Strategic dimensions for open source competition
- Appropriating value in a weak appropriability regime

Christopher Fabritius

Supervisor: Jens Frøslev
Institute: INO

Copenhagen Business School, December 2009
Abstract

Purpose Firstly, to place open source business models in a theoretical framework. Secondly, to identify generic complementary assets that are essential for the ability of open source firms to compete. Lastly, to relate the individual assets to the properties of open source licensing and software business models.

Design/methodology/approach An empirical approach that questions assumptions of existing theory. This is achieved through an examination of the mechanisms and unique characteristics of open source software, and how these influence business models with special regards to complementary assets. Using the findings from “Profiting from Technological Innovation” by David Teece and other key theorists to outline the interaction between complementary assets and available product markets. Finally, empirical data is used to identify and closely examine core complementary assets in weak appropriability regimes.

Findings Copyright legislation is used to ensure innovation through forcing openness of the technology in contrast to the traditional opposite approach. Teece can be used to place open source business models in a familiar framework, defining open source as a weak appropriability regime. Though subject to a peculiar set of rules regarding technology and innovation, open source business is fundamentally “business as usual” when approached from a strategic perspective. It is the constellation of complementary assets in combination with select product markets that enables the unique competitive position for players and defines business models.

Research implications The thesis provides a starting point for understanding the competitive position of a given open source business model, through an analysis of its complementary assets constellation and the selected product market.

Originality/value Modifications to existing theory. This thesis challenges Teece’s original assumptions in which protection of assets plays a central role. Within open source business models this assumption can be replaced in favour of an enforced emphasis on building complementary assets to capture value from technology innovations.

Keywords software industry, open source, Teece, weak appropriability regimes, open innovation, capturing value from technology innovations, complementary assets, business model, licenses, patents, lock-in, strategy, technology, innovation, master's thesis.
# Table of Contents

1 *Introduction* .......................................................................................................................... 1  
  1.1 Problem........................................................................................................................................... 1  
  1.2 Method............................................................................................................................................... 3  
  1.3 Structure.......................................................................................................................................... 4  

2 *Theory: Defining competitive dimensions of open source business* ..................................... 6  
  2.2 A Framework of Open Business Dynamics (Christensen, 2009)................................................. 10  
  2.3 Understanding Software................................................................................................................... 11  

3 *Open Source Software and Intellectual Property* ................................................................. 14  
  3.1 Understanding Open Source ............................................................................................................. 14  
  3.2 Open Source Licensing.................................................................................................................... 15  
  3.3 Open Source and Patents................................................................................................................... 21  
  3.4 The evolution of the GPL.................................................................................................................. 24  
  3.5 Aligning with open source licensing............................................................................................... 28  
  3.6 Corporate strategy and Open Source.............................................................................................. 30  
  3.7 Conclusion....................................................................................................................................... 32  

4 *Challenges for traditional software business models* ......................................................... 33  
  4.1 Traditional Business Models.......................................................................................................... 33  
  4.2 Challenges in today's competitive landscape................................................................................. 36  
  4.3 Consequences for strategy.............................................................................................................. 41  
  4.4 Conclusion....................................................................................................................................... 42  

5 *Open Source Business: Appropriating value in a weak appropriability regime* ................ 43  
  5.1 Examples of existing open source business models....................................................................... 44  
  5.2 The importance of complementary assets....................................................................................... 46  
  5.3 Identifying Complementary Assets in Open Source Business Models....................................... 48  
  5.4 Business Models with Open Source Software is “Business as Usual”....................................... 54  

6 *Conclusion*..................................................................................................................................... 55
1 Introduction

Conventional wisdom suggests that, in order to be competitive, a firm needs to pursue either a cost leadership or a differentiation strategy. It appears that some problems emerge when trying to explain how companies are able to maintain competitive advantages within the field of open source software.

1. Since source code primarily exists in the form of digitised information, the marginal costs associated with the distribution and production of an open source product is close to nil. With the increasing degree of internet availability as the major factor in levelling the playing field, the influence of differences in size and economic power between the players has little or no influence on the ability to compete. Thus the generic strategy of price leadership and the common assumption of economies of scale becomes virtually inapplicable.

2. There are differences between available open source licenses, but they share the common factor that intellectual property is by its very nature freely available and thus prohibits or reduces the advantages that can be gained by product differentiation.

At the same time, we can observe how numerous companies successfully compete and profit in today's market place, even though their business models are built around open source products. One example is Red Hat Inc., a Linux vendor, which in July 2009 was included in the S&P 500 index of NYSE ("Standard & Poor's Announces Change to U.S. Index," 2009)
1.1 Problem

In his 1986 article “Profiting from Technological Innovation”, David Teece applies a combination of transaction cost and resource-based approaches to explain the strategic mechanisms of competition based on technological innovations.

Given the license terms and processes surrounding open source software, some of the fundamental assumptions in the framework he presents are challenged. Throughout the thesis I will refer his framework for “Profiting from innovation” as PFI.

There is an assumption in the appropriability regime that the purpose of strategy and the configuration of complementary assets, is to restrict the access of others to the supposedly finite amount of value generated by the innovation. The strength of the appropriability regime determines the focus the firm must seek to when attempting to capture value from such.

While Chesbrough et al. examine how generic open source business models work, their overall focus is on the revenue models. Thus, they mainly bring the importance of communities forward and only briefly discuss the requirements for pursuing what they have termed “hybridization business models”.

My hypothesis is that firms basing their business models on open source, do so based on a combination of the product markets and complementary assets available to each individual firm. What I seek to accomplish in this paper is therefore to identify generic complementary assets, essential for the ability of open source firms to compete. Further, I seek to relate the individual assets with the properties of open source licensing and software business models.

The central questions for the thesis thus becomes:

**Which complementary assets are central for the establishment of a competitive open source company?**

**Do open source business models exhibit significant differences in the perspective applications of traditional business strategy?**
In order to answer these questions, I will examine how intellectual property relates to the possible business models that can be built on the basis of open source software, answering the following questions:

- *What defines open source software?*

- *How does open source licensing affect how technologies may be applied and distributed?*

- *What are the typical open source licenses, and how do they relate to the possible business models built upon them?*

- *How does software patents affect open source software and what is done to address them?*

- *How does open source software relate to Teece’s model for tight or weak appropriability regimes?*

### 1.2 Method

This paper has been written with the intention that academic readers should be able to digest and relate to the analysis regardless their degree of technical understanding. The focus will be on the dynamics introduced and the change of rules presented by open source software in relation to the foundation laid by Teece *(Teece, 1986)*. To achieve a logical flow in the argumentation the paper is presented in the following order:

- The theoretical foundation for understanding value capturing of technology innovations and their application in the software space.

- An examination of Open Source Software and the intellectual property constellations that challenge conventional business models.

- Empirical evidence on traditional software business models and a discussion on their current challenges

- New business models as a consequence of Open Source Software and the paradigm shift they impose on the competitive parameters of the software industry.

- A conclusion on how OSS business models can be explained within the theoretical framework given, and how some frameworks must be extended or modified to fit with the reality of the competitive landscape.
As a general rule, information and statements regarding individual companies' intentions and strategy are taken from the firm's own material (annual reports, websites etc.). They are treated with the characteristics of market signalling soundly in mind. To the extent that companies themselves are referenced as the data source, the information has been verified through additional research unless otherwise noted.

The empirical foundation for this paper is comprised by one part long personal interest and one part a technical foundation and professional experience with various open source projects including FreeBSD, Linux, Apache, MySQL and several others.

The empirical data of concrete business models found in real life, are central to the argumentation of this thesis, because they demonstrate an explanation gap in existing theory. Selected examples have therefore been chosen with great care to illustrate the issues present and in support of my main arguments.

1.3 Structure

Though the concept of open source has contributed to a fundamental shift in the foundations for software business models, empirical data demonstrates that this does not render open source incompatible with corporate strategy. It is a challenge to understand the implications of the openness of open source within the boundaries of existing theoretical frameworks.

To achieve this, a series of theoretical frameworks are analysed and combined to demonstrate how open source software is largely “business as usual”, and certain theoretical assumptions are challenged. The details of the logical flow in each chapter follows below:

In Chapter 2, "Theory: Defining competitive dimensions of open source business", I examine the central framework, presented by Teece that allows for companies to capture value from technology innovations. Special focus is given to the assumptions Teece uses to build the framework, which will become central to the main findings of this thesis. Part of Teece's assumptions are challenged to fit with the software industry business models and the influences of Open Source Software.
Further, I present works of Chesbrough, Appleyard and Rosenbloom that define and demonstrate a selection of Open Source Software business models, and suggest alternatives to Teece's principle of access restriction as a key element in value capture.

Christensen grants us a concept framework to define the ecosystem surrounding a company's core business and bridge the theoretical framework between Open Source Software and traditional proprietary software markets.

In support of this, West defines a framework for Open Innovation, in which he redefines the responsibility of companies’ Research & Development departments.

The chapter concludes with my definition of Open Source Software as a weak appropriability regime, and aligning the concepts of product markets and the business models to Teece's considerations regarding complementary assets in order to provide a method for theoretically understanding how value can be captured when innovations are open.

Chapter 3, "Open Source Software and Intellectual Property" focuses on establishing a basic understanding of software and open source software in particular. It moves on to a detailed discussion on software licensing, with special focus on GNU GPL licenses, a widely used family of licenses and a basis for many of the real life examples this thesis builds upon. An account is given for how these licenses influence the business model that can be built around them as well as how this fits in with corporate strategy.

Chapter 4, “Challenges for traditional software business models” looks at the challenges posed to traditional software business models by three fundamental trends: the increasing commoditisation of software, in part as a consequence of the continued proliferation of open standards, the lowering of entry barriers for technology innovations new entrants and the problems created by the increasing use of digital rights management's failure to effectively protect assets with conventional means for property protection.
Chapter 5, “Open Source Business: Appropriating value in a weak appropriability regime“ gives notable, real-life examples of different approaches to benefiting from open source business models. Open source software is placed in Teece’s theoretical framework posing challenges to select assumptions for a better fit with reality: the constellation of complementary assets in combination with the available product markets and business models replace the protection of assets as the focus for capturing value from technology innovations. The chapter also explores some of the core complementary assets that must be considered alongside the chosen product market, to establish competitive constellations of assets in open source business models.
2 Theory: Defining competitive dimensions of open source business

I order to define competitive dimensions of open source business, I will examine the mechanisms that result from the unique characteristics of open source software, and how these influence their business models.

Teece himself (Teece, 2006) mentions that, while the last two decades have shown work extending and polishing his original framework, there have been no attempts to replace it. Thus, it seems reasonable to assume that an understanding of Teece's findings is a reasonable point of departure for this thesis. From here, I supplement the theoretical perspective with insight into open source business models (Chesbrough & Appleyard, 2007), open innovation (West & Gallagher, 2006) and product markets (Christensen, 2009).

2.1.1 Profiting from open source innovation

There are important characteristics of open source software that might only be partially explained by the dynamics suggested by Teece's original article. Also there are mechanisms of open source that provide a significantly different rule set than assumed in Teece's original work. Therefore, we will have to delve into a closer examination of the underlying assumptions in order to find relevant extensions to build a useful theoretical apparatus for understanding open source business models.

In his 1986 article “Profiting from Technological Innovation” (Teece, 1986), David Teece applies a combination of transaction cost and resource-based approaches to explain the strategic mechanisms of competition based on technological innovations. Given the license terms and processes surrounding open source software, some of the fundamental assumptions in the “Profiting from innovation” (PFI) framework are challenged.
There is an assumption in the appropriability regime that the purpose of strategy and the configuration of complementary assets, is to restrict the access of others to the supposedly finite amount of value generated by the innovation. The strength of the appropriability regime determines the focus the firm must seek to when attempting to capture value from such. However, the fundamental approach of the mechanisms governing the extent to which value can be captured from innovation provides a useful guide for the dimensions that we must explore in order to evaluate the value capture mechanisms available to open source.

In the application of Teece's findings in this thesis, the focus will be the interaction between complementary assets and the available product markets.

This, in turn, is a consequence of the challenge that open source poses for the base assumption of Teece that capturing value from technological innovations require restriction of access to the technology through configuration of complementary assets, the choice of which are in part determined by the strength of the appropriability regime.

In open source, since the technology per definition is freely available to everyone, this focus on restriction of access to leverage the application of technology becomes less relevant. However, it is my hypothesis that the mechanisms and logic governing the establishment of a successful constellation of complementary assets to establish a competitive position remains sound.

The inherent concepts of the appropriability regime dictate that some of the linkages that Teece assumes are highly specialised become increasingly generic as open standards and open source proliferates.

Further, since source code can be said to be codified knowledge by its very nature, software increasingly appears unprotected or placed in a relatively weak appropriability regime. The relative inefficiency of patents combined with piracy challenges the foundations of business models based on the assumption of these mechanisms as efficient ways of securing value.

With regard to business models Teece retains his product-centric approach in describing the role of the business model in establishing the most efficient method for capturing value from an innovation (Teece, 2006).
As demonstrated by Cusumano (Cusumano, 2008) however, the pure product-oriented software business model is becoming increasingly rare as software companies typically apply some sort of combination of product- and service oriented business models to reduce risk and capture value from multiple sources.

This, in combination with the multitude of companies organised around implementation, consulting and customization of software demonstrate that significant value can be captured through network effects surrounding technologies.

2.1.2 Making money from “free”

An important insight for the analysis of open source business models is that “free open source software” refers to the freedom to use, change and distribute the software as described in chapter 3, and does not necessarily imply that software is distributed for free in a monetary sense.

Chesbrough and Rosenbloom's article on the role of the business model for capturing value from innovation provides a useful context for the continuing discussion on the construction of business models to appropriate value from innovation, not directly addressed by Teece in “Profiting from Innovation...” (Teece, 1986).

In my opinion, though Teece acknowledges this, he does not sufficiently take into account the unique characteristics of the individual firm's supply chains:

Having a differentiated (and hard to imitate) yet effective and efficient “strategic architecture” to an enterprise's business model is critical to success (Teece, 2006).

In relation to open source business models, there is a certain degree of merit in Teece's approach except that it relates to decisions surrounding technologies or projects rather that concrete individual business models surrounding it. My hypothesis is that the constellation of complementary assets available to any individual firm intrinsically provides a unique supply chain through the combination of available product markets and assets.
Chesbrough and Appleyard provide an important perspective for the understanding of open source business models. Instead of focusing only on the internal resources available to the firm, they argue that external resources may also contribute and create value even though these resources are not directly owned by the firm (Chesbrough & Appleyard, 2007).

They approach open source on the basis of a “public good” generating “network effects”, and seem to largely equate open source software contributions with the user-generated content of Youtube and Wikipedia.

They identify seven generic business models in open source, some of which are novel compared to traditional software: support, subscription, professional services, proprietary extensions, dual license, device, and community source.

These are broadly characterised by being, in some way direct revenue models, monetizing a product or service by means of direct sales.

> The ultimate role of the business model for an innovation is to ensure that the technological core of the innovation delivers value to the customer. (Chesbrough & Rosenbloom, 2002, p. 549)

If we are to be interpret this definition literally, however we must consider indirect appropriability approaches as well.

In “Challenges of open innovation: the paradox of firm investment in open-source software.” (West & Gallagher, 2006), the authors define a framework for open innovation with a specific focus on the role of the Research & Development resources.

In relation to this thesis, the goals of stimulating external spillovers in relation to external resources as well as the communities are especially interesting. Particularly the roles of the R&D resources in linking the company with the surrounding ecosystem(s) with regards to absorptive capacity in relation to new and the management of the knowledge flows with the community.

Of specific interest in relation to the discussion regarding the capture of value, the authors brings up the sale of complements as an example of how a product centric approach allows innovation in one product to be monetised through the sale of its complement (West & Gallagher, 2006).
2.2 A Framework of Open Business Dynamics (Christensen, 2009)

In this working paper, the Christensen provides a concept framework for defining the ecosystem surrounding technologies and product markets. It is a bridge in the theoretical framework between OSS and traditional proprietary software markets.

A large amount of the prevalent business strategy literature stemming from research in the 1980's takes its departure in a technology- or product-based definition of industry and marketplace, with internally cohesive analytical frameworks supplied or influenced by the Product Life Cycle, the Five Forces, and the Innovation Life Cycle (Christensen, 2009) comprising the “industry-bounded strategy and innovation paradigm”.

In Christensen's view, the PLC and Porter's five forces, “[...] view market structure as being predominantly exogenous”, resulting in firm strategies being shaped by existing industry structures and largely excluding the firm's abilities to shape industry structures.

In part, this appears to be the result of a foundation in analysis of manufacturing companies, resulting in an inordinate focus on the strategic challenges of competition within an industry of tangible single-product industries (Christensen, 2009).

The emergence of entire business models based on a foundation of licensing technology and focusing on research and brand control reduce the relevancy of these frameworks and it is necessary to at least attempt to identify parallel mechanisms to those defined in the original frameworks.

In other words, the coherency of the industry-bounded approach is reduced when analysing strategy in relation to a broader definition of what comprises the core business of a firm and the extent to which tangible products enter the picture.

A central factor in understanding the competitive environment for open source business is the understanding that the Porterian “industry” term must be expanded in order to establish a foundation for understanding the complex dynamics of continuous convergence and dynamics of technologies and product markets (Christensen, 2009).

“The product market concept is often used interchangeably with the industry concept. However, the former is not associated with features of robust boundaries and long life cycles as the case is with the latter.” (Christensen, 2009)
One of the most immediately relevant criteria for evaluating which external factors the firm must account for in its strategy are the clearly identifiable competitors in the technological field from a technology- and product specific standpoint. Therefore, I have elected to apply a segmentation based on product markets, since this provides sufficient distinction between individual competitive technology marketplaces.

Christensen provides a mechanism for decoupling technologies and product markets, allowing the product market in combination with the individual assets available to the firm to determine the relevant business model.

For example, while Microsoft Windows is clearly dominant in the consumer desktop market, with OS X as a laggard second and Linux practically non-existent. However, if we evaluate based on the server market, Linux has a much higher penetration. Thus it should be reasonable to expect distinctive difference between the competitive characteristics of each.

2.3 Understanding Software

Some elements of any software implementation are globally applicable – no matter what kind of source code (open source or proprietary) and no matter what the specific context of application.

In order to become known as what is generally perceived as ‘software’ the source code must be “compiled”\(^1\), i.e., being interpreted and translated to machine code for the specific platform on which one desires the end program or operating system to run.

In order for that process to be successful, several prerequisites are required, including a functional system, development tools etc.

While these steps are – in general – a good way beyond the average computer user’s scope, there are numerous issues and problems surrounding the transition from source code to the end product. In other words, the average user does not know (or need to know) the processes of making software or how programs ‘work’ - the supplier or distributor handles the preparation and distribution of software that is ‘ready to run’.

\(^1\) Def: “compile”, to run (as a program) through a compiler

Def: “compiler”, a computer program that translates an entire set of instructions written in a higher-level symbolic language (as C) into machine language before the instructions can be executed
(Source: Merriam-Webster Online Dictionary)
While the above is a quite simple example of the reasons that the average user will pay for a finished product such as, for example, a version of Windows from Microsoft or an installation disk of Novell's SUSE² Linux, the fundamental structure of the problem is applicable to business software as well.

What is interesting in relation to the analysis is that individuals or firms may prefer to receive or purchase ‘ready to run’ software even in cases where all the required information and technology to do the preparation is freely available.

The requirements for compiling software and implementing it in an IT infrastructure – regardless of size – unquestionably includes time (in the form of man hours), physical resources (i.e., hardware, power, hosting solutions etc.), and the knowledge required to transform source code into a usable piece of machine code.

Both man hours and physical resources can be regarded as commodities and do not represent inhibiting restrictions on the individual firm or user. Therefore they are unlikely to be the determining factor in creating the marginal benefits justifying the fact that software providers are able to sustain economic profitability.

This seems to indicate that there is a gap between the people or firms who provide the software and the people or firms that use the software. As such it could be argued that majority of the transaction cost is represented by the resources that must be committed to “translating” source code to working, implemented and supported applications. This touches upon one of the functions designated “absorptive capacity” (West & Gallagher, 2006) as one of the methods by which a company might apply it’s R&D organisation and capacity in order to benefit from open source.

In this discussion, however, it is necessary to distinguish between companies and organisations by the role in relation to a given open source project, since different interaction patterns require different sets of organisational capabilities which again has impact on the competitive dynamics in play.

² S.u.S.E (Software und System-Entwicklung) was founded in late 1992 as a UNIX consulting group, which among other things regularly released software packages that included SLS and Slackware, and printed UNIX/Linux manuals. S.u.S.E is an acronym for the German phrase "Software- und System-Entwicklung" ("Software and system development") (Source: http://en.wikipedia.org/wiki/Suse)
This might relate to other forms of software as well. One might hypothesise that this knowledge gap exists in all forms of software competition - the main difference between open source licensed software and proprietary software being how the additional value created by this knowledge gap is assigned.

The argument of (Chesbrough & Appleyard, 2007) that lessons from OSS competition might be applied in other IP-based value chains seems to be founded on a variation of an underlying assumption of such a gap as one of the value drivers of IP based competition.

This indicates that there the value drivers of software cannot easily be evaluated on the basis of directly committed resources alone. Thus, even though the core technology itself is freely available, the firms' assets and choice of business model provide the link to convert the technology into products or services that can be monetized (Chesbrough & Rosenbloom, 2002).
In this chapter, we will delve into the specifics of open source licensing. The question of which license a given piece of software is regulated by is essential in determining the competitive regime for the surrounding business models.

In other words, the extent to which the rules of competition are changed by the open source nature of a product is in part determined by the circumstances regulating the freedom to use and modify the code.

A definition on “Open Source” states these freedoms explicitly and is described in the first section. Under this definition, a multitude of software licenses qualify as open source and the implications in choosing such a license is thoroughly analysed hereafter.

The specific license chosen for an open source product determines what measures a firm may apply in the protection of its interests.

Further, in order to relate the mechanisms of open source licensing, I discuss how these relate to patent protection and copyright.

Examples are drawn out to illustrate the link between license and the available business models, empirical examples of open innovation in effect, including the implications for adoption in corporate R&D, how corporations are affected by this, and how it forms an integral part of their corporate strategy, allowing firms to benefit from reduced transactional costs and expand their value creation networks.

3.1 Understanding Open Source

In order to minimize ambivalence surrounding open source and corresponding licenses, we will consider the definition given by the Free Software Foundation, which defines Free Software as software that gives its users the “[...]freedom to run, copy, distribute, study, change and improve the software.” more specifically:
• The freedom to run the program, for any purpose (freedom 0).

• The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.

• The freedom to redistribute copies so you can help your neighbour (freedom 2).

• The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

(Free Software Foundation, Inc., 2009)

The Open Source Initiative (OSI) are the stewards of the Open Source Definition (OSD) and the community-recognized body for reviewing and approving licenses as OSD-conformant.

The OSD provide criteria for evaluating Open Source licenses thus providing a framework for evaluating new licenses. The Open Source Definition was derived from the Debian Free Software Guidelines 3.

3.2 Open Source Licensing

There are a number of different licenses that are widely applied in open source projects.

In the discussion regarding Linux, we will be focusing mainly on the GNU General Public License (GPL), specifically version 2 and 3, since the GPL or variants constitutes the most widely used licenses for open source projects. Also, the Linux kernel is licensed under the GPLv2.

However, the reader should be aware that a number of other OSI-approved licenses exist and are in use in high profile open source projects:

http://www.debian.org/social_contract.html#guidelines
The criteria for a conformant license are somewhat broader than what might be expected from the basis of GPL. It is important to remember that GPL was created with the express purpose of using copyright as a tool to preserve the availability of derived works.

This is an important point with regards to the economic systems emerging around Linux, but is in no way a prerequisite for a license to be recognised as “Open Source”.

The OSD criteria include the requirement of the license to allow free redistribution of the software and explicitly forbids the license to require “royalties or other fees.”

For example, BSD-licensed code can be “shut in” and incorporated into proprietary products as demonstrated in Apple's incorporation of the FreeBSD system as the basis for OS X. In contrast, GPL-licensed code is “viral” in the sense that the license dictates that derivative works are covered by the same license.

Furthermore, the source code must be freely distributable and made available with the software – either included or “[...]there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.”

The definition also explicitly forbids deliberately complicating or sabotaging other programmers’ ability to work with the source code.

---

In terms of what might be done with the source code, “The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.” and may not disallow redistribution of modified versions, although “The license may require derived works to carry a different name or version number from the original software.”

The criteria exclude any form of discrimination or other means of preventing or furthering the availability of the source to specific groups or field of endeavours. Further, once a program has been released under the license, further involvement from the original author may not be required for redistribution.

The requirements include a provision that the license must not be specific for a product and the rights provided are dependent on the program being released in a specific contest – for example in conjunction with a specific distribution. Also “The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.” Finally, “No provision of the license may be predicated on any individual technology or style of interface.”

3.2.1 Understanding the Implications of the Gnu General Public License (GPL)

The primary goal of the GPL is to ensure the preservation of the freedom (as in liberty) of information with regards to the content released under the license. A common misconception about content released under the GPL is that it is not possible to commercially exploit said content.

This is perfectly possible and even encouraged by the Free Software Foundation under a number of restrictions:

- The full source code for the published work must be included or delivered upon request.
Any person or legal entity are entitled to unlimited distribution of the source code. This right cannot be inhibited in any way (non-disclosure agreements etc.) if they have received a work released under the GPL.

The figure above is taken from “The Free-Libre / Open Source Software (FLOSS) License Slide (Wheeler, 2007)”

It is, however, worth noting that GPL uses copyright legislation to regulate the use of content released under the license. Provided permission can be acquired from all copyright holders to a work, there is nothing to prevent someone from releasing a work under additional licenses unrelated to the GPL (Dual Licensing). This is an important distinction since it is the foundation for business models taking advantage of dual licenses to be able to provide a version of their product without the GPL-related restrictions.

The “viral” effect of the GPL license is created through the rights granted in section 3 of the GPLv2.
You may copy and distribute the Program (or a work based on it, under Section 2) in object
code or executable form under the terms of Sections 1 and 2 above provided that you also
do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be
distributed under the terms of Sections 1 and 2 above on a medium customarily used for software
interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a
charge no more than your cost of physically performing source distribution, a complete machine-
readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and
2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source
code. (This alternative is allowed only for non-commercial distribution and only if you received the
program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it.
For an executable work, complete source code means all the source code for all modules it contains,
plus any associated interface definition files, plus the scripts used to control compilation and
installation of the executable. However, as a special exception, the source code distributed need not
include anything that is normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on which the executable runs,
unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated
place, then offering equivalent access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not compelled to copy the source along
with the object code.

(Free Software Foundation, Inc., 1991)
Although the GPL provides multiple means to comply with the licence's requirement to make the source code for a product based on GPL-licensed code available, the most common is the method in paragraph (a).

In summary, “Its effect is simple: if the original program is released under the GPL, then you may copy and distribute it, or a modified version, without infringing the copyright in that program provided you also distribute the relevant source code. The provision of your source code is integral to your ability to deal with the original source code. If you fail to distribute your source code, you have gone outside the terms of your original permission to deal with the copyright work, and (in the absence of any applicable defence, or other permission to deal with the work) you become liable for infringing the copyright in it. (Kremer, Ben, 2004)”

3.2.2 Potential Caveats of distributing software for Linux or other Free Software:
Due to the viral nature of the GNU GPL companies must review their strategies for software development and decide the extent to which they use external code in their project. It is important to notice that there is nothing to prevent a company from developing software for Linux without releasing it under the GPL. In order to do so, however, a firm must pay attention to how they choose to implement certain functionalities since this will have an impact on whether the product fulfils the terms of the GPL.

While there is nothing to prevent a firm from charging for their product, it must be considered, that once the product is released under the GPL, there is no legitimate way to prevent other companies from using the product and even releasing their own version of the product, as long as they respect the terms of the GPL.

Thus, it is entirely possible and completely legal for another person to create a competing version, which is, in essence the exact same product as the one offered the company or project originally developing the code.

Another possibility is for someone to create another version of the project, a so-called “fork” (Raymond, 2002), and developing the project towards another application, purpose or market segment than what was originally intended.
Provided a company has full ownership of the copyrights for a project, it may, however, choose to release the same version or later versions under a different license. As demonstrated by the MySQL dual license approach, which I discuss in Chapter 5: “Code ownership”,

While software can be released under multiple licenses, other licenses have no impact on the rights regarding the code released under the GPL.

### 3.3 Open Source and Patents

One item of particular interest, in software in general, is the patentability of software.

Currently, it is possible to be rewarded a patent for software implementations in U.S.A., where the legislation allows software to be patented directly. At the time of writing, it is not possible to patent software directly within the European Union. Therefore, patents within the EU must be applied for, based on underlying technologies or mathematical algorithms.

None the less, patents propose unique challenges for open source software, since it is a mechanism that has been developed on the implicit assumption that ownership of a product, technology or project must naturally fall to a clearly defined entity.

The patentability itself of software is a disputed point, and since patent disputes are often solved by cross-licensing agreements, it can be argued that the protection offered by patents in the realm of software is limited at best.

*First recognize that any patent that covers Free Software is going to cover non-free software, because licensing terms are irrelevant to a patent's scope. So, there aren’t any patents that only threaten Free Software; they all threaten all software, regardless of license terms.* (Ravicher, 2004)

Patents have unique characteristics though, primarily in the sense that a patent holder be awarded a ban on the perceived infringer's right to continue business until a patent dispute has been resolved.
In order to accommodate a similar protection for open source projects and reduce the effects of the nervousness generated by SCO's infamous lawsuit against IBM regarding the alleged use of SCO's intellectual property in Linux, the Open Invention Network (OIN) was established as a consortium of a number of prominent technology companies, including IBM, Nokia, Red Hat, Philips, Novell, NEC and Sony.

The consortium was created to extend the protection of the pledged patents to projects released under approved licenses. Similarly, the benefits normally reserved for companies with patent portfolios extensive enough to cross license are extended to open source projects, since the OIN provides a base for negotiating on similar terms but with the protection extending to the covered projects.

It is worth noting that the implications of software patents for OSI approved licenses can vary a great deal. In November 2005, IBM pledged royalty-free use of 500 of its 10,000 US patents, with more patents to be made available under the scheme in the future. Any software covered by the Open Source Initiative's Open Source Definition was granted royalty-free use of the named patents (IBM, 2005), (Marson, 2005). This serves as an example of the network effects of the economic systems surrounding open source technologies serving as a motivator for large players to shield projects, enabling all to benefit.

However, the characteristics of the individual licenses may require that alternate paths are taken in order to ensure that the goals in choosing a specific license are met.

---

9  http://www.groklaw.net/staticpages/index.php?page=20031016162215566
10 http://news.zdnet.co.uk/software/applications/0,39020384,39183594,00.htm
For example, the PostgreSQL project decided to replace the project’s cache management algorithm (Fournier, 2005) in order to avoid violating a patent filed by IBM, despite that IBM had granted royalty-free use of the patent in question for any software covered by an OSI-approved license, due to concern that using an IBM patent will prevent companies from reselling the open source database as closed source — something that is permitted under its OSI-approved BSD license (Marson, 2005).

The result was that companies that were using PostgreSQL in proprietary products were not protected by the pledge and would be required to address the issue of whether to contest the patent or be potentially liable for an eventual patent suit (Brockmeier, 2005). This reflects a different kind of integration of open source in a business model and will be briefly discussed in Chapter 5: “Legislative environment”.

Apart from the direct access to protection from claims extended to projects and technologies, a number of technology vendors, including HP and Red Hat, have made public pledges to keep their customers damage free, should they be approached with claims of infringement based on their products.

The unique property of patents, i.e. that they can be used as a foundation to prevent a competitor from continuing infringement, can have devastating effects, if the company holding the patent chooses to leverage the patent to block competitors.

In fact, the recent lawsuit impeding Microsoft’s own OOXML document format exemplifies the double edges of the issue of software patents:
Toronto, August 12, 2009 – Yesterday, in the U.S. District Court for the Eastern District of Texas, Tyler Division, the Honourable Leonard Davis issued a Final Judgement which upheld the verdict won on May 20, 2009, by i4i, a global technology company headquartered in Toronto, ON, Canada. The Final Judgement is an award in excess of $290 million and includes a Permanent Injunction against Microsoft Corporation (“Microsoft”) for custom XML in Word 2003 and Word 2007. (“Judge Grants Permanent Injunction against Microsoft in Favour of i4i, Damages now $290 million USD,” 2009)

The injunction preventing Microsoft from selling its Office Suite was stayed on September 3rd 2009 pending the appeal ruling. Disregarding the discussion regarding the validity of the claim, the case nonetheless serves to illustrate the scope of effect that a patent claim can have on a firm's ability to market its product.

### 3.4 The evolution of the GPL

On June 29th, 2007 GPLv3 was released from the Free Software Foundation. In an attempt to address some of the issues arising from the SCO lawsuit against IBM and Novell as well as the proposed bill regarding the patentability of software, this new version of the GPL incorporates a number of changes targeted at ensuring that software released under the new license is better protected against fraudulent lawsuits and circumventions of intended implications of using GPL-licensed code.

>“It means that[...] GPL code can be used in the cloud computing market in exactly the same way as BSD code can be used in the traditional software market. (Allison, 2009)

Also, there are trends (in part the reason for IBM's OnDemand strategy) pointing towards a significant amount of future solutions being provided via SaaS (Software as a Service) and similar delivery methods that could challenge the provisions of the traditional licenses.
Why did you invent the new terms “propagate” and “convey” in GPLv3?

The term “distribute” used in GPLv2 was borrowed from United States copyright law. Over the years, we learned that some jurisdictions used this same word in their own copyright laws, but gave it different meanings. We invented these new terms to make our intent as clear as possible no matter where the license is interpreted. They are not used in any copyright law in the world, and we provide their definitions directly in the license.

“GPLv3 isn’t a radical new license; instead it’s an evolution of the previous version” (Smith, 2008).

One of the challenges for the version 2 of the GNU GPL, is that the software patents were not a widespread phenomenon when the license was published in 1991.

The GPLv3 was specifically introduced to ensure that the license sufficiently addressed current challenges as well as incorporating some of the lessons from the real world applications and experiences with the license:

**GPLv3 must cope with threats to freedom that we did not imagine in 1989. The coming generation of computers, and many products with increasingly powerful embedded computers, are being turned against us by their manufacturers before we buy them—they are designed to restrict what we can use them to do.** (Stallman, 2007)

**Discriminatory patent deals:**

Microsoft has recently started telling people that they will not sue free software users for patent infringement—as long as you get the software from a vendor that’s paying Microsoft for the privilege. Ultimately, Microsoft is trying to collect royalties for the use of free software, which interferes with users’ freedom. No company should be able to do this (Smith, 2008).

While the statement is certainly political in nature, the goal of the license itself has been to establish a formal interpretation of the interaction between software licenses and the copyright mechanisms that the GPL framework relies on.
While the GPLv2 have been tested in court and proven to be enforcible (The Software Freedom Law Center, 2007) on several occasions, at the time of writing no one knows how effective the clauses regarding patents will be in court disputes.

The wording has been criticised as too broad given that it covers ‘future patents’ that might be filed or invented with regard to a technology area similar to that which a user contributes or licenses under GPLv3. (Laffan, 2007)

Furthermore if a FOSS Project were to be started post GPLv3 publication, and all other things being equal, it would probably be more advantageous to choose GPLv3 over GPLv2 for the additional patent protection it provides. (Laffan, 2007)

However, in my opinion it sound that a standardised successor to GPLv2 exist, providing an option for projects which for one reason or the other need to address specific use cases or needs.

Further, a fairly common method of ensuring the necessary flexibility in licensing is for projects to be released with a clause of “release under the GNU General Public License version 2 or later”, enabling projects to adopt to future versions of the GPL when deemed appropriate and – depending on the specific terms – in some cases allowing the licensee to choose which version of the license is desired.

The second big change that GPLv3 was introduced to address is the DRM trend that has pervaded the media industry. DVD encryption, copy protected CDs, music purchases that can be only downloaded a distinct number of times to your MP3 player are all examples of content owners attempting to redefine what constitutes “fair use” in the realm of copyright.

Legislation like the Digital Millennium Copyright Act and the European Union Copyright Directive make it a crime to write or share software that can break DRM. These laws should not interfere with the rights the GPL grants you. (Smith, 2008)
Thirdly and perhaps more expected is the GPLv3 section which attempts to deal with Tivoisation. Tivoisation is a term named after the Tivo product, […]. Tivo contains a small Linux OS which under GPLv2 requires the hardware manufacturer to make the source code available to users – which Tivo does […]. However, Tivo contains a special mechanism which shuts down if it notices changes to the code. Therefore whilst Tivo is fulfilling its obligations as required under GPLv2, it is actually inhibiting the four freedoms as set-out by the FSF, […]. (Laffan, 2007)

If a device can run arbitrary software, it’s a general-purpose computer, and its owner should control what it does. When a device thwarts you from doing that, we call that tivoization. (Smith, 2008)

The GNU Affero General Public License (“APL”) is an example of necessary modifications to address specific scenarios where “distribution” is no longer a useful trigger for the license’s requirements for redistribution. The APLv3 was introduced specifically to be able to address this issue.

“The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a public server, gives the public access to the source code of the modified version.” (Free Software Foundation, 2007)
In July 2007, version 3 of the GNU General Public License barely accounted for 164 projects. As of September 2007, nearly 600 projects of the 5000+ active projects listed on Sourceforge (the prevalent open source project repository) and licensed under GPLv2 or later had moved over to GPLv3. (Laffan, 2007)

A year later, the number had climbed past 2,000 total projects. Today, as announced by Google open-source programs office manager Chris DiBona, the number of open-source projects licensed under GPLv3 is at least 56,000.

And that’s just counting the projects hosted at Google Code. With more than 225,000 projects currently hosted at Google Code, that’s a lot of GPLv3. (Assay, 2009)

It is worth mentioning that the adoption of GPLv3 is in no sense mandatory and many large scale open source projects – including the Linux kernel – have so far elected to stay with version 2 of the license. That said there are multiple projects released under GPLv2 that includes in their license terms the ability to apply later versions of the GPL at the licensee’s option.

3.5 Aligning with open source licensing

One of the key points of the viral licenses is the fact that any compliant use of intellectual property in a GPL-licensed product will automatically take part in protecting that and related intellectual property in future releases.

The way these licenses are made help shape the competitive measures that can be applied in the market. Also, the future direction of licensing addresses the more immediate challenges to licensing (AGPL, Patent, Tivoisation).

The Linux kernel that constitutes the core of Linux distributions has still chosen not to adopt version 3 of the GPL, a decision that has caused some controversy, but ultimately boils down to a stance that “if it ain’t broken, don’t fix it” and concerns that the diverse coalition of contributors to the kernel might become less stable, or even fragmented due to concerns in relation to the patent portfolio and DRM clauses in the GPLv3. On a more practical note, the Linux kernel does not per default attain copyright to the contributed code, which poses the challenge of finding a compatible upgrade path for code contributed by a wide variety of sources (Morton et al.) (Vaughan-Nichols, 2007)
Further, while the lack of adoption by Linux has caused some dispute, the numbers seem to support the importance of the license regimes keeping up with the trends and direction of the marketplace, such as, for example, software as a service as well as other methods that inadvertently or not circumvent the redistribution clause of the GPL.

The PostgreSQL example serves to illustrate a critical aspect in the discussion of the availability of business models as determined by the licensing terms of the technology.

In my opinion, this illustrates one of the concerns of IBM in determining which licenses are compatible with the firm's overall strategies. Specifically, IBM has historically focused on developing an extensive patent portfolio, having received the highest number of US patents of any company in the world for fifteen consecutive years until 2007, according to the firms' licensing page (IBM, n.d.)

Thus, due to the nature of the firm's assets, IBM has a limited choice of plausible licenses for the firm's open source involvement, if compatibility with the firm's existing portfolio is to be maintained.

In IBM's case, the evidence so far seems to support that the congruity with the corporate portfolio is important, since the high profile projects (Eclipse, Apache) are either licensed under GPLv2 or variants specific to the individual project.

The viral element of the license is central for the fit with IBM. The GPL allows IBM to extend the protection of the firm's traditional IP protection schemes to projects that might otherwise have been in risk of infringement lawsuits by IBM itself or it's competitors. In this respect, the viral element acts as gatekeeper for the royalty-free access to patent licensing, since the license terms ensures that the knowledge is much less likely to be closed off and used to the exclusive benefit of any single firm or project.

It is interesting to note, however, that apart from the discussions of GPLv2 as a proven platform for viable business models, the GPLv3 becomes controversial when seen in combination with other elements of corporate strategy. For example, while SONY is a member of the Open innovation network, it is also a major copyright rights holder through the company's film and music divisions and deeply embedded in distribution chains relating to the sale and rental of media.
SONY has a history of applying different kinds of what is often referred to as “Digital Rights Management” software and systems – restricting the use and – in the United States – using the DCMA to prevent circumvention of these mechanisms.

Thus while participation in forums like the OIN and application of open source software can be interpreted as signs of commitment to the protection of technologies, it does not immediately or automatically signify internal consistency in between the branches of corporate strategy.

One prime example of a company circumventing the redistribution clause as a foundation for translating open source into proprietary competitive abilities is Google.

Google's use of open source technology, primarily in the form of Linux running on their server farms (Corbet, 2009) is largely exposed to users through the services that Google enables through it's use of technology. With the exception of Google's “Google Search Appliance” product, Google is not releasing it's customizations of the Linux kernel, nor is it required to, according to the terms of the GPL License.

It is doubtful whether this is directly a critical competitive decision, since simply replicating Google technology in any given iteration would not alone enable new companies to challenge Google in the firm's core market. It is likely, however, that a complete publication of the modifications that Google applies to it's code would reveal information about some of the data sources and mechanisms that provide the foundations for the data reporting that makes the “secret sauce” elements of Google's core technologies possible.

3.6 Corporate strategy and Open Source

Just like the decision about which licenses are compatible with the overall and/or corporate strategy must be made by any firm getting involved with development, or based on open source, the individual projects must also assure that primary stakeholders and community dynamics are not negatively affected by the project's license scheme.
One example is PostgreSQL, as illustrated earlier. Another prominent example can be observed in the case of the re-licensing of the Mozilla project, based on “perceived incompatibility” between the MPL (Mozilla Public License) and the GPL. As part of the re-licensing of the code a triple-license scheme was chosen, enabling users of the code to choose between the MPL, the GPL or the less restrictive LGPL, the latter being chosen to support further proliferation of the project by ensuring that it remains an option for the widest array of uses possible. (mozilla.org, 2007).

Thus the license of a given technology or project becomes an imperative consideration, since it determines both the compatibility and plausibility of integrating or developing code under a given license, as well as determining the fundamental business models and competitive measures that are applicable in business models based in whole or in part on said project or technology.

On the other hand, the availability and general adoption of standardised licenses help to reduce the transaction costs of selecting individual projects, provided a general policy regarding compatible licenses has been established.

On a more fundamental level, Teece’s basic distinction between strong and weak appropriability regimes and related plausible business strategies becomes central since open source licenses guarantees that assets cannot easily be made exclusive. Thus the competitive strategy must focus elsewhere, including the constellation of complementary assets, in order to gain from the innovation of a given open source technology. I will return to this in Chapter 5: “The importance of complementary assets“.

In my opinion, the rationale behind the establishment of the OIN follow the argument for maximization of R&D resources through patent pooling (West & Gallagher, 2006, p. 320) by establishing freedom to operate by reducing or eliminating the need to establish individual cross-licensing agreements for specific cases – thus reducing transaction costs throughout the ecosystem of technologies or projects that would otherwise directly or indirectly need to address the provided patents.

The application of patent pools in open source provide similar benefits but - just as the license ensures the level playing field in the game of competitive positioning – the open source licensing ensures that the benefits are available to everyone – including actors that do not participate or otherwise contribute to the pool.
Expanding the thought of value creation networks and accepting that there are dynamics that might not be directly observable leads to the possibility that these “shared benefits” allow contributions of strengthening other elements of the value network or ecosystem that in the long run provide benefits the original patent holders.

3.7 Conclusion

Open source software as a concept was originally defined by the Free Software Foundation, which is also responsible for the maintenance of, among others the GPL and AFL licenses. The Open Source Initiative has emerged as the authority for approving licenses that adhere to the definition.

Patents propose unique challenges for open source software. Since open source projects often exist without the traditional legal entities which are assumed in patent litigation, the Open Invention Network consortium extends the protection of their pledged patents to projects released under OSI-approved licenses. This provides the potential to reach cross-licensing agreements to solve patent disputes for projects that might otherwise have limited options for acting.

The use of open source licenses have strategic implications for corporate strategy, if products so licensed are used, since the competitive situation is affected by the ability of competitors to legally use the same code to create a competing product based on the same technology.

The specific license of individual projects have implications for business models it are possible to establish around the technologies provided, as illustrated above and in the cases of MySQL and PostgreSQL.

The establishment of the Open source software can be explained by the rationale of motivation of spillovers and maximization of internal innovation by through the pooled R&D and IP mechanisms thus provided.

In the next chapter, I will examine some the challenges that faces “traditional” product-oriented, proprietary software business models. Part of these challenges are enabled by the increasing move towards open standards in software that in part is caused by the open source movement.
4 Challenges for traditional software business models

Software business models have incrementally morphed into new shapes since its 'commercial breakthrough' in the 1970's. Companies have grown successful and large on proprietary software, where business value was directly linked to the code and core technology.

The emerge of OSS have challenged traditional software business models on competitive parameters and forced a shift in business offerings and value capture. The trend has transformed the nature of a number of large software houses from a product-oriented focus to hybrids of, or primarily service companies. This has emerged as a general trend: the software industry is shifting from product focus to service offerings.

This chapter accounts for the traditional software business models with examples of how they have been successfully applied in the past, and sheds light on the challenges same models face in today's redefined competitive landscape.

4.1 Traditional Business Models

In the article “The logic of revenue logic? Strategic and operational levels of pricing in the context of software business” (Sainio & Marjakoski, 2009) the authors present their findings on typical patterns for revenue sources in software companies. They conclude most software based companies generates revenue through offerings combining both software and services, sometimes in part or completely generated through partnerships or other third party mechanisms. Sole software providers stripped of service offerings, so-called “pure” software companies, are very rare.

Standardised, proprietary software is typically priced on the basis of one of the following:
Usage, including transaction models, database-specific models, CPU models, and server-connection-specific models

Purpose-of-use based on the fact that the same product can be used for several different purposes, and the charging is based on how it will be used

Profit-sharing models, often used in different forms of cooperation and strategic partnerships, as collaboration enables providers to charge more for better results and to collect a fair share of the increased revenues

(Sainio & Marjakoski, 2009)

Microsoft exhibits some of the core characteristics of what might be referred to as a typical software vendor, in that it has traditionally centred around product strategies, selling standardised products in very large quantities supplemented with partnership agreements with hardware vendors on multiple levels of the computing hardware supply chain.

Very broadly put, Microsoft has pursued a business strategy around proprietary software, vendor lock-in and IPR strategies since the early 80's and has been very successful.

Historically, Microsoft's software sales are to a high extent dependent on a re-purchase of licenses for their products in the sense that the firm will release a new version of its software tools requiring existing customers to purchase new licenses in order to use these new versions.

This strategy has proven successful in the past, getting customers to upgrade as new hardware and business requirements was developed, and strengthening Microsoft's market position by keeping the technology and file formats proprietary.

This has established a long running pattern of cyclical updates, where critical mass in the proliferation of a new version of, say, Microsoft Office, caused demand to shift to the new version as businesses had to be able to exchange documents in the new formats.

Microsoft has been very successful in establishing a near complete dominance in the desktop segment, providing a platform for the company to move into server computing. In this field Microsoft has used the familiarity with its products and relatively low adoption costs as a platform tabbing into the emerging high growth SMB segment.
Microsoft has historically been using a variety of different license models, selling both per-user, per-CPU, bulk licenses to companies and providing services like for example spam filtering directed at corporate clients. For all practical purposes Microsoft is not complementing their business with services revenues and is therefore a good example of a “pure” product-based business model (Cusumano, 2003)

This “traditional” approach to monetising software products has indisputably proven effective in of the history of the personal computer and for many software companies providing standardized software on the market today.

Same companies furthermore provide service offerings and subscriptions to complement the revenue generated from direct sales, but the pivotal point of business is in the product.

4.1.1 Co-specialisation
An interesting pattern of co-specialised relationships emerges when looking at a typical enterprise license sale in the world of proprietary software. The described example is common enough that it is used as basis for rule-of-thumb estimates for business case calculations.

A typical license agreement between a proprietary vendor will include 1) an up-front license fee, establishing the customers “perpetual right” to use that version of the software, 2) a maintenance agreement and 3) additional services such as support agreements, software integration and, customizations or extension of the software (Cusumano, 2007).

Vendors of proprietary software are typically the only part legally entitled to license the product to the customer and therefore exclusively able to command the terms of the first two elements of a license similar to the above outlined. This holds true even if the transaction is conducted by a third party, for example a business partner. This simply reflects an extension of the relationship.

It is possible for outside firms to provide services, extension or customizations of the product licensed but the parties are inherently committed in a co-specialized relationship (Teece, 1986), and placed in a dependency relationship to the software vendor.

There are definitely cases where the relation described might have different characteristics, but I would argue that this is largely dependent on the relative power of the constellation. This makes little difference from the customer's point of view.
This approach to monetisation is directly in extension of the underlying logic from the PFI approach. The more integrated a software product is in a company's core processes, the more the company develops a co-specialisation to the asset controlled by the software vendor, making switches both costly and risky. The lock-in effect of clients to their software vendor is in effect.

Microsoft's Windows and the Office Suite are examples of how this strong tie can be a cornerstone to a company's success and market dominance.

4.2 Challenges in today's competitive landscape

Bearing in mind that Microsoft currently hold comfortable market leader position in the firm's traditional core markets, emerging alternatives not only challenges Microsoft's market share but contribute to redefining the paradigm of the product itself. Google and Soho are two examples of online office suites that, while not be immediately comparable to desktop-centric model of traditional office suites, fulfil the same purpose. As evidence of their effect, Microsoft recently released an online version of its office suite.

Microsoft has historically been able to leverage from the firm's dominant market position and ownership to pursue a business model on the assumption 'owning' the dominant design (Teece, 1986). The increasing availability of open standards and technologies, adopted and compatible across vendors and niches, pose serious challenges to these foundations.

Microsoft has built a huge success based on the fundamental approach of the PFI, appropriating value through enforced restrictions. In my view, however, a number of trends now fundamentally in challenge these foundations:

- Commoditisation through increased convergence towards standards which reduce the traditional opportunities for lock-in.
- Expanded availability of technologies and distribution channels challenges the constellation of assets and redefines competitive parameters.
- Digital Rights Management and how they can be enforced changes the game for protection of innovations.

The following sections will go into further details.
4.2.1 **Commoditisation reduces the traditional opportunities for lock-in**

Constant emergence of new products and technologies is accelerating and conventions in design and key terminologies have emerged, creating a convergence towards a set of standards. In contrast to earlier phases of information technology, today's products and technologies are increasingly based on open standards - providing interoperability across software and hardware stacks from multiple vendors.

This heightened degree of interoperability reduces the effect of vendor lock-in that has helped protect business models like Microsoft's from outside threats. The same tactics that helped Microsoft effectively kill-off competition in the market for office suites, like Wordperfect and Lotus Symphony, in the 90's are no longer applicable. The competitive situation is changed by the emerge of open formats and access to a productivity suite no longer depends on a purchase decision, but is available “on demand” through online distribution. Competitive advantages from lock-in as a result of path dependency become less significant when the transaction cost for choosing a competitive product is nil.

When all business applications supports a firms fundamental requirements, to effectively conduct operations, the motivation to endure switching costs in connection with a migration lessens. This helps explain some of the challenges that software vendors like Microsoft experience in today's competitive landscape.

Microsoft can no longer compete as the low-cost provider in many market segments. Server software commoditisation - driven by open source initiatives and enterprise preferences for function-limited, best-of-breed products - is exerting continued pressure in many of Microsoft's core markets. (Kobielus, 2003)

Many of these solutions are additionally made available as online applications which implies platform independence. Solutions locked to the desktop space then looses their effect as complementary assets. Further, the increasing focus on virtualisation technologies has resulted in Microsoft committing patches to the Linux kernel (Microsoft, 2009) in order to improve Windows performance in virtualised environments. This illustrates how the tie-in between hardware and operating platform that was the basis for companies like HP and IBM's server businesses is practically eliminated in the server market today.
The current situation still provides room for benefiting from lock-in, but the trends of increased standardisation and commoditisation of core operating systems and productivity applications pose significant challenges to traditional product based software business models.

4.2.2 How the availability of technologies and distribution channels changes the constellation of assets and redefines competitive parameters

At the time when the basics of industry-bound paradigm were established, technological innovations typically stemmed from a sort of basic research within enterprise research labs such as IBM Research or Bell labs, or through government-funded programs in education or the military. This included most of the research necessary to maintain a competitive position in electronics or software.

Today, thanks to the proliferation of standardised frameworks, global manufacturing and cheap processing power, high technology come from a variety of sources. While the need to establish economies of scale on the basis of a dominant design may still be sound in manufacturing, but the effects on the entry barriers for software innovations may be rightfully challenged today.

In broad terms, to develop a novel technology service or software product today, resources can be focused and dimensioned precisely on the basis of the firm's specific requirements, with both proprietary and open source platforms and toolkits available to enable implementation and development. Standardisation also determines that hardware platforms and communication protocols given and require virtually no barriers to overcome. This is a glaring contrast to some of the challenges faced by, for example Xerox's spin-offs in the late 1980's and early 1990's. Then, standards either had to be introduced, as was the case and core business model of Adobe and 3com, or were unchangeable and thus became exogenous variables for the companies' strategies (Chesbrough & Rosenbloom, 2002).

To some extent, open source licensing provides an opportunity to transform capital requirements of software innovation towards marginal investment in the development of the specific service or product implementations now that foundations for providing the majority of base functionality is freely available. Whether such a service or product itself must be open sourced depends on how external projects are applied and incorporated as well as the firm's strategic position.
We will without a doubt see continuous change in the way that software and services are delivered and developed and transaction economics dictate a trend towards increased degree of standardisation as the global economy continues to develop. This will inevitably continue to provide opportunities for niche entries, and competition based on constellations of assets, integrations and process support.

4.2.3 Digital rights management and lack of enforceability changes protection of innovations

As has already been established, it is no simple feat to restrict access to software innovations. Many factors influence the degree to which a given technology can be protected. While attempts are still made to establish and defend software patents, such protection models are unlikely to provide sufficiently effective control methods to provide a basis for a business model on its own.

Up until now, the content industry have had limited success with attempts to protect its physical distribution-based business model by prosecuting consumers sharing media files online. The current focus is on incrementally restrictive technologies applied to consumer hardware as well as embedding so-called “copy protection” through encryption of consumer media. These technologies, dubbed “Digital Rights Management” (DRM), include DVD encryption, HDMI “secure path”, windows media file formats and Apple's AAC encapsulate media in cryptographic “shells”, in order to retain control of where and how media can be applied with the rights holder.

This means is that “fair use” as it applies to music media, up until the 90's, is attempted redefined into a different model: the lack of physical media is used as a template to redefine the product purchased.

No user is ever going to say: "Oh no. I can now play the music I bought for my PC on my Mac. I must install a patch so I can't do that any more." (Schneier, 2006)

While it recently became possible to purchase music through, for example, Apple's iTunes store without embedded encryption, uninformed consumers are bearing the risk of losing access to their purchased music when technologies or license schemes reach end of life or end of service, as happened when Microsoft decided to rethink their PlaysForSure media licensing scheme:
Microsoft’s decision to turn off the MSN Music authorization servers serves as a painful reminder that DRM ultimately severely limits your rights. Companies that control various DRM schemes, as well as the content providers themselves, can yank your ability to play the content which you lawfully purchased (and now, videos) at any moment—no matter what your expectation was when you bought it. (Cheng, 2008)

The core problem of DRM schemes in media protection is that it will eventually need to be decrypted and decoded in order to be transmitted as music or video. Circumventing the protection scheme is, in simple terms, simply a question of where to “plug in” in the decryption chain, effectively rendering this type of protection unenforcible for the committed consumer. Circumvention techniques has been developed for every form of copy protection introduced so far.

[...] we have primarily discussed the relations between the regimes of IPR protection and rates of innovations, basically concluding that either the relation is not there, or, if it is there it might be a perverse one, with strong IPR enforcement actually deterring innovative efforts.

[...]However we know also that IPR protection is only one of the mechanisms for appropriating returns from innovation, and certainly not the most important one.

(Dosi, Marengo, & Pasquali, 2006)

As a result, a lot of effort is put into restricting and “locking down” access to media by combining hardware protection schemes and having the operating system commit significant resources to re-encrypt media content in multiple steps. It is interesting how this provides no value for the consumer. On the contrary, these restrictions are placed solely as a method to enforce a business model for media companies. In other words, platform control is being used as a complementary asset to support media business models.
(Chesbrough & Appleyard, 2007) argue that enforcing DRM is a costly basis for a business model. Interestingly, Microsoft has chosen to pursue this path of increasing restrictions on customer freedom and options in order to maintain lock-in. From a customer perspective, even if accepting the premise that DRM will only work as intended without applying restrictions erroneously, there are no ways to ensure that such errors will be corrected, should they occur.

From a strategic perspective, the central challenge to this approach of implementing restrictions and inhibiting possibilities of your product lies in the reduction of entry barriers for competitors. On a fundamental level, consumers can purchase a copy of Windows that will reduce the possible ways to interact with content that consumers themselves have produced or purchased the rights to use. Alternatively they can purchase or download cost free an operating system designed to be applicable in whatever way necessary without restrictions.

While this may be a bid from Microsoft to establish dominance of the supply chain for content delivery, there are counteractions like VESA's free display port (as opposed to HDMI “secure path”), that does not incur license fees for hardware manufacturers, undermining the extent to which restrictions can be incorporated on the hardware level. In short, the effort to establish co-specialisation of the hardware components necessary to establish such dominance is unlikely due to multiple economic mechanisms working against it.

4.3 Consequences for strategy

A fundamental characteristic of DRM-based business models is that they are intrinsically unenforceable, in practice unless limited with access restrictions in the purchasing environment. The restricting measures exposes customers to risks, higher costs and reduced functionality. In short, business models that restrict what users can or cannot do with their legally purchased products is problematic to enforce and muddles the value proposition for the customer.
The paradigmatic success stories of the industry have been introduced on the basis of open standards that have fostered increased competition and driven innovation. In this setting, it could be viewed as risky to pursue a strategy based on the notion of closing off technology. Corporate giants like Microsoft might be able to leverage its near-total dominance in the consumer and business desktop segments to gain a foothold in distribution, but this business model does little to provide solutions for other product-oriented software companies facing similar challenges.

Conclusively, software firms with product-centric business models must respond to the changes in software delivery models. Cusumano argues for the necessity of choosing a strategic focus “[...] because selling products is different from selling services, even hybrid solutions companies need to have a primary strategic orientation of either services or products.” (Cusumano, 2003)

There is evidence to support the notion that “the pure product” software company is being replaced by hybrid models expanding the notion of core business to encompass surrounding services or other ecosystems. There are multiple examples of firms which have traditionally been based on product-centric business models changing towards service models. IBM, Siebel SAP are examples of companies whose software business model has evolved to include complementary assets in the form of integration and business services (Cusumano, 2008)

4.4 Conclusion

We have seen that traditional software models are challenged by three market trends in the software industry:

- Commoditisation of software and standards has created a modular and highly flexible platforms that reduce traditional software lock-in opportunities.

- Lower entry barriers as a consequence of strong availability of technologies and distribution channels changes the traditional competitive parameters on size, funds and human resources.
• Protecting assets through digital rights management only works in constellations where customer value is actively reduced, rendering alternative business models where IP is eliminated as a competitive factor.

The combined effects of these trends is a general shift in the traditional software business models. From product centric business models they have extended into including complementary services built around products or technologies, a natural consequence of the increasing entanglement of technology in business processes. IBM's purchase of Price Waterhouse Coopers consulting division is one example of a scope-oriented approach to software sales, where the business process integration acts as a fundamental value driver for the software sales.

5 Open Source Business: Appropriating value in a weak appropriability regime

This chapter brings the pieces together of the previous chapters and gives real life examples of different approaches using open source software models to increase value capturing and competitiveness. Next it is discussed how Teece's assumption in protection of assets may be less relevant with a strong palette of complementary assets that seems to be central to open source software models. I then attempt to identify important complementary assets that can make an open source business model competitive.

Based on the findings of this thesis work, that open source software business are the generally face the same strategic challenges as proprietary software business models.

I furthermore link this to existing theory and conclude that normal market powers are at play, and that the findings of Teece's framework provides a useful tool-set for evaluating strategic assets of open source firms, although the benefits are established through the surrounding assets rather than the core technology.
5.1 Examples of existing open source business models

For the use in the discussion and conclusions on open source business models and complementary assets I have outlined empirical examples that exhibit relevant characteristics for the further analysis. The

We need to see how these examples demonstrate:

• that OSS can provide new opportunities both in value capturing, and in increased competitive capabilities.

• that software business models are each determined by the complementary assets available to them and their chosen relevant product market

5.1.1 IBM

Open source, provides value for IBM in two central ways:

Firstly it provides a well-supported and maintained platform on top of which IBM can build and sell specialised applications and services, example: IBM used to spend approximately 500 mil. USD on development of AIX. Now, an investment of 100 mil. USD in open source initiatives returns developments progress worth 900 mil. USD.(Chesbrough & Appleyard, 2007)

This gives IBM strengthened development capabilities and frees internal resources to focus on other initiatives.

Secondly, the integration of Apache into IBM's WebSphere offering provides direct, tangible benefits by allowing some of the R&D processes to happen externally (West & Gallagher, 2006), while the integration with a number of IBM's proprietary components allows IBM to maintain a unique value proposition. Note that Apache uses a more permissive license than the GPL – thus enabling this particular sort of integration.

5.1.2 Google

Google have taken a strong supporting position to open source, with an increasing number of the firm's software endeavours being released as open source.

In short Google is active in releasing projects and patches for projects where it does not intercede with core business, in part through programs such as Google's “Summer of Code” (Dibona, 2008), (Google, 2009). By their own account:
“[…]Google is releasing a new project every two or three weeks. Estimating that he firm releases about a million lines of code a year from the company every year, not counting Android or the really large open-source projects like GWT.” (Dibona, 2008)

Apart from Android and Google Web Toolkit (GWT), notable extensive open source projects backed or released by Google includes Wave and Chrome OS. Further, Google has been active in sponsoring student projects:

As we brought people in, we wanted to be sure more open-source developers were being created. So that's where we came up with the Google Summer of Code, and now we have a high-school flavour of that as well. I think we've made a very real impact in creating new people in the open-source world. (Dibona, 2008)

While some of these activities can might be construed as altruistic, they are, in effect, a very persuasive example of stimulating network effects that function as value drivers for the central asset in Google's business model.

Google's strategic and competitive advantages stem from near-total dominance in the search and online advertisement markets. Technologies that drive web usage, “cloud computing” and online services are directly stimulating the economic network that Google has prominent competitive advantages in. Thus, the development of open source technologies helps to strengthen the community surrounding Google's products – in effect allowing everyone free access to the technology without losing any competitive advantages, since technologies, open standards and community are all complementary assets supporting the core competitive dimensions that Google operate on.

What Google successfully experience beneficiary spillovers of the ecosystem and the complementary assets that supports Google's core business and core.
5.1.3 Nokia

Nokia explains how the company's research and development benefit from open source: "Instead of having a closed proprietary system, this is a way of opening the development environment to a larger community," says Auli Luukkanen-Laaperi, who heads Nokia's intellectual-property strategy. She continues with explaining that this open development environment enables Nokia to deliver mobile phones that are more useful to customers, and this in turn translates into more sales. This example illustrates an interesting aspect that open source business models can facilitate increased value capturing. (The Economist, 2005)

5.1.4 Sun Microsystems

When Sun opened Solaris, this was a way to attract independent developers to the platform, which opens it up to innovation from a broader community. Without this sort of ecosystem, the firm itself would be responsible for all enhancements to the product. That would make it not only slower and more costly but probably also less innovative than Linux, with which it competes in certain areas. (The Economist, 2005).

Today Sun is quoted for sharing this insight to the benefits they experienced as a direct consequence of their opening of Solaris “[Sun] discovers 70000 leads per day (based on daily downloads of the MySQL database software) (Interview with Jonathan Schwartz[11])

5.2 The importance of complementary assets

In order to evaluate the problems that the inherent sharing of innovation and technology exhibited by open source technologies pose in relation to Teece's strategic framework, we must consider how Teece's conclusions are affected when the strength appropriability regime approach zero.

Interestingly, the aspect of establishing a constellation of complementary assets that enable a firm to appropriate value becomes the focal point for firms that create business models around open source technologies.

In his summary of the impact of his original article, Teece reaffirms that his focus and the core relevance of the article continues to be the direct link between the innovation and appropriating value from it:

This complete focus on direct appropriation provides the pivot point for the gap between empirical evidence of open source business models and the PFI apparatus. We shall see how this gap can be closed with a more extensive use of complementary assets, which in turn will help us understand the continued validity of traditional business mechanisms:

In the case of zero appropriability which is the domain of OSS, there is little need for restricting access to the innovation apart from an effort to ensure that copyright legislation is enforced where relevant. In this case openness only serves to proliferate the innovation even further. Though this challenges the theoretical assumption for capturing value from technology innovations, another facet of Teece's theory rises to play a more central role to achieve same. As West points out, Teece (Teece, 1986), provides an insight into how the combination of assets can be used to present a unique value proposition that allow for the appropriation of value (West & Gallagher, 2006).

As illustrated the competitive situation of a given firm in open source will be dependent on the specific product market that the firm is competing in. For example Ubuntu Linux is primarily competing in the desktop segment in which Microsoft Windows is the dominant player with near 90% market share (“W3Counter - Global Web Stats,” 2009). On the other hand, Red Hat Linux is primarily focused on server operating systems, in which the competition is more evenly matched (“IDC - Press Release - prUS21856409,” 2009), while Novell is pursuing both segments with SUSE Linux.

While all three firms are competing on the basis of what is fundamentally the same technology, the competitive position is unique the for each firm as a result of a combination of their core business model, strategic focus and strength of their individual constellations of complementary assets.

Conclusively, complementary assets becomes key in appropriating value from an innovation and forms a useful perspective for understanding the competitive prerequisites for open source business models.
To better understand how to build complementary assets that support the desired value capturing we must examine Chesbrough & Appleyard’s seven generic business models for OSS. Unfortunately these seven generic business models are mainly evaluated in relation to how the associated revenue is generated. Chesbrough and Appleyard thus focus on what OSS business models sell and how they generate revenue. However, the issue of the prerequisites for competing with an open source business model are only briefly touched upon, more precisely in context with the mention of the relation between copyright ownership and hybridization business models (Chesbrough & Appleyard, 2007).

Thus, the rest of this chapter attempts to identify some of the prerequisites for building complementary assets and competing with an open source business model or capturing value from technology innovations.

### 5.3 Identifying Complementary Assets in Open Source Business Models

From the previous chapters and empirical data presented in section 5.1 on existing open source business models, I propose a number of complementary assets that seem to influence the competitive position of the individual firm:

- **Legislative environment and project license**
- **Code ownership**
- **Community and R&D**
- **Brand value**
- **Partnerships and alliances**

Each may be relevant in the strategic considerations that are prior to building a competitive base of complementary assets. In the following sections I will examine these factors closer and use real life examples to illustrate their influence, and account for some of the challenges that arise when using them.

#### 5.3.1 Legislative environment

The Legislative environment is determined by the combination of licenses regulating the firm’s core technology and the perceived level of legislative threat to same, i.e. patents. This is consistent with the approach of Teece in the PFI framework.
However, while strong legal mechanisms in the original article is evaluated by how effective it is in preventing imitators from developing or bringing to market a similar innovation, open source software is the inverse to this. In open source, the legal protection is applied to ensure the continues openness of the innovation. In reference to chapter 3, the specific license, regulating the terms under which the technology may be applied and distributed, can establish a fundamental parameter for the firm's competitive position that is exogenous to the firm's strategy.

In his discussion of what defines the strength of an appropriability regime, Teece weighs the major factors determining the choice of strategy as: the nature of the technology and the strength of the legal system protecting the innovation.

The original approach in profiting from innovation puts a lot of weight on the legislative environment role in effectively protecting innovation and capturing value from same. It is an important conclusion that the legislative is no less important for open source technologies, only the purpose is precisely opposite the purpose of the original suggested by Teece.

The license for a given project is a primary factor in determining the ground rules for how the technology of a project may be applied, distributed or incorporated.

Another central point is the adopted policy for assigning or maintaining ownership of contributions. The central mechanism for regulating the use and (re)distribution rights related to the code of a given project is copyright, but the question of ownership over code becomes central for the control of license terms.

A few examples is given of legislative environments in open source business models:

- The GNU project maintains, among other projects, the GNU Core utilities, and provides the operating system supporting the Linux kernel. This is an example of a project which requires contributors to assign copyright to the project in order for the project to be able to control the license terms.

- In comparison, the Linux *kernel* project has adopted a policy of retaining the copyright with the original author, which, among other things, complicates a re-licensing of the project due to the large number of participants.
In reference to chapter 3, copyright is a central mechanism for the control of technological innovation in open source. The choice of open source license and copyright ownership are central to which business models are available. The close link between license and business model created the business need to re-implement specific features of PostgreSQL so firms using PostgreSQL in a proprietary software business model did not infringe the IBM patent.

5.3.2 Code ownership

In the case of MySQL (dual licensing) or SugarCRM (proprietary extensions) both companies require that contributors assign copyright to the company in order to retain the options of re-licensing without having to track down original code contributors. Similarly, the large number of contributors to the Linux kernel renders it a prohibitively large task to change the license terms of the full kernel source.

The assignment of copyright helps keep a “tidy ship” with regards to retaining the option to vary licenses or produce proprietary extensions. However, it has potential to cause concern if major circumstances change the perceived commitment of the company. This was lately illustrated when Oracle purchased Sun, owner of MySQL, which caused concerns regarding the priorities of MySQL under the new ownership (Arnö, 2009).

Thus, the ownership of a project's code determines whether the hybridisation revenue models are applicable for a given technology. This is valid whether the project's code is centralized, as in the case of MySQL, or aggregated, as in the case of the Linux kernel.
This explains the paradox that, as today's technology firms acquire more rights to exclude others in the form of patents, they also feel a greater need to share. They are not doing this for philanthropic reasons, but because intellectual property is so fundamental to their business. As Mr Kelly of IBM puts it: “IP will still be the cornerstone of the successful companies in our industry.” Many more patent pledges, he says, are on the way. (The Economist, 2005)

5.3.3 Community

A critical resource for open source is the strength of the community that emerges and surrounds a given technology or individual project. Communities surrounding technologies can be “spread” over multiple individual projects. One example is Linux, where the end product, or distribution such as Red Hat Linux or Novell SUSE both have communities of their own, partially overlapping the kernel project that is anchored around kernel.org and Linus Torvalds. Also individual applications or utilities usually have participants from a variety of sources, so the developers employed by Red Hat, Novell or Ubuntu may be involved in individual projects centered around these applications.

A viable and active community is generally a good indicator of whether a project is actively developed and can give an indication of how widespread it is used.
• 90 percent of members are watchers, reading all but contributing nothing. We call them the lurkers. Bear in mind, of course, that lurking is a valid method of participation, albeit passive.

• 9 percent of members will contribute answers to direct questions or modify existing information. We call them editors. Editing is a sporadic but regular activity, essential for growing community and maintaining participation.

• 1 percent of members will be actively involved in starting new discussions and generating new material. These are the creators. Creating is fully active participation, and these members are the core of your community. Cherish them.

(Osier-Mixon, 2009)

While the quality, size and efficiency of a community is not easily quantifiable, it is an important indicator on the viability and relevance of a project. Also, community plays a significant role in the capitalisation of open source projects (Rimer, 2005), (Mickos, 2009).

The interaction between a project and the surrounding community becomes central to the success of the project. This especially holds true when relating participants' experience of the firms intentions behind the project, and whether it remains compatible with the goals that helped build the community in the first place (Chesbrough & Appleyard, 2007).

Projects that fail to execute successfully in this respect primarily benefits from open sourcing their products through significant extensions developers or users. This is well illustrated by Sun's OpenSolaris project, where the license of the project became a point of dispute (Wheeler, 2007).
Community Effects:
“[...]and so I made the decision to go to the GPLv2, which has been fantastic for the company in all respects. I mean literally, leads went up, page views went up, downloads went up, our registered community went up in significant percentages. Our sales went up 50 percentage. Our average sales price went up 25 percent, meaning the size of the deals went up. We're now getting a thousand, two thousand leads per week. So everything was positive from it, we've made more money, not less. We have more community, not less. More community involvement. And that was the big driver behind it, is we wanted to license Alfresco under a license that meant that nobody had to be concerned with the company.”

source: “Selling software that sells itself: An interview with Matt Asay”

The importance of efficiently exchanging knowledge with external parties becomes central in conjunction with the community maintenance effort. The resources offered by volunteer efforts or other companies are likely to be allocated in other communities if the continued interaction between the project and the community becomes inefficient or alienated. This means that open source projects must compete for the available resources (Chesbrough & Appleyard, 2007), in the sense that the project must continue to engage and attract community if it is to retain access to external contributions.

5.3.4 R&D as a competitive dimension in open source business
The firm’s absorptive capacity defines how effectively the firm identifies relevant external innovations, applicable to or compatible with the firm’s internal innovation. External innovations might also support other elements of the firm’s business, such as IBM’s adoption of the Apache web server into it’s WebSphere offering (West & Gallagher, 2006, p. 326)

While the term “Research and Development” in 1986 might typically be exemplified by IBM Research and Bell Labs, we have since seen a proliferation of different approaches to the function as well the business models built around different R&D strategies.

Within the Open Innovation paradigm, “(...) external ideas and external paths to market [are placed] on the same level of importance[...]” (Chesbrough, 2006, p. 43) as the more traditional approach dictates.

Research models have shifted from predominantly existing as internal monolithic entities to externally oriented organisational entities interacting and exchanging knowledge outside the firm. (West & Gallagher, 2006, p. 321) argue that building absorptive capacity should be included in the approaches R&D divisions apply in this interaction. This notion of capacity as a critical resource provides a useful approach for defining the requirements for interaction with relevant communities.

5.3.5 Brand Value

One aspect where open source business differs from more traditional strategic scenarios is in relation brand management and value.

While Novell SUSE and Red Hat are trademarks of their respective firms, the Linux kernel is a collaborative project which started without a clearly legal entity to manage the use of the Linux brand itself.

The Linux Mark Institute (“LMI”) was formed in opposition to attempts by parties unrelated to the project to extract license fees for the use of the brand from individual projects (Hughes, 1997), and ownership of the Linux trademark was eventually permanently transferred to Linus Torvalds (LWN & Staff, 2002), (Hillesley, 2007).

The protection of the trademark has necessitated that Linus Torvalds has sometimes felt need to clarify what is done to protect the trademark and why (Torvalds, 2000) (Torvalds, 2005) Today, the LMI provides “free (as in beer), perpetual, world-wide sub-license to approved sub-license applicants. In return, the sublicensee holders must agree not to challenge Linus Torvalds’ ownership of the Linux mark in any jurisdiction, and to provide proper attribution of ownership on their goods, services and elsewhere. “(LINUX MARK INSTITUTE, 2009)"sub-licenses for the Linux trademark.

The protection of the brand of individual projects is central for establishing the purpose and track record for the project. While the decision to trademark the technology of the firm will depend on the specific context, it must be resolved in any strategic considerations and bare in mind the potential challenges of shared ownership.
5.3.6 **Partnerships and alliances**

IBM previously demonstrated that a major technology commitment of one firm can to a significant degree define the paradigm that the rest of the industry works within. This was the case in 1982 when IBM defined the IBM PC standard, leveraging its unique position and strength (Shapiro & Varian, 1999) in the marketplace to introduce a compatibility scheme that established the foundation for the computer usage from then till now.

More firms relied on their hardware vendors for support than on the Linux distribution vendor\(^\text{13}\).

5.4 **Business Models with Open Source Software is “Business as Usual”**

Complementary assets are key in value capturing in a setting where protection options have been reduced. These assets are key to competing successfully and choosing the right constellation of complementary assets is pivotal in defining the competitive strength of the firm. I have given my suggestions for important complementary assets to consider when building competitive business models, compatible with a chosen product market, as defined by Christensen. Teece's framework provides a useful tool-set for evaluating strategic assets of open source firms, although the benefits are established through the surrounding assets rather than the core technology.

From chapter 4 we know that traditional software business models are changing, and I expect that they converge to the same characteristics as OSS business models. In other words, OSS and proprietary software business models are not intrinsically different, but are each determined by the complementary assets available to them. They also fit into Teece's framework, with a few modifications. In other words, OSS is *Business as Usual*.

Additionally, we have seen indicators that OSS can provide new opportunities both in value capturing, and in increased competitive capabilities.

We, furthermore, have a list of possible complementary assets that may serve as a reference for combining appropriate complementary assets and product market into a competitive business model.

\(^{13}\) (Bauer & Robinson, 2004)
6 Conclusion

This thesis has attempted to explain how companies with open source software business models can be viewed in relation to more traditional approaches. I seek to understand how companies can profit from 'free' in co-existence with theoretical frameworks. To understand this, a list of central questions was posed for later answering, achieved by extensive use of what I deemed to be notable examples of the project or company in question.

Open source software is defined by the Free Software Foundation and is characterised by public availability of the full source code, and unlimited distribution rights to same for all people. Licenses are reviewed by the Open Source Initiative that approves licenses that adhere to the definition. Though several license models exist, this thesis has focused on the terms of GPL licenses and variants thereof.

The interplay between copyright ownership and license terms determines the available business models, and how technologies may be applied and distributed in the case of open source. The applications of open source technologies or individual projects are to a wide extent determined by the possibilities provided by the license scheme of the technology, and in turn have consequences for the business and revenue models applicable. The copyright legislation is a prerequisite to ensure the openness of the technology, and thus a central factor in maintaining the freedom to continued development and innovation in open source technologies. Patents pose unique challenges for open source software as they rarely have clear ownership of the software, and disputes are most commonly solved by cross-licensing agreements. In order to address these challenges, the Open Invention Network consortium extends the protection of their pledged patents to projects released under OSI-approved licenses. The original approach in profiting from innovation put a lot of weight on the legislative environment to effectively protect an innovation and strengthen firms ability to capture value. An important conclusion is that, the legislative is no less important for open source technologies, but for the opposite purpose than originally suggested.
With foundation in Teece's framework, I propose some modifications which allow for evaluation of the strategic assets of firms competing in a “weak appropriability regime”. Here, it is impossible to exert restrictions on the access to technology and this is the foundation for open source business models. This shifts the focus almost entirely to the constellation of complementary assets as the new foundation in establishing a defensible competitive position. This shift challenges the role of the complementary assets thus making them key to value capturing.

These assets are defined by Teece in his 1986 article: “Profiting from innovation”. The insights of Chesbrough and West are important in understanding how knowledge traverses between projects, firms and communities in an open innovation regime. Christensen’s framework for open business dynamics provides useful definitions for understanding the variety of product markets. It is also integral in determining the appropriate business model and critical defining the competition for the individual firm.

Based on a set of generic revenue models alongside notable examples of indirect value capture models, I have identified a number of core complementary assets, critical in the strategic positioning of open source firms. I explain their main characteristics and the challenges they pose for firms attempting to utilize them in their business model. Thus competitiveness is ensured by the right combination of complementary assets and product markets, linked by the business model.

The emphasis on complementary assets in open source business models is an interesting difference to traditional business strategies, which for their part face three challenges: decreased lock in effects, lower entry barriers that changes the competitive parameters, and fundamental changes in innovation and asset protection. In essence, however, traditional software business models adhere to the same mechanisms and can be explained using the same framework. I see a trend towards comparable competitive parameters and value drivers for both open source and traditional business models. In essence, there is no differences between open source business and traditional software business from a strategic perspective. Furthermore, open source business models do not preclude the development of proprietary technologies or advantages through tight integration. For the “pure” open source business models, these are not as significant a source of competitive advantage as the successful constellation of assets.
In conclusion, open source, while subject to a peculiar set of rules regarding technology and innovation, is fundamentally “business as usual” when approached from this strategic perspective. The core complementary assets I have highlighted do not exclude more traditional considerations of strategy, but emphasizes the key aspects in open source strategy.

In other words, it appears that the tenet of differentiation as a generic strategy is still relevant in open source. However, the path to differentiation is less product-centric and a to a higher extent defined by the firm’s ability to establish a successful constellation of relevant complementary assets.
A. Notable organisations and definitions

**Groklaw**

[...] a journalistic enterprise, with interviews, research, and reporting of legal events important to the FOSS community, covering – among other things - We cover a number of topics regularly, such as the ODF/ Microsoft Open XML and other standards stories and news about the GPL, DRM, Patents, and we have a page of resources on the Novell-Microsoft patent deal. (P Jones, 2009)

**Open Invention Network**

Open Invention Network is an intellectual property company that was formed to promote Linux by using patents to create a collaborative environment. It promotes a positive, fertile ecosystem for Linux, which in turns drives innovation and choice in the global marketplace. This helps ensure the continuation of innovation that has benefited software vendors, customers, emerging markets and investors. (Open Invention Network, n.d.)

**Free Software Foundation**

The Free Software Foundation (FSF) is a non-profit with a worldwide mission to promote computer user freedom and to defend the rights of all free software users. The Free Software Foundation is working to secure freedom for computer users by promoting the development and use of free (as in freedom) software and documentation — particularly the GNU/Linux operating system — and by campaigning against threats to computer user freedom like Digital Restrictions Management (DRM). (Free Software Foundation, n.d.)

**Electronic Frontier Foundation**

EFF is a donor-funded non-profit and depends on your support to continue successfully defending your digital rights. Litigation is particularly expensive; because two-thirds of our budget comes from individual donors, every contribution is critical to helping EFF fight — and win — more cases. (Electronic Frontier Foundation, n.d.)
**The Software Freedom Law Center**

[...]provides legal representation and other law-related services to protect and advance Free, Libre and Open Source Software (FLOSS). Founded in 2005, the Center now represents many of the most important and well-established free software and open source projects. (The Software Freedom Law Center, n.d.)

**The Open Source Initiative**

The Open Source Initiative (OSI) is a California public benefit corporation, with 501(c)3 tax-exempt status, founded in 1998.

The OSI are the stewards of the Open Source Definition (OSD) and the community-recognized body for reviewing and approving licenses as OSD-conformant.

The OSI is actively involved in Open Source community-building, education, and public advocacy to promote awareness and the importance of non-proprietary software. OSI Board members frequently travel the world to attend Open Source conferences and events, meet with open source developers and users, and to discuss with executives from the public and private sectors about how Open Source technologies, licenses, and models of development can provide economic and strategic advantages.

OSI was jointly founded by Eric Raymond and Bruce Perens in late February 1998, with Raymond as its first president SI was conceived as a general educational and advocacy organization to execute the same mission agreed on at the Free Software Summit. (The Open Source Initiative, n.d.)
B. Selected Characteristics

**Emergent Standards of Enterprise-oriented Linux Distributions**

- A commitment on the part of the Linux distribution vendor to support the Linux distribution and contribute back into the Linux/open source developer community

- A long-term road map associated with the distribution, covering future features and enhancements as well as efforts that will be employed to build and grow the related ISV ecosystem

- A support plan that offers active version support (2–5 years), extended support (5–10 years), and long-term extended support for customers that are unable to abandon an older distribution

- Support that is available on either an 8 x 5 or a 24 x 7 basis in local time zones

- Support that may include regional or country-specific extensions in local languages

- A program in place to work with hardware OEMs to obtain certification of Linux distribution aboard industry-standard hardware products

- Specific set of certification tests that are made available by distribution vendor to determine and qualify both hardware compatibility and application compatibility

- ISV and infrastructure partners that certify applications against distribution vendor–provided test suites

- ISVs and infrastructure vendors that offer specific support programs designed to dovetail into the distribution provider's support offerings

- Partner companies that have an escalation process for problems that are in the core operating system rather than in an application

- Some form of legal protection, including indemnification against intellectual property infringement claims or some form of financial protection in the event of
Strategic strengths of open source software

(in part summarized from Interview: IBM's Adam Jollans)

- Virtual elimination of vendor lock-in; both regarding hardware and software.
- Freedom of choice and relatively low migration costs in case of a switch
- Increased security, rapid response time to security holes and options of auditing code of having third parties audit code without restrictions or NDAs.
- Open Standards ensuring possibilities of integrating solutions across vendor platforms (dependent on specifics)
- Option to tailor solutions to specific needs due to the availability of source code and documentation
- Preservation of continuity and compatibility across versions. Options of continuous support even for very old platforms. Ensured by access to source code.
- Flexibility in architecture and platform choices. Option of integrating new solutions with old architecture.
- Freedom of choice in support providers. Options of changing providers with no inhibitions from proprietary knowledge
- Rapid feature development, due to access to the underlying code.
- Open source software products offer flexibility regarding initial acquisition costs, as well as reduction or elimination of ongoing software licensing costs compared to proprietary systems
- Free choice as to whether to develop solutions/organizational units internally or use external vendors. Extremely broad selection of companies to choose from since relevant information and source code is freely available.

- Risk Management wise, Cost and risks in developing new features no longer have to be born by any single company, since a paradigm already exists for collaboration and new development are largely incremental.

- A tangible benefit for the customer is that the actual software and source code are available no matter what happens to the vendor. This means that the maintenance and migration of legacy systems is detached from specific vendors. In effect, this lowers or negates the entry barriers surrounding operations on these older solutions.
C. Bibliography


IBM. (2005, November 1). IBM's Legally Binding Commitment Not To Assert the 500 Named Patents Against OSS. IBM.


Judge Grants Permanent Injunction against Microsoft in Favour of i4i, Damages now $290 million USD. (2009, August 12).


