 Commercialization Framework

The nature of technology can also contribute to the appropriability of an innovation. The product or the process involved might be difficult to imitate. The knowledge (tacit or codified) can also affect the degree of imitation. Codified knowledge can be easily transmitted and exposed. Tacit knowledge, however, is difficult to articulate and transfer. This requires those who possess the knowledge to demonstrate it to others.

When appropriability regime is weak, imitators can easily copy an innovation reducing the firm's ability to profit in the market. If appropriability regime is strong,

the firm has a high probability of profiting from a market offer enabled by the innovation or by directly licensing the innovation to others.

However, most innovations require complementary assets for commercialization. Complementary assets refer to diverse competence and resources needed to bring the innovation to the market. The complementary assets are divided into generic, specialized, and co-specialized (Teece, 1986). Access to these assets is important for profitability. Who controls the assets determines who among the innovator, the imitator, and the owner of the complementary asset will profit from the innovation.

When the appropriability regime is weak, access to complementary assets becomes the more important. Innovators can build the required complementary assets themselves or they can contract it out. If the complementary asset required is generic, the innovator can contract for access. But when the complementary asset is specialized or co-specialized and the appropriability regime is weak, contracting becomes dangerous and it is better to build the complementary asset within the firm. Also, contracting may result in the owner of the complementary asset becoming a competitor especially when imitation of the innovation is easy. Unfortunately, when the innovator is a start-up and the imitator is the incumbent, the incumbent most often owns the complementary asset required for commercialization (Gans and Stern, 2003). Thus, the owner of the complementary asset may be the one to gain from the innovation

(Rothaermel, 2001; Tripsas, 1997). Tripsas (1997) examined the typesetter industry and found that incumbents were able to survive the battles with new entrants because they owned the specialized complementary assets required for commercialization.

In some instances, substitutors may have competing innovation in the market. The company's commercialization decisions are influenced if there is already an established dominant design. When there is no dominant design for an innovation, it is said to be in the pre-paradigmatic stage. In this stage, there is no accepted and agreed upon "standards" which the design of a product can be based on. Companies fight to have their product design set as the dominant standard. When a promising design emerges and is able to meet most of the user's needs, then the paradigmatic stage has been reached. At this stage, competition shifts away from design towards price. Companies strive towards producing the product at a lower cost than their competitors.

Companies also use access to complementary assets to be able to set dominant designs. Funk (2001) examined how suppliers of phones to DoCoMo established dominant designs over competitors in the mobile phone device market due to DoCoMo giving preferential access to information of the Japanese digital phone standard that DoCoMo controlled.

2.6 Strategic Alliances, Freeware, and Standards

2.6.1 Strategic Alliances

Competition in the high-tech industry is fierce. Companies must constantly invest in the creation of new technological capabilities (Vanhaverbeke, Duyster, and Noorderhaven, 2002). Companies develop these capabilities by developing the technology independently, acquire another company that already possess the technology, or enter into technology-sourcing agreements with outside parties (Schilling and Steensma, 2002).

The cost of developing capabilities internally have become expensive due to short development times and increasing complexity of the technology.

Companies are starting to look outside of their boundaries to acquire technology. Companies acquire technology from external sources through licensing, mergers and acquisitions, or strategic alliances (Vanhaverbeke et al., 2002).

When a company licenses for a technology, the company purchases the rights to another organization's patents or technology for a lump sum payment and royalties. A formal contract between the companies specifies the terms of the arrangement. Licensing agreements give the company limited stake in the technology, yet it can enable to increase its commitment at a later date.

Companies license for technology to gain an early window on emerging technologies to which they may want to commit more fully in the future as the threat of commercial failure subsides (Schilling and Steensma, 2002).

A downside to licensing is that it provides the company somewhat limited control over the technology (compared to an acquisition) because the rights to the technology may be limited by the contract, and because the company does not control the human capital that created the technology.

Companies who want complete control of the technology either acquire or merge with the owner of the technology. But mergers and acquisitions are expensive due to duplicates in assets available within each company prior to the merger (Lambe and Spekman, 1997). To avoid this, companies enter into strategic alliances especially if the technology required comes from multiple companies.

Strategic alliances are voluntary arrangements between two or more firms involving exchange, sharing, or co-development of technologies (Gulati, 1998; Vanhaverbeke et al., 2002). Alliances have advantages over conventional licensing contracts because technological capabilities are frequently based on tacit knowledge making it difficult to draft simple contracts covering sale or licensing of such capabilities (Mowery, Oxley, and Silverman, 1996).

Companies use strategic alliances for several reasons not only to access external technology. Reasons for companies to enter into strategic alliances can be summarized as follows: (Robertson and Gatignon, 1998)

1. gain faster access to new technologies or markets

2. gain advantages of scale in R&D
3. access technological expertise located beyond the boundaries of the firm
4. leverage the comparative advantage of each partner
5. increase the firm's openness to its environment and stimulate internal innovativeness
6. share the risks of R&D beyond the resource of any one firm

However, strategic alliances also involve market contracts. One problem with market contracts that govern exchanges of goods and services is that they are costly to write and enforce. Consequently, such contracts are typically 'incomplete': some terms and conditions of the anticipated exchange are left uncontracted, subject to later negotiations between the parties. These uncontracted dimensions of the exchange, which in essence property rights, are called 'residual rights of control' (Liebeskind, 1996).

2.6.2 Freeware

Software firms distribute a low quality version of its commercial software as freeware (Haruvy and Prasad, 2005). In freeware software, companies retain property rights to the software but the software is available for free (Liao-Troth and Griffith, 2002). Companies release a freeware version in hopes that users pay to upgrade to the commercial version (Haruvy and Prasad, 2005).

To avoid complete cannibalization of the commercial version, companies balance between the quality of the freeware and the price of the commercial version. The freeware version must be of sufficiently high-quality that it is adopted over available substitutes but not of such high-quality that users will not upgrade to the commercial version. The price of the commercial version must also not be too high that users prefer the freeware version over the commercial version (Haruvy and Prasad, 2001).

Companies use freeware to achieve four objectives (Haruvy and Prasad, 2005). First, freeware is used as a signal for quality of the commercial product. It allows users to evaluate the product before buying. Second, freeware deter rivals from entering the market. Third, freeware is used to grow the network of users of the software without lowering the price of the commercial version. Fourth, freeware is used to slow down a competitor attempting to build a network of users.

2.6.3 Standards

Standards are functional specifications that prescribe how the components in a system interact with each other. Most standards go through a formal development process and are approved by an official standards-making body while other standards are developed outside of this formal process (Severance, 1995). One criticism of the formal standard development process is that the process takes too long and that it has no sense of urgency. Sometimes, conflicting interests among participating organizations delay the standardization

process (Sherif and Sparrell, 1992). Standards now lead the technology. Product developers wait for the completion of standards before completing product designs (Hohmann, 1989).

Standards have gained strategic importance in companies. Standards participation is no longer a peripheral activity for companies but has become central activity in implementing its strategies. Companies now have to manage standards participation together with product and technology development (Bailetti and Callahan, 1995).

Companies pursue standardization for several reasons. Companies seek standardization to serve the public interests or comply with anti-trust laws. For new entrants, standardization is a way to challenge dominant or established companies. Standardization may also help legitimize a new technology and allow the company owning or mastering this new technology to gain central position (Sherif, 2001). Companies whose technology is adopted as industry standard make above-average profits (Funk, 2001).

2.7 Lessons Learned from the Literature Review

This section provides the lessons learned from the literature review.

1. *OSS literature can be organized into: (i) motivations of contributors; (ii) innovation process; and (iii) competitive dynamics.* Current literature deals

with why developers contribute to OSS projects; how these projects were developed the software; and how companies attempt to profit from OSS.

2. *Lack of research on how companies that establish open source software projects use the software in their market offers.* Literature is available on how companies attempt to profit from OSS but nothing on how these companies integrate the open source software into their market offers.
3. *No literature on the strategic risks of depending on OSS projects.* Companies participate in OSS projects by incorporating OSS into their market offers. However, there is no literature on the risks for companies depending on OSS.
4. *Motivations to participate in OSS projects are different between individuals and companies; most research has focused on the motivation of individuals.* Companies participate for profit reasons. Participating in OSS projects involves investing resources and affects how they are able to generate revenue from OSS.
5. *Companies use strategic alliances, freeware, and standards to acquire complementary technology, improve speed of development, and improve access to markets.* Developing technological capabilities internally has become expensive. Companies enter into strategic alliances using contractual agreements to develop technological capabilities together with others.

Freeware improve access to markets and standards helps establish dominant design.

6. *Commercialization frameworks are based on ownership of IP and access to complementary assets owned by others.* Who profits from an innovation depends on the innovator's ability to protect the IP of its innovation and its access to complementary assets. When an innovator can protect its IP, it has a high probability of profiting from the innovation. If the complementary asset required to commercialize the innovation is specialized, the owner of that specialized asset has a better chance of profiting from the innovation.
7. *Companies have to integrate programs, services, and systems with their products to create attractive market offers.* It is not enough for companies to develop products but they have to provide add-ons in order to meet or exceed customer expectations.
8. *A company's competitive environment affects its ability to create value for the customer and win over rivals.* A company's competitive environment (the company and its industry's investment in advanced factors; the presence of policies that encourage investments; the presence of strong rivalries and demanding customers within its local market) affects how the company creates market offers and their ability to compete in the international markets.

A company can improve its competitive environment through strategic activities such as philanthropy.

3 RESEARCH METHOD

This research is exploratory and fits in the descriptive stage of the theory building process as defined by Christensen and Carlile (2005). The research method used in this research follows Eisenhardt's (1989) recommendations for exploring management phenomena.

The following steps were undertaken in this research:

1. Define a research question and identify why it is relevant
2. Select criteria for sample inclusion and build database of companies
3. Select theoretical perspective
4. Define data collection protocols
5. Select first wave of 4 companies and collect data
6. Identify market offers and how OSS is integrated into the market offers
7. Generate insights using commercialization frameworks
8. Choose second wave of companies from the database, collect data, and repeat steps 6 and 7
9. Choose third wave of companies from the database, collect data, and repeat steps 6 and 7
10. Reach closure on how OSS contributes to market offers and affects competitive environment
11. Classify the types of market offers and ways OSS is integrated
12. Identify how OSS improves competitive environment
13. Benchmark against strategic alliances, freeware, and standards

14. Identify benefits and risks of setting up OSS projects

3.1 Unit of Analysis, Study Period, and Sample Selection

The unit of analysis was a company who established an OSS project between 1993 and 2005. Two types were considered. One, the company established an open source project. Second, the founders of an open source project created a company to profit from the project they established.

A database that includes the companies that established open source projects does not exist. Thus, a database of such companies was produced using information available online. A search was performed on Google for the terms “open source company”, “open source business model”, and “open source strategies”. Companies mentioned from the search results were investigated to determine if they established open source projects.

The criteria for case selection was:

- The OSS project was established between 1993 and 2005
- In 2005, the company must be operating and have market offers that were enabled by the software from the open source project they established
- Data on the company, its products and services, as well as its open source initiative were available on the Web.

A careful sample selection was done for the second and third wave of companies to introduce variability (Eisenhardt, 1989; Yin, 1984). This reduced the chance of selecting companies with very similar market offers.

3.2 Data Collection

Data was collected in three waves of four companies each. Sources of information about the company included: project websites, annual reports, product literature, press releases, mailing lists, analysts' reports, and media coverage.

Information about the OSS project was collected from company and project websites. Information such as when and why the project was established almost always appears on the Frequently Asked Questions (FAQ) web page of the open source project. Press releases and news related to the OSS project were collected from company websites. Mailing lists, newsgroups, and forums in the OSS project were mined for possible useful data. Information about the market offers was collected from the products and services pages from the companies' websites. A search of the company on Google resulted in a list of press and analyst coverage. These press and analyst coverage information were also collected.

Data collected about each company was stored in separate directories. All collected information was used to determine the companies' market offers, how

OSS is integrated into the market offer, and how OSS affects the company's competitive environment.

3.3 Data Analysis

3.3.1 How OSS is Integrated into Market Offers

Examining the webpages named "Products" and Products and services identified the market offers of each company in the sample. The market offers were then classified based on the extent of the integration between the OSS and the company's proprietary assets (minimal or comprehensive) and the extent to which OSS contributes to the value of the market offer (significant and not significant).

The structure of each market offer was defined using the model advanced by Kotler and Turner (1995) and Anderson et al. (2001). For each market offer, the researcher decided whether the extent of the integration between the OSS and the proprietary assets the company owned was minimal or comprehensive and whether the OSS contributed significantly to the market offer. A comprehensive integration required that the OSS be tightly integrated with the company's proprietary assets. Typically comprehensive integration involves either value-added integration or add-on module integration.

The decision on whether or not the software from the OSS project contributes significantly to a market offer was based on the number of times open source was mentioned in the literature describing the market offer. If open source was mentioned frequently in the literature on the market offer, then the researcher decided that the OSS contributes a significant value to the market offer. If open source was not mentioned once in the literature describing the market offer, the researcher decided that open source did not contribute significantly to the value of the market offer.

3.3.2 Insights using Teece's Commercialization Framework

OSS was conceptualized to be a complementary asset that the company establishes as a public good.

Using constructs from Teece (1986), the researcher first generated insights on how a company uses OSS as a complementary asset and then based on the insights made suggestions on how to modify existing commercialization frameworks.

To generate insights, the researcher examined:

- if the company can protect intellectual property IP
- if the technology is in paradigmatic stage
- if OSS is a generic or specialized complementary asset

3.3.3 How OSS Improve Competitive Environment

For each company in the sample, the researcher used online information to determine how OSS affects the company's competitive environment (Porter, 1990).

For each company, the researcher identified the changes in factor conditions, demand conditions, context for strategy and rivalry, and related and supporting industries directly linked to OSS.

3.3.4 Benefits and Risks of Establishing OSS Projects

Companies publicly announce the benefits they hope to gain from establishing the OSS projects. For each company in the sample, the researcher used online information to identify the strategic benefits of using the OSS from the open source projects the company established.

Companies seldom mention the risks they face when establishing OSS projects. For each company in the sample, the researcher identified the risks of establishing OSS projects mentioned in online information and inferred what the risks were using the available literature.

3.3.5 Benchmark against Alliance, Freeware, and Standards

To benchmark OSS against strategic alliances, freeware, and standards, the researcher first reviewed the literature and then for each company in the sample the researcher reasoned what OSS allowed the company to do that it was unable to do using strategic alliances, freeware, and standards.

4 RESULTS

This chapter presents the results of the study. Section 4.1 identifies the twelve companies included in the sample and describes their OSS projects and market offers. Section 4.2 presents the types of market offers enabled by OSS. Section 4.3 presents the different ways OSS is integrated into the companies' market offers. Section 4.4 presents the insights generated using Teece's commercialization framework (Teece, 1986) and Section 4.5 presents how OSS improves the companies' competitive environments. Section 4.6 provides the benefits and the risks of establishing OSS projects and Section 4.7 describes what companies can do with OSS that is different from strategic alliance, freeware, and standards.

4.1 Companies in the Sample

Appendix A provides a list of companies found to have established open source projects from 1993 to 2005.

Twelve companies were included in the sample. Table 6 shows the companies in the sample with their OSS projects and their market offers organized by the wave in which they were examined.

The first wave of companies included: Apple, RealNetworks, Red Hat, and JBoss.

Apple Corporation established the Darwin project to develop software which is a core operating system that is an integral part of the Mac OS X. Mac OS X is the company's operating system software that runs on all its Macintosh computers. Apple announced in the middle of 2005 that they will start shipping Macintosh computers with Intel microprocessor, instead of PowerPC microprocessors, starting in 2006.

RealNetworks established the Helix project in 2002. The market offers examined were the RealNetworks Broadcast Network (RBN), Real consumer portal, Media Server, and RealNetworks Commercial Source License (RCSL). RBN provides hosting and content delivery services for content providers such as Major League Baseball and ABC News television network. Real is a portal to the company's consumer market offer. Consumers can view ad-supported content for free or subscribe to premium digital content. Media Server is the proprietary server built on top of Helix. The Media Server includes advanced networking functionalities not available in the OSS version. The RCSL is a commercial license for customers that do not or cannot abide by RealNetworks' Public Source License (RPSL). Customers who buy RCSL can keep their applications developed on top of Helix private. They are also permitted to integrate and distribute RealNetworks' proprietary codecs into their applications.

Red Hat established the Fedora project in 2003. The company takes applications from the project, test it, and package it as Red Hat Enterprise Linux (RHEL) for

its customers. Red Hat provides training and professional certification for those who want to get certified on RHEL. Third parties can also certify their hardware and software for compatibility with RHEL. Customers can purchase subscription to Red Hat Network (RHN) to receive updates, upgrade, and fixes, as well as management and monitoring capabilities of their RHEL deployments.

Table 6 Companies and OSS Projects in the Sample
1st Wave (4 Companies, 11 Market Offers)
<ul style="list-style-type: none"> • Apple (Darwin) – Macintosh • RealNetworks (Helix) – RBN, Real, Media Server, and RCSL • Red Hat (Fedora) – Training, HW/SW Certification, and RHN • JBoss (JBoss) – Professional Certification, Solutions Certification, and Subscription
2nd Wave (4 Companies, 7 Market Offers)
<ul style="list-style-type: none"> • IBM (Eclipse, Derby) – WebSphere Studio, DB2 • SGI (OpenGL SI) – Trademark License • MySQL (MySQL) – Commercial License, MySQL Network, and Training • Digium (Asterisk) – Telecom Cards
3rd Wave (4 Companies, 6 Market Offers)
<ul style="list-style-type: none"> • Sourcefire (Snort) – VRT, DC, RNA • JasperSoft (JasperReports) – JasperDecisions • SpikeSource (SpikeSource) – Support • ActiveGrid (ActiveGrid) - Support

JBoss started out as a project led by the CEO Marc Fleury in 1999. He then established, in 2002, JBoss Inc. to profit from the JBoss project. Like Red Hat, JBoss provides professional certification, hardware and software compatibility certification, and also subscriptions to JBoss Network. Individuals can receive training from JBoss and its partners to get professional certification. Some

companies, who join JBoss' partner program, are required to have a minimum number of employees certified on JBoss. Partner companies can also certify their hardware and software with JBoss to prove compatibility. JBoss offers subscription to JBoss Network. Customers who subscribe get updates, upgrades, and fixes as well as ability to manage their JBoss deployments.

The second wave of companies included: IBM, SGI, MySQL, and Digium.

IBM established Eclipse in 1999. The market offers examined are WebSphere Studio and DB2. WebSphere Studio is a visual development tool built on top of Eclipse. Cloudscape is a relational database that IBM released as open source to the Apache project in 2004. Cloudscape's open source project is called Apache Derby. The company provides customers technical support on Cloudscape.

However, Cloudscape does not have enterprise level features found in commercial enterprise databases. Customers who want to get enterprise level functionalities can upgrade to IBM's proprietary database product called DB2. IBM provides plug-ins to Eclipse based tools, such as WebSphere Studio, so customers can migrate easily from Cloudscape to DB2.

Silicon Graphics Inc. (SGI) released OpenGL Sample Implementation (OpenGL SI) in 1999. OpenGL is the industry standard for 3D graphics. Companies that want to support OpenGL in their graphics hardware use the sample implementation to create drivers and graphics libraries for the hardware.

Companies pay SGI to use the OpenGL trademark in their hardware products. Software companies that develop applications to take advantage of OpenGL capabilities do not have to pay for trademark licenses. This encourages software companies to use OpenGL instead of other competing graphics libraries.

MySQL established their OSS project in 1995. MySQL offers commercial license for customers who do not or cannot abide by the open source license. Through subscriptions, customers can have access to the MySQL Network to receive upgrades and fixes. The company also offers training for those who want to get certified on MySQL. MySQL incorporates another OSS from a company called InnoDB. The software from InnoDB adds advanced transaction capabilities which is critical functionality missing in MySQL. However, InnoDB was purchased in 2005 by Oracle which is a competitor of MySQL.

Asterisk started out as a project by Mark Spencer when he was establishing his Linux consulting business. When Asterisk started to become popular, Mark Spencer concentrated on the project and set a company called Digium to provide support for Asterisk. Digium builds telecom cards to make Asterisk work with the traditional telecommunications infrastructure.

The third wave of companies included: Sourcefire, JasperSoft, SpikeSource, and ActiveGrid.

Sourcefire was founded by Martin Roesch to provide a commercial version of the Snort intrusion detection technology he wrote and released as open source. The company offers Sourcefire Defense Center (DC), Sourcefire RNA (Real-time Network Awareness) and subscription to the Vulnerability Research Team (VRT) Certified Rules. The Defense Center and RNA complements Snort to give customers a Real-time view of the security situation on their network infrastructure. Together these market offers provide a defense system for their customers against threats. The VRT Certified Rules are used by the intrusion detection technology to detect, assess threats, and actions to neutralize those threats. Sourcefire was acquired by Check Point in October of 2005.

JasperSoft is the company behind JasperReports, an open source reporting engine. JasperReports was started by Teodor Danciu in 2001 and founded JasperSoft to profit from the project. He remains the lead developer and architect of the project. JasperReports contain libraries that allow user to develop applications to create static business reports that can be displayed on the screen or sent to a printer. If users want dynamic reports that are highly interactive and deliver critical business information in real-time, they can upgrade to the companies proprietary application called JasperDecisions. When customers upgrade to JasperDecisions, their investments are not lost because the applications they built to generate static reports using JasperReports works inside JasperDecisions.

SpikeSource takes software from several open source projects and assembles them into a stack. They integrate, test, and certify this stack assuring customers that the software from different project works well with each other. The company adds software to manage the stack and also released as open source.

SpikeSource offers professional support and customization services to the customers of its stack.

ActiveGrid was founded in 2003. The company develops two products:

ActiveGrid LAMP (Linux/Apache/MySQL/Perl/PHP/Python) Application Server and ActiveGrid Application Builder. These products allow customers to develop rich and highly interactive internet applications that integrate a number of data from different sources. The Application Server is able to scale using low-end commodity hardware unlike those from the competition where scaling involves expensive specialized hardware. The company provides advanced features and commercial support to their customers.

4.2 Types of Market Offers

The data suggests that there are six types of market offers enabled by OSS.

Table 7 identifies the 24 market offers of the companies in the sample organized into six types.

Table 7 Types of Market Offers		
Types	No.	Company (Market Offer)
A) Hosting and Content Services	2	RealNetworks (RBN), RealNetworks (Real)
B) Training and Testing Services	5	JBoss (Professional Certification), Red Hat (Training), MySQL (Training), JBoss (HW/SW Certification), Red Hat (HW/SW Certification)
C) Support Services	2	ActiveGrid (Support), SpikeSource (Support)
D) Subscription Services	4	JBoss (JBoss Subscription), Red Hat (RHN), MySQL (MySQL Network), Sourcefire (VRT)
E) Commercial Licenses	3	RealNetworks (RCSL), MySQL (Commercial License), SGI (Trademark License)
F) Product/Applications	8	RealNetworks (Media Server), IBM (WebSphere Studio), IBM (DB2), JasperSoft (JasperDecisions), Apple (Macintosh), Digium (Telecom Cards), Sourcefire (DC), Sourcefire (RNA)
	24	

A) The hosting service is a content delivery network that allows content creators to outsource delivery of their digital multimedia content. Content creators pay the company to host, manage, and deliver the content to their intended audience. Content services deliver ad-supported for free and subscription to premium content to end-users through the consumer website maintained by the company.

B) Training and testing services provide professional certifications for those who want to show they have the skills, knowledge, and capability to work with the software. The companies tests and certify hardware and software

from their partners. When the partners certify their products against the company's software, this gives assurance to customers that the products integrates and works effectively together.

- C) Companies provide support services to their customers of open source software. They provide fixes to the software, advice and suggestions for deploying the software in their customers' environment. The companies also provide custom development to meet certain requirements from their customers.
- D) Companies provide subscription services to their customers. Customers subscribe to receive updates, upgrades, monitoring and management capabilities of their software deployments. They continue to receive these services as long as they are subscribed. Companies do not only offer subscription to updates but also on complements, e.g., VRT Certified Rules, critical to the software.
- E) For customers, that do not or cannot abide by the open source license, they can purchase commercial licenses from the companies that have undisputed rights to the software. When customers purchase commercial licenses, the company may also allow certain proprietary assets (e.g., codecs) to be integrated into the customers' applications. Customers may also use trademarks owned by the company.

- F) Companies develop products and applications that integrate or complement the open source software to create a system. The products and applications may provide advanced features and capabilities over what is available in the open source version. The companies then sell these products bundled with other add-ons to their customers.

4.3 Ways OSS is Integrated into Market Offers

The market offers that rely on OSS projects can be classified based on the extent of integration and OSS contribution to the market offers as shown in Figure 5 and Figure 6.

Comprehensive integration means that the software is tightly integrated with proprietary assets to create the market offer. This involves either value-added integration or add-on module integration. Figure 5 presents the market offers of the companies and how many market offers belong to each cell. Figure 6 shows a pictorial representation of the different ways the software is integrated into the different market offers. The pictorial representation illustrates that although companies can provide the same type of market offers, there are different ways they can integrate OSS into their market offers. The letter inside the square brackets refers to the type market offer explained in the previous section.

Appendix B provides information that was used to create Figure 5 and Figure 6.

4.3.1 Minimal-No

The Minimal-No cell includes the two market offers where the OSS was minimally integrated with the proprietary assets of the firm and the OSS does not contribute significantly to the value of the market offer.

The two market offers in this cell include: the sale of hosting and content services. In this type of integration, denoted as (i) in the cell, the OSS is used as one of the components to the network that delivers the hosting and content services. RealNetworks provide hosting and content services by integrating Helix into the company's content delivery network. The OSS (black box as shown in Figure 6) and proprietary software are comprehensively integrated internally into an application module which is then minimally integrated with the network (white box) that delivers the hosted service and the sale of content (oval shape).

Hosting and content services are market offers that use OSS as a component of the network that delivers the services. Although, OSS is comprehensively integrated into an application module, the application is minimally integrated with the network.

Figure 5 OSS Integration and Contribution to Market Offer

		Can OSS contribute significantly to the company's market offer?	
		No	Yes
How extensive is the integration between proprietary assets and OSS?	Minimal	Sale of: (i) content services using delivery systems that use OSS (RealNetworks RBN and Real) 2 of 24	Sale of: (i) training and testing services anchored on OSS (Red Hat and JBoss training and testing, MySQL training), (ii) proprietary product using OSS as entry-level product (IBM DB2), (iii) support services on a stack that includes other OSS and proprietary complements (SpikeSource and ActiveGrid support), (iv) commercial and trademark licenses (RealNetworks, MySQL, SGI) 11 of 24
	Comprehensive	Sale of: (i) products with OSS embedded in a component (Macintosh) 1 of 24	Sale of: (i) applications that integrate OSS and proprietary software/hardware (RealNetworks Media Server, IBM WS Studio, Digium Telecom cards), (ii) subscription services (Red Hat, MySQL, JBoss, and Sourcefire VRT), (iii) proprietary product that use OSS as entry-level product (JasperSoft JasperDecisions), (iv) proprietary complements that works with the OSS (Sourcefire DC, RNA) 10 of 24

4.3.2 Minimal-Yes

The Minimal-Yes cell includes the 11 market offers where the OSS was minimally integrated with the proprietary assets of the firm and the OSS contributes significantly to the value of the market offer.

The 11 market offers in the Minimal-Yes cell can be organized into four types of market offers as shown in Figure 5: (i) training and compatibility certification services; (ii) OSS as entry-level to proprietary product (the proprietary product is different from the entry-level open source product), (iii) a certified stack of OSS with commercial support, (iv) and the company offers commercial and trademark licenses.

In the integration type denoted (i) in the Minimal-Yes cell shown in Figure 6, OSS is used to anchor training and testing services. There are 5 market offers that used this type of integration (see Table 7). Companies, such as Red Hat and JBoss, provide professional certifications to individuals who want to get certified on the OSS. These certificates are usually tied to the OSS and carry the brand and trademarks owned by the company. The companies maintain tight control of how others can use the brands associated with the project. Others can offer skills certification but they are not allowed to use JBoss' trademarks in their offers. By controlling the brand and trademarks, the company can let partners provide

training to individuals while the company provides the examination and certificates.

Partners that want to certify their hardware and software to prove compatibility have to pass rigorous testing requirements established by the company. Partners use the test tools, provided by the company, to run tests on their hardware and software. Red Hat provides a test tool to its partners to certify hardware and software against RHEL. Results of the tests are sent to the company and evaluated if they pass conformance tests. If the company is satisfied with the results, the company issues a certificate of compatibility to the partner.

In the integration type denoted (ii) in the Minimal-Yes cell shown in Figure 6 the company uses two OSS projects it established to create attractive market offers. IBM uses both Eclipse and Apache Derby to deliver the market offer.

Cloudscape, IBM's commercial version of the open source Apache Derby, is used as an entry-level product to seed demand for the company's high-level proprietary database product called DB2 (white box in Figure 6). IBM offers Cloudscape for free and companies can purchase limited support from the company. However, Cloudscape does not have features critical to deliver enterprise-level capabilities. If customers want those advanced enterprise level features, they have to upgrade to DB2. So that customers will not have any difficulty migrating from Cloudscape to DB2, the company created plug-ins that customers can install into tools (e.g., WebSphere Studio) developed on top of

Eclipse, the company's other OSS project. In this instance, the proprietary product replaces the entry-level product in the customer's deployment environment.

The integration type denoted (iii) in the Minimal-Yes cell shown in Figure 6, take software from OSS projects that they did not establish and proprietary software from other companies. They then integrate them into a stack, add advanced functionalities such as management capabilities to the software, perform tests, and certify the integrated software. ActiveGrid, for example, adds advanced grid functionalities so the applications developed using their stack can scale using low-cost commodity hardware. SpikeSource, on the other hand, offers management features of the stack that it creates. Both companies then provide support and customization services to their customers.

The integration type denoted (iv) in the Minimal-Yes cell shown in Figure 6, uses commercial and trademark licenses as its market offer. Commercial licenses allow customers to use other proprietary assets such as components or trademarks owned by the company. For example, RealNetworks allows its customers to integrate the company's proprietary codecs when they purchase commercial license. SGI allows customers to use the OpenGL trademark in their hardware products when they purchase a trademark license.

Some companies incorporate other OSS from partner companies. These companies have contractual agreements to allow the company to integrate its partner's software and charge for a commercial license. MySQL has such an agreement with InnoDB who owns the rights to the software that provides MySQL's advanced transaction capabilities. Customers can purchase a commercial license for the software directly from MySQL without the need to purchase another commercial license from InnoDB.

4.3.3 Comprehensive-No

For the Comprehensive-No cell shown in Figure 6, there is one market offer with a single example of OSS integration: (i) the OSS is integrated with proprietary software and hardware. In this integration, the OSS and proprietary software (i.e., hardware drivers and application framework) are comprehensively integrated internally into a module (i.e., operating system) which is placed at the interface between what the company can protect and the hardware it purchases; the interface module is a small part of the product. Apple has determined that their customers value the user experience delivered by their graphical interface and know that the company can protect the technology that facilitates this. At the same time, the company heavily depends on a single supplier for its specialized hardware (i.e., microprocessor) needed to create its Macintosh market offer. The company integrates Darwin with the proprietary software that delivers the user experience to create Mac OS X, Apple's operating system for the Mac. Mac OS X runs on top of the proprietary hardware used in the Macintosh. Darwin

separates what the company's customer value and what the company depends on from its supplier.

4.3.4 Comprehensive-Yes

For the Comprehensive-Yes cell, there are 10 market offers and five examples of how OSS is integrated with the market offers: (i) the OSS is integrated with proprietary software and hardware; (ii) the company integrates OSS with network infrastructure to provide subscription services; (iii) the company offers entry-level product for its proprietary product, the entry-level product becomes a component of the proprietary product; (iv) the company integrates the OSS with other complementary products to work together as a system and the OSS community develops complementary content that is required by the software.

The integration type denoted (i) in the Comprehensive-Yes cell shown in Figure 6, illustrates companies integrate OSS with proprietary complements to create a product or application. These proprietary complements may be in the form of software or hardware that delivers advanced functionalities. As an example, RealNetworks adds advanced networking capabilities to its Media Server while Digium develops hardware to allow customers using Asterisk to be able to connect to traditional telecommunications infrastructure.

The integration type denoted (ii) in the Comprehensive-Yes cell shown in Figure 6, illustrates companies create infrastructures that provide extra added services and functionalities to the OSS. Companies, such as Red Hat and JBoss, offer subscription to customers for access to these infrastructures. The company offers the software at low cost or free to their customers. To get updates, fixes, upgrades, and monitoring and management capabilities, the customers have to subscribe to be able to access the company's infrastructure that stores and delivers these extra added services and capabilities. Another company, Sourcefire, offers subscription to its VRT Certified Rules (white circle). These Certified Rules are maintained by security experts giving customers early access to protection against new threats discovered by the Vulnerability Research Team.

The integration type denoted (iii) in the Comprehensive-Yes cell shown in Figure 6, the company uses OSS to seed demand for the company's proprietary application. JasperSoft offers proprietary application, called JasperDecisions, that has advanced functionalities not available in the entry-level JasperReports. When the customer migrates to JasperDecisions, the customer is still able to use the applications built using JasperReports inside the proprietary application. This is different from the IBM example above where the proprietary application replaced the OSS application in the customer's deployment environment.

The integration type denoted (iv) in the Comprehensive-Yes cell shown in Figure 6, the company develops complementary products that work with each other to

create a system. Sourcefire develops complementary products called Sourcefire Defense Center and Sourcefire RNA. These complements and the Snort intrusion detection technology form the customer's defense infrastructure. The complementary products have different purposes such as collecting data and monitoring the network. The intrusion detection technology monitors for threats and the RNA collects real-time status of the equipments within the network. The data is sent to the Defense Center which aggregates the information to give the customers a real-time view of security events in its infrastructure. The OSS project also generates Snort Rules (black circle) which users can use in their intrusion detection deployments to discover threats. This is a case where the project not only develops software but content critical to the software as well.

4.4 Insights Using Commercialization Framework

Using Teece's commercialization framework (Teece, 1986), insights were generated with the perspective that OSS is a complementary asset key to the companies' commercialization strategies. Appendix C provides the information used to generate the insights.

OSS lowers development cost and avoids the hazards of contractual agreements for access to complementary assets

OSS has allowed companies to lower the product development costs and avoid the hazards of contracting for complementary assets. The companies reduce risks of irreversible investments when business relationships anchored around

contractual agreements for access to complementary assets does not work out. Apple established Darwin to get the community involved in accelerating support for the Intel microprocessor. The company lowered cost and avoided contracting for the development a core operating system for their Mac OS X without exposing the proprietary components that their customers value. RealNetworks established Helix and got the community involved in porting the platform into other environments. Companies, such as Nokia, participated in the project to make the platform work in their mobile phones. RealNetworks lowered the cost and avoided contracting for the development of Helix which is the foundation of their market offers.

OSS allows the company to select the right design, couple with the market, and prototype quickly during the pre-paradigmatic stage

The digital content delivery technology is still in the pre-paradigmatic stage.

RealNetworks and Microsoft are battling it out to establish a dominant standard.

By releasing Helix as open source, RealNetworks is trying to couple with the market and influence dominant design of the technology. The community is involved in the design, development, and roadmap of the Helix platform. The company receives feedback from the user community on what they value.

Sometimes, the users implement it themselves and contribute it to the project.

RealNetworks has also prototyped market offers quickly across several layers of market segments. RCSL is a market offer for the technology market, Media Server is an offer for those who want to roll their own content delivery

infrastructure, hosting services are for those that want to outsource content delivery to the experts, and content services are targeted towards end users. This prototyping allows RealNetworks to determine which market offers are profitable and establish to become a dominant player in that segment.

Table 8 Insights using Commercialization Frameworks	Company
OSS lowers development cost and avoids the hazards of contractual agreements for access to complementary assets	Apple, RealNetworks
OSS allows the company to select the right design, couple with the market, and prototype quickly during the pre-paradigmatic stage	RealNetworks
OSS allows the company to set and maintain dominant design	IBM, SGI
OSS allows the company to integrate other OSS functionality without contracting	MySQL, SpikeSource, ActiveGrid
OSS projects produce software and content critical to the software	Sourcefire

OSS allows the company to set and maintain dominant design

IBM established Eclipse and Eclipse's architecture has since then become a dominant standard for visual development environments. Competitors have adopted its plug-in architecture in the design of visual development tools. SGI has maintained the dominant standard for 3D graphics by releasing OpenGL. Open source projects, such as XFree86 and Mesa, are able to provide support for Linux and other environments. The community can help improve OpenGL as the 3D hardware graphics technology evolves helping keep OpenGL the dominant standard.

OSS allows the company to integrate other OSS to add functionality without contracting

Companies have integrated software from other OSS projects into their OSS software without contracting. MySQL uses InnoDB's open source transaction engine that provides advanced transaction capability of MySQL. Both companies offers commercial license for customers who do not want to release applications built using MySQL and InnoDB as open source. Companies, such as SpikeSource and ActiveGrid, take software from other projects to build their LAMP stack without contracting for access. These companies provide value-added features to the LAMP stack then provide commercial support.

OSS project produce software and content critical to the software

An OSS project can produce not only the software but a critical complement to the software as well. Sourcefire's Snort community contributes Snort Rules that complements the intrusion detection software. Snort Rules provides the data required by the software to detect attacks and the actions to take when a particular attack is detected.

4.5 How OSS Improves the Companies Competitive Environment

Table 9 identifies how OSS improves the company's factor conditions, demand conditions, context for strategy and rivalry, and related and supporting industries. Appendix D provides the information used to provide the insights.

4.5.1 Factor Conditions

OSS helps improve the companies' advanced factor conditions. Companies get members of the community to enhance the value of the company's products. Members of the community help develop components and complements for the company's products. For example, SpikeSource and ActiveGrid take software from other OSS projects and integrate them into a certified stack. RealNetworks' Helix is ported into new environments by members of the community and Snort's community members contributing Snort Rules.

OSS projects provide companies with highly skilled labor. Companies can lookout for highly skilled talent and hire from the community. Members in the community develop skills when they use and contribute to the project. This lowers the cost of training new hires for the company. Members of the community train and upgrade their skills keeping abreast of new technologies and methodologies. Apple hired Jordan Hubbard who was the leader of the FreeBSD project. Jordan Hubbard brought with him credibility in the open source community as well as the experience and lessons learned on the challenges he faced when he was leading the FreeBSD project. JBoss hires the top three developers on the projects the company wants to support. This gives JBoss' customers confidence that the company can help them solve their problems because the knowledge experts work for the company.

Table 9 How OSS Improves the Companies OSS Environment	
Factor Conditions	<ul style="list-style-type: none"> • Get OSS developers to do work that enhances the value of the company's products • Develop stacks of components that can be embedded into customer's products • Obtain products that complements company's products • Hire talent for company • Use OSS to experiment with new technology
Demand Conditions	<ul style="list-style-type: none"> • Experiment with new product concepts • Lower costs of acquiring customers • Create demand for new market offers
Context for Strategy and Rivalry	<ul style="list-style-type: none"> • Set and maintain dominant standard • Create dominant position in delivery of services associated with the brand of the OSS project • Acquire external technology without the need for contractual agreements • Shift motive to license to customers • Low barrier to imitation pressure companies to develop novel ways to profit from innovation
Related and Supporting Industries	<ul style="list-style-type: none"> • Promotes knowledge sharing and co-create value • Influence technical efforts of suppliers and partners

Companies use OSS projects to experiment with new technology. Some of these technologies are the result of government or academic research. Companies can use the project as a proving ground for the new technology. Red Hat uses the Fedora project to tests out new technologies. The community helps in improving and refining these new technologies. Some members often deploy these technologies at their work environment to test their applicability. When the technology has finally matured, Red Hat packages it in RHEL.

4.5.2 Demand Conditions

OSS improves companies' demand conditions by allowing companies to experiment with new market offers; lower cost of acquiring customers; and create demand for new market offers.

In OSS, all developers are users. Companies can experiment with new market offers based on feedback from its users. Sometimes, developers implement features that they want and contribute them to the project. The result is companies get clearer and earlier insights into emerging customer needs.

OSS allows potential customers to try out the company's product to determine if it solves their problems. Once they are satisfied that they are able to make use of the software, the customer can purchase from the company's market offers. For example, customers are able to download certified stacks from SpikeSource or application server from JBoss. They can deploy it into their infrastructure and purchase support of subscriptions from the company. This lowers the cost of customer acquisition as customers, who have tried the product, approaches the company to buy market offers.

OSS allows companies to create demand for high-end proprietary products. Customers use entry-level products if the available functionalities meet their needs. When the customers outgrow the entry-level products features, they can

upgrade without losing their investments to the company's high-end proprietary products. JasperSoft and IBM both offer customers entry-level products with options to migrate to high-end proprietary products.

4.5.3 Context for Strategy and Rivalry

OSS allows the companies to set and maintain dominant standard; create dominant position in delivery of services associated with the brand of the OSS project; acquire technology without contractual agreements; change motivation for licensing; develop new ways of profiting from innovation.

Companies use OSS to set and maintain dominant standards without undergoing through long formal standardization process. OSS can let the market decide on which design to adopt. RealNetworks and Microsoft are battling it out to set dominant standard for digital content distribution. IBM has developed a plug-in architecture standard, which competitors have adopted, for visual development tools.

OSS weakens appropriability regime due to loss of ability to protect IP on the software's source code. However, companies are able to establish dominant position on delivery of services because they control the brand of the OSS project. Others can offer competing services but are not allowed to use the brands associated with the OSS project. This makes it difficult for others to appropriate value from OSS projects established by the company. For example,

other companies are allowed to distribute software created by Red Hat's Fedora project but are not allowed to use Red Hat's trademarks and brand. Third parties can offer training and certification on JBoss' application server but are not allowed to the company's trademarks.

OSS allows the company to acquire external technology without the need for contractual agreements. Members of the OSS project can develop and improve technology which the company can use to create attractive market offers. There is no formal contractual agreement between the company and the community. This provides new challenges for management teams on how to motivate the community to deliver within the company's timeline and roadmaps.

OSS has changed the motivation of customers and companies to licensing technology. Previously, companies license their technology to customers with provisions that prevents the customer from stealing the IP to the technology. With OSS, it is the reverse. Customers purchase licenses to prevent losing the IP on their technology. Companies, such as RealNetworks and MySQL, offer commercial licenses because they are not worried about customers stealing their IP but customers want to keep their own IP.

Low barriers to imitation have pressured companies to develop novel ways to profit from the OSS project. Companies are building expensive and specialized complementary assets to deliver services enabled by OSS. Red Hat developed

infrastructure to deliver updates, upgrades, and management and monitoring capabilities to subscribers of Red Hat Network.

4.5.4 Related and Supporting Industries

OSS allows the companies to share knowledge and co-create value with members of its related and supporting industries. Members, through the project, share ideas, solve problems, and create value. Membership to the OSS project is not constrained by proximity. Companies set up infrastructure that allows members to participate through mailing lists, forums, and source code management systems. The company also benefits from participation of experts outside the boundaries of the company. RealNetworks involves members of the community to develop roadmaps on the Helix project. Companies, like Nokia, provide the roadmaps and timelines to offer support for Helix in their mobile environment.

OSS allows the companies to influence technical efforts of its supplier and partners. Companies through their OSS projects try to influence other OSS projects that they depend on. The OSS project can serve as a test bed for assessing and improving compatibility of products. The project also accelerates the pace of innovation as companies work together to share ideas and tackle technical problems.

4.6 Benefits and Risks of Establishing OSS Project

This section identifies the benefits and risks of using the OSS produced by an OSS project a company establishes.

4.6.1 Benefits

Companies that established OSS projects have reaped the following benefits:

1. Gain mindshare with developers and influence other open source projects.
Through their project, they are able to influence other OSS projects to meet their own objectives and goals. Having established an OSS project gives them credibility to the open source community resulting in little or no resistance when they push their own agendas. JBoss and Red Hat not only controls their own project but also try to influence other OSS projects where they participate.
2. Choose license of the project and have the ability to charge for license fees and royalties. Companies do not give up copyright to the source code of the software created in their project. Thus, they are able to charge for commercial license fees to customers who are not able to abide by the open source license. RealNetworks and MySQL offer dual licenses for their software.
3. Corner position in value network such as training and certification by controlling the brand. Companies own the trademarks associated with the OSS project. This gives them a dominant position over others in providing services because they can brand their offering tightly to the OSS project.

4. Control technical architecture, roadmap to make sure it aligns with company objectives, i.e. become a lead developer instead of user. Companies can still control or influence the project to meet their objectives. They provide resources to the project and set its direction.

4.6.2 Risks

Establishing OSS projects carries the following risks:

1. Risk that a competitor may own (through acquisition) a critical complementary asset. Companies that depend on other OSS projects are at risk when a competitor suddenly acquires the rights to that project. This has a greater likelihood of happening when the other OSS project is a sponsored project. A sponsored project is usually owned by a single organization. MySQL integrates InnoDB's transaction engine into its database but in October 2005, Oracle purchased InnoDB. It is still not known how this affects MySQL's future market offer.
2. Risk that a community does not develop or disappears. OSS projects depend on having a vibrant community to benefit from contributions. If the project is not managed properly or interests from contributor wanes, the company may end up as the only contributor of resources without benefiting from OSS. Some companies in this study are new and the community has not yet developed. Companies such as SpikeSource and ActiveGrid have to convince outsiders to participate in their project.

Apple's relationship with its community is starting to sour because of outside developers having difficulty in contributing to the project.

3. Risk that customers may not upgrade to proprietary products because entry-level is "good enough". Companies that provide entry-level products from their project are at risk that the entry-level product catches up to their high-end products. This may result in lost opportunity for the company.

4.7 Benchmark against Strategic Alliance, Freeware, and Standards

OSS projects have allowed companies to achieve strategic objectives that are not attainable with strategic alliances, freeware, and standards.

4.7.1 Strategic Alliances

Companies use strategic alliances to acquire external technology. Strategic alliances involve contractual obligations that are difficult to write and enforce.

OSS allows the company to acquire external technology without the burden of contractual obligations.

OSS has allowed Apple to migrate architecture from integral to modular and change key supplier of their microprocessor from IBM to Intel. Apple could have established a strategic alliance to make their software work on Intel microprocessors and give the company options to switch platforms for its Macintosh market offer.

Previously, companies create strategic alliance to gain faster access to new markets. With OSS, partners who participate in the OSS project help the company to prototype market offers and access new markets. RealNetworks has gained significant foothold in the mobile market when companies such as Nokia added support for Helix in their environment. RealNetworks has prototyped a market offer delivering digital multimedia content to mobile devices and found that it is a source of tremendous profits.

Companies provide entry-level products to seed demand for high-level proprietary products without investing significant resources on the development of the entry-level product. Previously, companies develop entry-level products internally or through strategic alliances. OSS can be used to develop these entry-level products. JasperSoft and IBM have used software from OSS projects to seed demand for their proprietary products. The companies make sure that investments made by customers in using the entry-level products are not lost when they migrate to the high-level products. JasperSoft enables reports created using JasperReports to run within JasperDecisions and IBM created plug-ins to Eclipse so customers can move the data stored in the Cloudscape database into DB2.

OSS allows the company to use the community to develop content that adds value to the software. The community for Sourcefire's Snort project develops Snort Rules that are needed by the intrusion detection software. OSS project has

allowed Sourcefire to access expertise outside the boundaries of the firm. Some members of the community are security experts and security administrators of networks. The company is able to tap into this community to identify and handle new security threats without the need to setup strategic alliances and spending significant investment in resources.

4.7.2 Freeware

Freeware allow companies to create and gain access to new markets. OSS allows the companies to do the same things and mentioned above where companies use OSS to seed demand for other market offers and prototype market offers.

OSS differs from freeware in creating and appropriating value. In freeware, value is created and appropriated only by the company. The company offers two versions of the software: freeware and commercial. The freeware has less available features than the commercial version. In both cases, the company creates value and appropriates value. In OSS, the company co-creates value with the community in the OSS project. The company can appropriate from the portion created by the community. Others are able to appropriate value from the OSS project but they are not able to use proprietary assets, such as brands and trademarks, owned by the company that established the project.

4.7.3 Standards

With respect to standards, OSS allows the company to standardize implementations not just functional definitions. Standards define how components behave while OSS actually implements these behaviors. In most instances, standards are first set by standards making bodies before companies start implementing the functional definitions. In OSS, companies implement the technology first, tests, and prove it in actual situations, then, when the technology has proven to work, it gets accepted as a standard. JBoss' implementation on some of its application server technologies are being incorporated into the J2EE standard.

OSS allows the company to own the copyright of the structure that produces the standard. In the traditional standardization process, the standards making body owns the copyright of the standard. In OSS, when a technology becomes a dominant standard, the owner of that technology retains the copyright. SGI retains the copyright of the OpenGL Sample Implementation while becoming the standard for 3D graphics. Other companies use OpenGL SI to develop drivers and graphic libraries for their hardware.

5 DISCUSSION OF RESULTS

This chapter is organized into two sections. Section 5.1 provides the key findings of this research. Section 5.2 identifies opportunities to improve Teece's framework (Teece, 1986).

5.1 Key Findings

5.1.1 OSS Based Solutions

The companies in the sample developed six types of market offers enabled by the software from the open source projects they established. These market offers are: (i) hosting and content services, (ii) training and testing services, (iii) support services, (iii) subscription services, (iv) commercial licenses, and (v) products and applications.

Companies integrate OSS with proprietary development and proprietary assets to deliver what their customers pay for. Companies make money not from the source code of the open source software but from market offers that deliver value to their customers. Some companies offer their IP, such as copyrights and trademarks, to customers so they can use the companies' brands in their products. The companies also sell commercial licenses to customers who cannot abide by the open source license. Other companies build infrastructure to deliver fixes and upgrades to the software, and advanced software functionalities such as management and monitoring of the software applications. Customers obtain access to this infrastructure by paying subscriptions to the companies' services.

5.1.2 OSS as an Alternative to Buy or Build

Prior to OSS projects, a company had two alternatives to access complementary assets: to own the organization that produced the asset or contract the asset from the supplier. OSS provides a third alternative. The company can obtain the asset without owning the organization that produces it or buying it from the supplier.

Take the situation at Apple. When Apple migrated from an integrated to a modular product design, there was no need to outsource extensively. The company relied on OSS. Previously, a company migrating from an integrated to a modular product design would have had to setup strategic alliances with other organizations to outsource the development of technology. OSS lowered development costs and avoided the hazards of contracting.

Apple identified which part of the technology it could protect (i.e., user experience) and established an OSS project that developed the part of the technology that the company could not protect (i.e., core operating system). The company outsourced development of its core operating system to the open source community enabling them to reduce their dependence on one of their key suppliers.

5.1.3 Lower Cost Solution to Achieve Strategic Objectives

Companies approached OSS as a lower cost solution to accomplish strategic objectives such as: change key suppliers, prototype market offers to get clearer and earlier insights on emerging markets and customer needs, seed demand for proprietary products, develop components and complements for their products, and set dominant standards.

Table 10 shows the main objectives that the companies were trying to achieve using OSS.

Table 10 Companies Use OSS to Achieve Strategic Objectives	
Objective	Company
Migrate architecture from integral to modular and change key supplier	Apple
Accelerate adoption of offer (i.e. prototype new offers fast)	RealNetworks
Provide an entry-level product to seed demand for proprietary product	IBM, JasperSoft
Use community to develop content that adds value to the software	Sourcefire
Use OSS project as proving ground of technology and address adjacent markets	JBoss, Red Hat
Set and maintain dominant design or standard	IBM, RealNetworks, SGI
Incorporate critical functionality into the company's product	MySQL
Develop complementary products	Digium, Sourcefire
Acquire components to assemble into a stack to create a product	ActiveGrid, SpikeSource

5.1.4 Companies control the OSS projects and influence other projects

Companies do not give up control of the OSS projects that they establish. A company that sets up an OSS project sets the direction of the project by providing resources, technical architecture, and roadmaps and makes sure the open source project aligns with the company's objectives. A company can control the OSS project by enforcing the trademarks and brands associated with the project that it owns. This limits how competitors are able to use the software from the project in their own market offers.

Companies also use the project to influence other open source projects that they depend on. These projects may be the source of critical components that the company needs in their products. Companies can provide technical inputs and resources so that these projects' direction aligns with the companies' objectives. However, a company face risks when it depends on open source projects that it does not control. A competitor of the company that depends on open source projects may purchase the company that controls these projects. This may negatively affect the company's ability to integrate the open source software to create attractive market offers.

5.2 Issues with Teece's Commercialization Framework

This study found that the current commercialization framework advanced by Teece (1986) does not adequately deal with how OSS is used as a complementary asset in companies' commercialization efforts. Table 11 identifies

what was observed in the cases that is not properly explained by Teece's commercialization framework.

Companies can setup complementary assets that are keys to their commercialization efforts as public goods

Teece (1986) considers two ways of accessing complementary assets. One is to build the complementary asset inside the organization and the other is to contract for access to the complementary asset from the supplier. OSS provides a third option: to establish an OSS project as a public good. The company weakens its appropriability regime and control of the complementary asset but it lowers development costs and the risks of entering into contractual agreements.

Companies lose control for access to the software but they do not lose the copyrights and trademarks on the open source project. Companies that establish the OSS project do not give up complete control of the complementary asset.

They can still influence the development and roadmap of the open source project. A company that establishes an OSS project can also influence other OSS projects that complement its market offer.

Table 11 Observations not Supported by Teece's Commercialization Framework

- Companies can setup complementary assets that are keys to their commercialization efforts as public goods
- Companies can convert specialized asset into generic complementary assets to decrease dependence on other critical specialized assets
- Need to identify risks of integrating critical specialized assets into specialized asset
- Assess the relationship between innovation and the market offer over time

- Use complementary asset as low-end product to seed demand for company's market offers

Companies can convert specialized asset into generic complementary assets to decrease dependence on other critical specialized assets

Teece (1986) does not deal with companies transforming complementary assets to reduce dependence on critical specialized complementary assets. Establishing OSS projects can reduce a company's dependence on other key complementary assets. This is the case with Apple. By establishing the Darwin project, Apple has turned a specialized asset (core operating system) into a generic asset. This generic asset, in turn, reduced Apple's dependence on another key specialized asset (PowerPC) it contracts from a supplier (IBM).

Need to identify risks of integrating critical specialized assets into specialized asset

Teece (1986) deals with who is going to profit from innovation: the innovator, imitators, and owner of complementary assets. The framework focuses on competing for the innovation. It does not deal with the risks of companies integrating critical specialized complementary assets owned by others and the possibility of an imitator acquiring those critical specialized assets.

MySQL integrates InnoDB's open source transaction engine to give MySQL database a much needed functionality. Oracle, a provider of a competing database, acquired InnoDB. Although, MySQL have contractual agreements with

InnoDB on integrating the transaction engine, there might be some difficulties when the contract negotiations come up in 2006.

Assess the relationship between innovation and the market offer over time.

Teece (1986) assume that the innovation continuously contributes a significant value to the market offer. Over time, the value of the innovation as part of the value of the market offer may get smaller. Companies can assess if they can protect the value of the innovation and convert the part it cannot protect into generic assets. Apple assessed that the value of the Macintosh is in Mac OS X and its user experience. The company can still protect the technology that delivers the user experience but not the core operating system. Apple separated what it can protect from what it cannot protect. What it can protect remained a critical specialized asset and what it cannot protect was converted into a generic complementary asset.

Use complementary asset as low-end product to seed demand for company's other market offers

Teece (1986) deals with using complementary asset to commercialize an innovation into a market offer. The framework does not deal with using complementary assets to seed demand for a company's market offer. OSS can be used to seed demand for a company's proprietary offer. IBM and JasperSoft use the software from their OSS project to seed demand for their proprietary software. The OSS serves as entry-level products with less functionality than the

proprietary product. If the customers outgrow the functionalities of the entry-level product, they can upgrade to the proprietary software.

6 CONCLUSIONS, LIMITATIONS, AND FUTURE RESEARCH

6.1 Conclusions

This research examined how 12 companies that established OSS projects integrated the software from these projects into their market offers. Previous studies have identified different ways companies profit from OSS but not how the OSS is integrated into their market offers. This is the first study on companies that established OSS projects.

This research led to the following conclusions:

1. Companies use OSS projects to develop software that they integrate with proprietary assets they own to create market offers. Companies offer services such as testing and training, support, and subscriptions tied to OSS as well as products and applications that use the OSS.
2. OSS changes the customers' motive to license technology. Previously, customers license to access the technology and the owners of the technologies protect the IP by legal means. With OSS, the customers license the technology not for access but in order to protect their own IP.
3. Companies do not give up complete control of the OSS project and they influence the direction of the project to make sure that it aligns with the companies' objectives. Companies control the brands associated with the project. Competitors will have difficulties providing alternative offers without having to use the project's IP (i.e., brand).

4. OSS allows companies to do more than what they can do with strategic alliances, freeware, and standards. OSS allows companies to acquire external technology without the need for contracts, co-create value with the community and appropriate the value created by others. It also allows companies to create dominant standards and continue owning the rights to the technology that has been established as a dominant standard.
5. Existing commercialization frameworks are based on ability to protect IP and access to complementary assets. Companies access complementary assets by building them inside the company or contracting them from suppliers. With OSS, companies can set up complementary assets that are keys to their commercialization efforts as public goods. Companies can setup OSS projects as complementary assets and get access without the need for contracts. OSS lowers development cost and avoids the hazards of contracting for access to complementary assets. Commercialization frameworks need to be modified to take this into account.
6. OSS is a competitive environment and not a free environment. Companies establish OSS projects not to create free software but to achieve strategic objectives. Companies use OSS to create and appropriate value aside from reducing cost and shortening time to market. They use OSS to seed demand for proprietary products, offer services, and develop products and applications. Companies use OSS

to experiment with new technologies and product concepts. OSS project server as proving grounds for new technologies and when these technologies mature, the companies incorporate them into their market offers. Companies also prototype new market offers to gain clear and early insight on emerging markets and customer needs.

6.2 Limitations

The first limitation of this research is that it uses secondary sources of data. The research is not based on in depth interviews with companies representatives.

The second limitation of this research is that it examines the companies that established the OSS project and their market offers but not the OSS community. This was partly due to time and resource constraints because mature OSS projects from companies such as Red Hat, Apple, JBoss, and IBM have vast amounts of data such as those in their mailing lists, forums, and documentations. It takes time to collect and digest useful information from this sea of data. Other companies in the sample were young and are in the process of establishing a community.

A third limitation is that there was no database of companies that established OSS projects. Some companies may not have been included in the sample that

provides another example of how to integrate OSS into their market offers and how OSS is used to improve their competitive positions.

6.3 Opportunities for Future Research

An OSS project is a valuable complementary asset that the company needs to manage and protect. This study provides two opportunities for future research:

1. Examine how companies must manage projects that they established.

Companies have to create a vibrant community to benefit from resources and capabilities that are outside the boundaries of the firm. What are the things companies need to do to create and maintain this community?

2. Examine how competitors gain control of an OSS project. Access to the technology of an OSS project poses no problems for imitators. However, the company that established the project still controls and owns the copyrights and brands of the project. Can projects that have successfully forked offer insights on how imitators can wrest control of an OSS project?

7 REFERENCES

Anderson, J., G. Carpenter, and J. Narus. 2001. Managing Market Offerings in Business Markets. *Kellogg on Marketing*. John Wiley & Sons. 330-365.

Baghai, M., C. Coley, and D. White. 2000. *The Alchemy of Growth: Practical Insights for Building the Enduring Enterprise*. Cambridge, Mass.: Perseus Publishing.

Bailetti, A.J. and J.R. Callahan. 1995. Managing Consistency between Product Development and Public Standards Evolution. *Research Policy*. 24: 913-931.

Berlecon Research. 2002. FLOSS Final Report – Part 3: Basics of Open Source Software Markets and Business Models. http://www.opensource.gr/content/modules/downloads/Basics_of_OSS_Markets_Part_III.pdf (accessed 7 July 2005).

Bonaccorsi, A. and C. Rossi. 2003. Why Open Source Software can Succeed. *Research Policy*, 32(7): 1243 – 1258.

Bonaccorsi, A. and C. Rossi. 2004. Comparing Motivations of Individual Programmers and Firms to take part in the Open Source Movement. From Community to Business. <http://opensource.mit.edu/papers/bnaccorsirossimotivationlong.pdf> (accessed 23 February 2005).

Bonaccorsi, A. and C. Rossi. 2005. Licensing schemes in the production and distribution of Open Source software. An empirical investigation, <http://opensource.mit.edu/papers/bnaccorsirossilicense.pdf> (accessed 28 August 2005).

Christensen, C., and P. Carlile. 2005. The Cycles of Theory Building in Management Research. <http://www.innosight.com> (accessed 25 November 2005).

Dahlander, L. 2004. Appropriating the Commons: Firms in Open Source Software. <http://opensource.mit.edu/papers/dahlander.pdf> (accessed 28 August 2005).

Dahlander, L. and M. Magnusson. 2005. Relationships Between Open Source Software Companies and Communities: Observations from Nordic Firms. *Research Policy*. 34: 481-493.

Eisenhardt, K. 1989. Building Theories from Case Study Research. *Academy of Management Review*. 14(4): 532-550.

Evans, D., B. Reddy, and A. Layne-Farrar. 2001. Open Source Software. http://www.microsoft.com/japan/sharedsource/Articles/NERA_OS_%20whitepaper.pdf (accessed 27 July 2005).

Franck, E. and C. Jungwirth. 2003. Reconciling Rent-Seekers and Donators – The Governance Structure of Open Source. *Journal of Management and Governance*. 7(4): 401

Franke, N. and E. Von Hippel. 2003. Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software. *Research Policy*. 32(7): 1199-1215.

Funk, J. 2003. Standards, Dominant Designs and Preferential Acquisition of Complementary Assets Through Slight Information Advantages. *Research Policy*. 32: 1325-1341.

- Gacek, C. and B. Arief, 2004. The Many Meanings of Open Source. IEEE Software. January 2004: 34 – 40.
- Gans, J.S. and S. Stern. 2003. The Product Market and the Market for “Ideas”: Commercialization Strategies for Technology Entrepreneurs. Research Policy. 32(2): 333-350.
- Grant, R. 1991. Porter’s Competitive Advantage of Nations: An Assessment. Strategic Management Journal. 12(7): 535-548.
- Gulati, R. 1998. Alliances and Networks. Strategic Management Journal. 19: 293-317.
- Hannah, L. and P. Williamson. 1990. Realising the Potential of Scale and Scope: A review Article. Business Strategy Review. 1(3): 91-97.
- Hars A., and S. Ou. 2002. Working for free? Motivations for participating in Open Source Projects. International Journal of Electronic Commerce. 6: 25-39.
- Haruvy, E. and A. Prasad. 2001. Optimal Freeware Quality in the Presence of Network Externalities: An Evolutionary Game Theoretical Approach. Journal of Evolutionay Economics. 11: 231-248.
- Haruvy, E. and A. Prasad. 2005. Freeware as a Competitive Deterrent. Information Economics and Policy. 17: 513-534.
- Hecker, F. 2000. Setting up Shop: The Business of Open-Source Software. <http://www.hecker.org/writings/setting-up-shop.html> (accessed 7 February 2005).

Henkel, J. 2003. Open Source Software from Commercial Firms – Tools, Complements, and Collective Invention. http://www.inno-tec.bwl.uni-muenchen.de/forschung/henkel/OSS_JHenkel_2003-05.pdf (accessed 7 February 2005).

Henkel, J. 2004. Patterns of Free Revealing – Balancing Code Sharing and Protection in Commercial Open Source Development. <http://opensource.mit.edu/papers/henkel2.pdf> (accessed 23 February 2005).

Hertel, G., S. Niedner, and S. Herrmann. 2003. Motivation of Software Developers in the Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel. *Research Policy*. 32(7): 1159-1177.

Hohmann, L. 1989. Process Control in Standards Bodies. *IEEE Communications Magazine*. September. 59-61.

Iansiti, M. and J. West. 1997. Technology Integration: Turning Great Research into Great Products. *Harvard Business Review*. May-June. 69-79.

Kotler, P. and R. Turner. 1995. *Managing Products, Lines, Brands, and Packaging, Marketing Management – Analysis, Planning, Implementation, and Control*. Prentice Hall Canada. 420-449.

Krishnamurthy, S. 2002. Cave or Community? An Emperical Examination of 100 Mature Open Source Projects. http://www.firstmonday.org/issues/issue7_6/krishnamurthy (accessed 23 February 2005).

Koenig, J. (2004), *Seven open source business strategies for competitive advantage*.

<http://management.itmanagersjournal.com/article.pl?sid=04/05/10/2052216>

(accessed 23 February 2005).

Lakhani, K.R., B. Wolf, J. Bates, and C. DiBona. 2002. The Boston Consulting Group Hacker Survey. Presented at the O'Reilly Open Source Conference, San Diego, CA.

Lakhani, K., and E. von Hippel. 2003. How Open Source Software Works: "Free" User-to-User Assistance. *Research Policy*. 32(6): 923-943.

Lambe, C.J. and R.E. Spekman. 1997. Alliance, External Technology Acquisition, and Discontinuous Technological Change. *Journal of Product Innovation Management*. 14: 102-116.

Lerner, J., and J. Tirole. 2000. The Simple Economics of Open Source. Working Paper No. 7600, National Bureau of Economic Research, Cambridge, MA.

Lerner, J., and J. Tirole. 2001. The Open Source Movement: Key Research Questions. *European Economic Review*. 45: 819-826.

Lerner, J. and J. Tirole, 2002. The Scope of Open Source Licensing. MIT Working Paper. <http://opensource.mit.edu/papers/lernertirole2.pdf> (accessed 23 February 2005).

Levitt, T. 1980. Marketing Success Through Differentiation – of Anything. *Harvard Business Review*. January-February. 58: 83-91.

Liao-Troth, M. and T.L. Griffith. 2002. Software, Shareware and Freeware: Multiplex Commitment to an Electronic Social Exchange System. *Journal of Organizational Behavior*. 23: 635-653.

Liebeskind, J.P. 1996. Knowledge, Strategy, and the Theory of the Firm. *Strategic Management Journal*. 17: 93-107.

Mowery, D.C., J.E. Oxley, and B.S. Silverman. 1996. Strategic Alliances and Interfirm Knowledge Transfer. *Strategic Management Journal*. 17: 77-91.

Nambisan, S. 2002. Complementary Product Integration by High-Technology New Ventures: The Role of Initial Technology Strategy. *Management Science*. 48 (3): 382-398.

Nissila, J. 2004. Towards a Better Understanding of Open Source Business Models, <http://w3.msi.vxu.se/users/per/IRIS27/iris27-1202.pdf> (accessed 23 February 2005).

O'Mahony, S. 2003. Guarding the Commons: How Community Managed Projects Protect Their Work. *Research Policy*. 32: 1179-1198.

Open Source Initiative, 2005, <http://www.opensource.org> (accessed 27 July 2005).

Pal, N. and T.R. Madanmohan. 2002. Competing on Open Source: Strategies and Practice. <http://opensource.mit.edu/papers/madanmohan.pdf> (accessed 23 February 2005).

Pearson, H. 2000. Open Source – The Death of Proprietary Systems? *Computer Law and Security Report*. 16(3): 151-156.

Porter, M. 1990. The Competitive Advantage of Nations. *Harvard Business Review*. April-May. 79-93.

Porter, M. and M. Kramer. 2002. The Competitive Advantage of Corporate Philanthropy. *Harvard Business Review*. 80(12): 56-68.

Raymond, E.S. 1999a. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol: O'Reilly.

Raymond, E.S. 1999b. *The Magic Cauldron*.

<http://www.catb.org/~esr/writings/magic-cauldron/magic-cauldron.html> (accessed 22 February 2005).

Raymond, E.S. 1999c. *Halloween I: Annotated Version*.

<http://www.opensource.org/halloween/halloween1.php> (accessed 22 February 2005).

Robertson, T.S. and H. Gatignon. 1998. Technology Development Mode: A Transaction Cost Conceptualization. *Strategic Management Journal*. 19(6): 515-531.

Rossi, C. and A. Bonaccorsi. 2004. Intrinsic Motivations and Profit-oriented Firms in Open Source Software. Do Firms Practice What They Preach?

http://freesoftware.mit.edu/papers/rossi_motivations.pdf (accessed 23 February 2005).

Rothaermel, F. 2001. Incumbent's Advantage Through Exploiting Complementary Assets Via Interfirm Cooperation. *Strategic Management Journal*. 22: 687-699.

- Schilling, M. and H.K. Steensma. 2002. Disentangling the Theories of Firm Boundaries: A Path Model and Empirical Test. *Organization Science*. 13(4): 387-401.
- Sengupta, S. 1998. Some Approaches to Complementary Product Strategy. *Journal of Product Innovation Management*. 15: 352-367.
- Severance, C. 1995. The Value of the Formal Standards Process. *Computer*. 28(8): 82-83.
- Sheriff, M. 2001. A Framework for Standardization in Telecommunications and Information Technology. *IEEE Communications Magazine*. 39: 94-100.
- Sheriff, M. and D. Sparrell. 1992. Standards and Innovation in Telecommunications. *IEEE Communications Magazine*. 7: 22-29.
- Stewart, K., A. Ammeter, and L. Maruping. 2005. A Preliminary Analysis of the Influences and Organizational Sponsorship on Success in Open Source Projects. *Proceedings of the 38th Hawaii International Conference on System Sciences – 2005*. 1-10.
- Teece, D. 1986. Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing, and Public Policy. *Research Policy*. 15: 285-305.
- Tripsas, M. 1997. Unraveling the Process of Creative Destruction: Complementary Assets and Incumbent Survival in the Typesetter Industry. *Strategic Management Journal*. 18: 119-142.

- Valimaki, M. 2003. Dual Licensing in Open Source Software Industry.
<http://opensource.mit.edu/papers/valimaki.pdf> (accessed 23 February 2005).
- Valimaki, M. and V. Oksanen. 2005. The Impact of Free and Open Source Licensing on Operating System Software Markets. *Telematics and Informatics*. 22: 97-110.
- Vanhaverbeke, W., G. Duysters, and N. Noordhaven. 2002. External Technology Sourcing Through Alliances or Acquisitions: An Analysis of the Application-Specific Integrated Circuits Industry. *Organization Science*. 13(6): 714-733.
- van Wendel de Joode, R., J.A. de Bruijn, and M.J.G. van Eeten. 2002. Protecting the Virtual Commons.
[http://dlc.dlib.indiana.edu/archive/00001075/00/Protecting the Virtual Commons .pdf](http://dlc.dlib.indiana.edu/archive/00001075/00/Protecting_the_Virtual_Commons.pdf) (accessed 23 February 2005).
- Von Hippel, E. and G. Von Krogh. 2003. Open Source Software and the “Private-collective” Innovation Model: Issues for Organization Science. *Organization Science*. 14(2): 209.
- Von Krogh, G. and E. Von Hippel. 2003. Special Issue on Open Source Software Development. *Research Policy*. 32(7): 1149-1157.
- Von Krogh, G., Spaeth, S., and K.R. Lakhani. 2003. Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy*, 32(7): 1217-1241.
- West, J. 2003. How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies. *Research Policy*. 32(7): 1259-1285.

West, J. and S. Gallagher. 2004. Key Challenges of Open Innovation: Lessons from Open Source Software.

http://www.cob.sjsu.edu/west_j/Papers/WestGallagher2004.pdf (accessed 23 February 2005).

West, J. and S. O' Mahony. 2005. Contrasting Community Building in Sponsored and Community Founded Open Source Projects.

<http://opensource.mit.edu/papers/westomahony.pdf> (accessed 3 December 2005)

Wichmann, T. 2002. FLOSS Final Report – Part 2: Free/Libre Open Source Software: Survey and study – Firms' Open Source Activities: Motivations and Policy Implications. Berlecon Research.

http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf (accessed 7 July 2005)

Wilcox, J.. 2000. IBM to spend \$1 billion on Linux in 2001. CNET News.com.

<http://news.cnet.com/news/0-1003-200-4111945.html> (accessed 23 February 2005)

Yetton, P., J. Craig, J. Davis, and F. Hilmer. 1992. Are Diamonds a Country's Best Friend? A Critique of Porter's Theory of National Competition as Applied to Canada, New Zealand and Australia. *Australian Journal of Management*. 17(1): 89-119.

Yin, R. 2003. *Case Study Research: Design and Methods*. 3rd Edition. Thousand Oaks, California. Sage Publications.

APPENDIX

Appendix A List of Companies

Company	OSS Project	Start	Comments
Activegrid www.activegrid.com	Activegrid	2004	
Alfresco www.alfresco.org	Alfresco	2005	
Alkalcon www.alkalcon.com	Open CMS www.opencms.org	2000	
Apple www.apple.com developer.apple.com/darwin	Darwin	1999	
Compiere www.compiere.org	Compiere	1999	
Computer Associates www.ca.com opensource.ca.com/projects/ingres	Ingres	2004	
Covalent www.covalent.com		1998	

db4objects www.db4o.com	db4o	2000	
Digium www.digium.com www.asterisk.org	Asterisk	2001	
Gluecode	Apache Geronimo		Bought by IBM – Apache Geronimo
Greenplum www.greenplum.com	Bizgres	2005	
HP opensource.hp.com/	LaserJet Drivers		
IBM www.ibm.com	Eclipse	2001	
InnoDB	InnoDB	2001	Bought by Oracle – Oct 2005
Jaspersoft www.jaspersoft.com	Jasper Reports	2001	
JBoss www.jboss.com	JBoss	1999	
Miro www.miro.com.au	Mambo	2000	
MySQL	MySQL	1995	

www.mysql.com			
Pingtel www.pingtel.com	SIPxchange	2004	
RealNetworks www.realnetworks.com www.helixcommunity.org	Helix	2002	
Red Hat www.redhat.com	Fedora	2003	
Sendmail www.sendmail.com	Sendmail	1998	
SGI oss.sgi.com/projects/ogl-sample	OpenGL SI	1999	
Sleepycat www.sleepycat.com	Berkeley DB	1996	
Spikesource www.spikesource.com		2005	
Sourcefire www.sourcefire.com	Snort	1998	Bought by Checkpoint - 2005
SugarCRM www.sugarcrm.com	SugarCRM	2004	

Sun www.opensolaris.org	Open Solaris	2005	
Sun www.openoffice.som	Open Office	1999	
Trolltech www.trolltech.com	Qt	1998	
Xensource www.xensource.com	Xensource	2005	
Zend	Zend Engine	1999	
Zimbra www.zimbra.com	Zimbra	2005	
Zope	Zope	1998	

Appendix B How OSS is Integrated with Market Offers

Company	How extensive is the integration between proprietary assets and OSS?	Can OSS contribute significantly to the company's market offer?
Apple		
Macintosh Personal Computer	Comprehensive Integration OSS and proprietary software (i.e., hardware drivers and application framework) are comprehensively integrated internally into a module (i.e., operating system) which is placed at the interface between what the company can protect and the hardware (i.e., microprocessor) it purchases; the interface module is a small part of the product	No
RealNetworks		
RCSL	Minimal Integration Commercial licenses allow customers to keep applications based on Helix proprietary. OSS and license are minimally integrated.	Yes
RBN	Minimal Integration	No

<p>OSS and proprietary software are comprehensively integrated internally into an application module which is minimally integrated with the network that delivers the hosted service.</p>	
<p>Real</p>	<p>Minimal Integration</p> <p>OSS and proprietary software are comprehensively integrated internally into a module which is minimally integrated with the network that delivers the sale of content.</p>	<p>No</p>
<p>Media Server</p>	<p>Comprehensive Integration</p> <p>OSS and proprietary software providing advanced networking functionalities are comprehensively integrated internally into a module (i.e., application runs with the OSS) which is what the company sells.</p>	<p>Yes</p>
<p>Red Hat</p>		
<p>Training</p>	<p>Minimal Integration</p> <p>OSS is used to anchor training services. Company owns certificate given to</p>	<p>Yes</p>

	those that pass skills testing. OSS and certificate are minimally integrated.	
HW/SW Certification	Minimal Integration OSS is tested to assess compatibility with complementary products from partners using the company's test suite. Company provides compatibility certification when partner's products pass required tests. OSS and certificate are minimally integrated.	Yes
Red Hat Network	Comprehensive Integration OSS is used to anchor subscription service. OSS is comprehensively integrated with infrastructure that delivers update, upgrade, monitoring, and management functionalities.	Yes
JBoss		
Professional Certification	Minimal Integration OSS is used to anchor training services. Company owns certificate given to those that pass skills testing. OSS and certificate are minimally integrated.	Yes
Solutions Certification	Minimal Integration	Yes

<p>OSS is tested to assess compatibility with complementary products from partners using the company's test suite. Company provides compatibility certification when partner's products pass required tests. OSS and certificate are minimally integrated.</p>	
<p>Subscription</p>	<p>Comprehensive Integration</p> <p>OSS is used to anchor subscription service. OSS is comprehensively integrated with infrastructure that delivers update, upgrade, monitoring, and management functionalities.</p>	<p>Yes</p>
<p>IBM</p>		
<p>WebSphere Studio</p>	<p>Comprehensive Integration</p> <p>OSS is comprehensively integrated with proprietary software that provides enterprise-level project management, advanced Java development, and web-services support.</p>	<p>Yes</p>
<p>DB2</p>	<p>Minimal Integration</p> <p>OSS2 (i.e., Cloudscape) is an entry-level product for the company's high-end proprietary product (i.e., DB2). OSS1 (i.e., Eclipse) facilitates ease of migration</p>	<p>Yes</p>

	from Cloudscape to DB2.	
SGI		
Trademark License	Minimal Integration Customers use OpenGL Sample Implementation as foundation for hardware drivers. Customers buy license to use OpenGL trademark. OSS and license are minimally integrated.	Yes
MySQL		
Commercial License	Minimal Integration Minimal Integration Commercial licenses allow customers to keep applications based on MySQL proprietary. OSS and license are minimally integrated.	Yes
MySQL Network	Comprehensive Integration OSS is used to anchor subscription service. OSS is comprehensively integrated with infrastructure that delivers update and upgrade.	Yes
Training	Minimal Integration	Yes

<p>OSS is used to anchor training services. Company owns certificate given to those that pass skills testing. OSS and certificate are minimally integrated.</p>	
Digium		
<p>Telecom Cards</p>	<p>Comprehensive Integration</p> <p>Hardware is comprehensively integrated with OSS to connect to traditional telecommunications networks.</p>	<p>Yes</p>
Sourcefire		
<p>Defense Center</p>	<p>Comprehensive Integration</p> <p>OSS and proprietary products create a system to form the customer's security infrastructure. OSS uses Snort Rules (i.e. black circle) to detect and respond to security threats.</p>	<p>Yes</p>
<p>VRT Rules</p>	<p>Comprehensive Integration</p> <p>OSS is comprehensively integrated with Vulnerability Research Team (VRT) Certified Rules (i.e., white circle). OSS uses VRT Certified Rules to detect and respond to security threats.</p>	<p>Yes</p>
<p>RNA</p>	<p>Comprehensive Integration</p>	<p>Yes</p>

<p>OSS and proprietary products create a system to form the customer's security infrastructure. OSS uses Snort Rules (i.e. black circle) to detect and respond to security threats.</p>	
<p>JasperSoft</p>		
<p>JasperDecisions</p>	<p>Comprehensive Integration</p> <p>OSS (i.e., JasperReports) is an entry-level product for the company's high-end proprietary product (i.e., JasperDecisions). When customer upgrades, OSS is integrated into proprietary product.</p>	<p>Yes</p>
<p>SpikeSource</p>		
<p>Support</p>	<p>Minimal Integration</p> <p>OSS from other projects are integrated and tested to create a certified stack and add management capabilities. Company provides support for the stack.</p>	<p>Yes</p>
<p>ActiveGrid</p>		
<p>Support</p>	<p>Minimal Integration</p>	<p>Yes</p>

OSS from other projects are integrated and tested to create a certified stack and add scalability capabilities. Company provides support for the stack.	

Appendix C Insights Using Commercialization Frameworks

Company	Insights from Commercialization Frameworks	
Apple		
Macintosh Personal Computer	<p>Can protect user experience and application environment with effective legal mechanisms</p> <p>Cannot protect all components of the generic product</p> <p>Paradigmatic stage</p>	<p>OSS is a critical, co-specialized asset</p> <p>According to current commercialization frameworks, company should build the co-specialized asset.</p> <p>OSS:</p> <ul style="list-style-type: none">• Provides an additional alternative to (i) own or (ii) buy the required co-specialized asset• Lowers the cost of developing a clear interface between what the company can protect and what it buys from large suppliers• Reduces supplier lock-in. OSS facilitated the switch to a hardware platform better suited to satisfy new customer needs (i.e., switched from Motorola to Intel due to customer demands for mobile computing required a better microprocessor and greater manufacturing capability)• Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down

RealNetworks		
RCSL	<p>Can't protect the OSS, can protect the license</p> <p>Pre-paradigmatic stage (i.e., fighting with Microsoft to set the dominant design)</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should become intimately coupled to the market so user needs can impact design.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Provides the generic product (i.e., OSS plus commercial license) the firm sells • Allows the company to charge for licenses so customers can protect their own IP • Eliminates the need to pay to develop or acquire the software • Provides way to establish the dominant design during the pre paradigmatic stage • Eliminates the cost of porting software into different hardware platforms; suppliers of hardware can incorporate the OSS into their own platforms
RBN	Can't protect application module or the network	OSS is a critical, specialized asset for the application module.

	<p>that delivers the service</p> <p>Pre paradigmatic stage, both the OSS and the application module are at the pre paradigmatic stage</p>	<p>According to current commercialization frameworks, company should become intimately coupled to the market so user needs can impact design.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Lowers the cost of developing the application module • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down • Captures value for the service
Real	<p>Can't protect technology</p> <p>Pre paradigmatic stage, both the OSS and the application module are at the pre paradigmatic stage</p>	<p>OSS is a critical, specialized asset for the application module.</p> <p>According to current commercialization frameworks, company should become intimately coupled to the market so user needs can impact design.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Lowers the cost of developing the application module • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down

		<ul style="list-style-type: none"> • Captures value for the service
Media Server	<p>Can't protect application module</p> <p>Pre paradigmatic stage</p>	<p>OSS is a critical, specialized asset</p> <p>According to current commercialization frameworks, company should become intimately coupled to the market so user needs can impact design.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Lowers the cost of developing the generic product • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down
Red Hat		
Training	<p>Can't protect certification service, particularly if large companies start certificate programs</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset, company owns certification process</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p>

		<p>OSS:</p> <ul style="list-style-type: none"> • Provides value to the certification service • Lowers the cost of acquiring customers • Eliminates the need to contract and pay to access the generic asset • Makes it easy to sub-contract training services to others (i.e., universities and colleges provide the training, company issues certificate)
HW/SW Certification	<p>Can't protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset, company owns test suite</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Provides value to company's test services • Eliminates the need to contract and pay to access the generic asset • Lowers the cost of acquiring customers • Adds value to the offers of the company's partners to their clients because of the certification

Red Hat Network	<p>Can't protect the technology used to deliver the service</p> <p>Paradigmatic stage</p>	<p>OSS is a critical, co specialized asset</p> <p>According to current commercialization frameworks, company should develop the co-specialized asset.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Captures value for the subscription service • Lowers the cost of developing the software that provides the value to the subscription service • Provides an additional alternative to (i) own or (ii) buy the required co specialized asset • Lowers the cost of acquiring customers • Avoids the hazards of contractual agreements for specialized assets. <p>These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down</p>
JBoss		
Professional Certification	<p>Can't protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and</p>

		<p>conditions upon which the required complementary assets can be accessed</p> <p>OSS;</p> <ul style="list-style-type: none"> • Provides value to the certification service • Lowers the cost of acquiring customers • Eliminates the need to contract and pay to access the generic asset • Makes it easy to sub-contract training services to others (i.e., universities and colleges provide the training, company issues certificate)
Solutions Certification	<p>Can't protect test suite</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset, company owns test suite</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Provides value to company's test services • Eliminates the need to contract and pay to access the generic asset • Lowers the cost of acquiring customers • Adds value to the offers of the company's partners to their clients because of the certification

Subscription	<p>Can't protect the technology used to deliver the service</p> <p>Paradigmatic stage</p>	<p>OSS is a critical, co specialized asset</p> <p>According to current commercialization frameworks, company should develop the co-specialized asset.</p> <p>OSS:</p> <ul style="list-style-type: none"> • Captures value for the subscription service • Lowers the cost of developing the software that provides value to the subscription service • Eliminates the need to own the co-specialized asset • Lowers the cost of acquiring customers • Avoids the hazards of contractual agreements for specialized assets. <p>These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down</p>
IBM		
WebSphere Studio	<p>Can't protect the technology</p> <p>Paradigmatic stage. Has established plug-in</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p>

	architecture as dominant design for visual development environments	<p>OSS:</p> <ul style="list-style-type: none"> • Has allowed company to establish dominant design • Eliminates the need to contract and pay access to generic asset
DB2	<p>Can protect technology of high-end database</p> <p>Database technology is in paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Has allowed company to provide entry-level product (i.e., Cloudscape) to seed demand for its enterprise-level proprietary product (i.e., DB2) • Eliminates the need to contract and pay access to generic asset
SGI		
Trademark License	<p>Can protect technology and trademark</p> <p>3D graphics technology is in paradigmatic stage.</p>	<p>OSS is a critical, specialized asset</p> <p>According to current commercialization frameworks, company should develop specialized asset.</p>

		<p>OSS:</p> <ul style="list-style-type: none"> • Provides the generic product (i.e., OSS plus trademark license) the firm sells • Provides a way for company to maintain dominant standard for 3D graphics technology • Eliminates the cost of porting software into different hardware platforms; suppliers of hardware can incorporate the OSS into their own platforms
MySQL		
Commercial License	<p>Can't protect the OSS, can protect the license</p> <p>Database technology is in paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Provides the generic product (i.e., OSS plus commercial license) the firm sells • Allows the company to charge for licenses so customers can protect their own IP

		<ul style="list-style-type: none"> Allows the company to integrate another OSS owned by another organization (i.e., InnoDB) into their product
MySQL Network	<p>Can't protect the technology used to deliver the service</p> <p>Paradigmatic stage</p>	<p>OSS is a critical, co specialized asset</p> <p>According to current commercialization frameworks, company should develop the co-specialized asset.</p> <p>OSS:</p> <ul style="list-style-type: none"> Captures value for the subscription service Lowers the cost of developing the software that provides value to the subscription service Eliminates the need to own the co-specialized asset Lowers the cost of acquiring customers Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down
Training	<p>Can't protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and</p>

		<p>conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Provides value to the certification service • Lowers the cost of acquiring customers. • Eliminates the need to contract and pay to access the generic asset • Makes it easy to sub-contract training services to others (i.e., partners, company issues certificate)
Digium		
Telecom Cards	<p>Can protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Develops complement and seed demand for its hardware • Lowers the cost of acquiring customers. Customers need the hardware if they want to connect to traditional telecommunications networks. • Eliminates the need to contract and pay to access the generic asset

Sourcefire		
Defense Center	Can protect technology	OSS is a critical, co-specialized asset
	Paradigmatic stage	<p>According to current commercialization frameworks, company should develop the co-specialized asset</p> <p>OSS:</p> <ul style="list-style-type: none"> • Develops complement and seed demand for proprietary product • Community develops content critical (i.e., Snort Rules) to the software • Provides an additional alternative to (i) own or (ii) buy the required co-specialized asset • Lowers the cost of acquiring customers. Customers can install OSS but need the proprietary product if they want real-time view of security events on their network. • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down
VRT Rules	Cannot protect Certified Rules	<p>OSS is a critical, specialized asset</p> <p>According to current commercialization frameworks, company should develop</p>

	Paradigmatic stage	<p>the specialized asset</p> <p>OSS:</p> <ul style="list-style-type: none"> • Develops complement and seed demand for VRT Rules • Provides an additional alternative to (i) own or (ii) buy the required co-specialized asset • Lowers the cost of acquiring customers. Customers can install OSS but need the VRT Certified Rules for tested and vested rules that detect and protect from new security threats • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down
RNA	<p>Can protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a critical, co-specialized asset</p> <p>According to current commercialization frameworks, company should develop the co-specialized asset</p> <p>OSS:</p> <ul style="list-style-type: none"> • Develops complement and seed demand for proprietary product • Community develops content critical (i.e., Snort Rules) to the software

		<ul style="list-style-type: none"> • Provides an additional alternative to (i) own or (ii) buy the required co-specialized asset • Lowers the cost of acquiring customers. Customers can install OSS but need the proprietary product if they want real-time view of security events on their network. • Avoids the hazards of contractual agreements for specialized assets. These contracts require commitments of irreversible investments which will be valueless if the contractual relationship breaks down
JasperSoft		
JasperDecisions	<p>Can protect technology</p> <p>Paradigmatic stage</p>	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Has allowed company to provide entry-level product (i.e., JasperReports) to seed demand for its enterprise-level proprietary product (i.e., JasperDecisions) • Eliminates the need to contract and pay access to generic asset

SpikeSource		
Support	Cannot protect technology Paradigmatic stage	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p> <p>OSS:</p> <ul style="list-style-type: none"> • Eliminates the need to contract and pay to access the generic assets. • Has allowed the company to integrate and certify software from other OSS projects • OSS has allowed the company to appropriate value through support services
ActiveGrid		
Support	Cannot protect technology Paradigmatic stage	<p>OSS is a generic asset</p> <p>According to current commercialization frameworks, company should contract to access the generic asset, commercial success swings to the terms and conditions upon which the required complementary assets can be accessed</p>

		<p>OSS:</p> <ul style="list-style-type: none">• Eliminates the need to contract and pay to access the generic assets.• Has allowed the company to integrate and certify software from other OSS projects• OSS has allowed the company to appropriate value through support services
--	--	---

Appendix D How OSS Improves Competitive Environment

Company	Factor Conditions	Demand Conditions	Context for Strategy and Rivalry	Related and Supporting Industries
Apple	- hires Jordan Hubbard, former lead of FreeBSD project	- option to sell Mac OS X independently from hardware	- increased support for Intel microprocessor without contracting for development	- reduced dependence on supplier of microprocessor
RealNetworks	- suppliers and partners hire talent from Helix community	- provide different market offers to different market segments	- customers purchase commercial license when they cannot abide by open source license - fights with Microsoft to set dominant standard for content distribution	- companies port Helix in their own platforms themselves
JBoss	- hires top three developers of an OSS project	- customers ask JBoss to provide support on other OSS projects	- other companies cannot use JBoss' project related brands in their market offers - implementation of technologies adopted in J2EE standard	- tries to influence other OSS projects
Red Hat	- use project as proving		- does not allow others to use	- tries to influence other

	ground for new technology		Fedora trademarks in their distributions	OSS projects
Digium	- community develops complement to company's hardware	- creates demand for company's hardware product		
IBM	- Eclipse based tools eases migration from Cludscape to DB2	- Cludscape seeds demand for company's DB2 product	- Eclipse established dominant design for visual development tools environment	
SGI	- community provides support to the Linux platform		- maintain dominant standard for 3D graphics	
MySQL	- incorporated critical functionality from another OSS project owned by InnoDB		- customers buy commercial licenses if they cannot abide by the open source license	
Sourcefire	- community contributes Snort Rules	- creates demand for complementary products such as Defense Center and RNA		- uses community to detect and respond to security threats quickly and efficiently
ActiveGrid			- integrates software from other OSS projects to create a stack	

			(LAMP) without contracting	
JasperSoft		- uses JasperReports to seed demand for JasperDecisions		
SpikeSource			- uses software from other OSS projects to create a stack (LAMP) without contracting	

Appendix E Benefits and Risks of Establishing OSS Projects

Company	Benefits of establishing OSS projects	Risks of establishing OSS projects
Apple	<ul style="list-style-type: none"> Pool expertise with community and receive tips on design of core operating system from developers who have spent a long time working on operating systems. 	<ul style="list-style-type: none"> Risk that community disappears. Developers are starting to become frustrated due to difficulties in contributing to the project.
RealNetworks	<ul style="list-style-type: none"> Charge for commercial license 	
Red Hat	<ul style="list-style-type: none"> Sits on board of other OSS projects and try to influence the project. 	
JBoss	<ul style="list-style-type: none"> Choose license that is developer and business friendly 	<ul style="list-style-type: none"> Risk that code contributed by community infringes on IP of others. Company provides indemnification to customers.
IBM		<ul style="list-style-type: none"> Risk that customers do not upgrade to company's proprietary product, DB2.
SGI	<ul style="list-style-type: none"> Charge for trademark license 	
MySQL	<ul style="list-style-type: none"> Charge for commercial license 	<ul style="list-style-type: none"> Risk that when depending on other OSS project owned by another company, that company gets acquired by competitor. MySQL depends on InnoDB's OSS software but InnoDB was acquired by Oracle.

Digium		<ul style="list-style-type: none"> • Risk that competitors develops competing hardware that works with the OSS.
Sourcefire		
JasperSoft		<ul style="list-style-type: none"> • Risk that customers do not upgrade to company's proprietary product, JasperDecisions.
SpikeSource		<ul style="list-style-type: none"> • Risk that community does not develop. OSS project is still young and company is attempting to attract developers.
ActiveGrid		<ul style="list-style-type: none"> • Risk that community does not develop. OSS project is still young and company is attempting to attract developers.