IPR Series B:4

IPR UNIVERSITY CENTER
Immateriaalioikeusinstituutti

Rosa Maria Ballardini

Proprietary Software vs FOSS

Challenges with Hybrid Protection Models

IPR Series B

ISSN 1458-9494

P.O. Box 4 (Yliopistonkatu 3)
00014 University of Helsinki
www.iprinfo.com

IPR Series B
ISSN 1458-9494

IPR UNIVERSITY CENTER
Immateriaalioikeusinstituutti

Title	Proprietary Software vs FOSS
Subtitle	Challenges with Hybrid Protection Models
Author(s)	Rosa Maria Ballardini
Series, number, ISSN	IPR Series, B:4, ISSN 1458-9494
Publication date	May 2012
Size	38 pages
Academic thesis	Dissertation (1 essay)
Refereed	Yes (Spring 2012)
Language	English
Publisher	IPR University Center

Abstract:

This article analyses the reasons and consequences of the fact that open source software has become a portion of the technology used by proprietary companies. It focuses on problems arising from the use of what is designated here as the 'hybrid' protection model by commercial companies. The term 'hybrid' model refers to a situation where companies incorporate both open source and proprietary code into the final software they release to the market. The coexistence of both the proprietary and the open source software model is essential to promote innovation in the software field. Due to the different and allegedly conflicting principles under which they are based, however, the relationship within the two systems might not always be peaceful. By combining legal theory and empirical research, this paper provides a comprehensive analysis of the "core" legal challenges surrounding the implementation of the 'hybrid' model in the context of commercial software, and sheds light on the coping mechanisms companies implement in order to navigate such risks.

Keywords:
copyright, patents, free software, licences

Proprietary Software vs FOSS: Challenges with *Hybrid* Protection Models

*Rosa Maria Ballardini**

Abstract

This article analyses the reasons and consequences of the fact that open source software has become a portion of the technology used by proprietary companies. It focuses on problems arising from the use of what is designated here as the 'hybrid' protection model by commercial companies. The term 'hybrid' model refers to a situation where companies incorporate both open source and proprietary code into the final software they release to the market. The coexistence of both the proprietary and the open source software model is essential to promote innovation in the software field. Due to the different and allegedly conflicting principles under which they are based, however, the relationship within the two systems might not always be peaceful. By combining legal theory and empirical research, this paper provides a comprehensive analysis of the "core" legal challenges surrounding the implementation of the 'hybrid' model in the context of commercial software, and sheds light on the coping mechanisms companies implement in order to navigate such risks.

* Researcher in IP law at HANKEN School of Economics; member of the INNOCENT Graduate School in IP law, IPR University Centre, University of Helsinki. The author thanks Professor Niklas Bruun, Professor Marcus Norrgård, Professor Graeme Dinwoodie, LLM Wim Helwegen, and LLM Tanja Liljeström for the valuable comments. Sincere thanks also go to all the companies and people who kindly contributed to the empirical part of this research.

1. INTRODUCTION

This article investigates how firms are developing the relationship between proprietary (mostly copyright and patents) and open source software (OSS/FOSS/FLOSS[1]) under current rules. Specifically, the paper analyses the reasons and consequences of the fact that open source software has become a portion of the technology used by proprietary companies.

In recent years, the proprietary and the FOSS protection models have been increasingly used simultaneously in the same software packages. In fact, almost every company operating in the software field uses both open source and proprietary software. One could argue that the coexistence of both the proprietary and the open source software model is essential to promoting innovation in the software field. Due to the different and allegedly conflicting principles under which they are based, however, the relationship within the two systems might not always be peaceful.

Thus far, research has focused on either FOSS as a phenomenon or proprietary software individually. Very few studies have investigated the relationship and interactions between the two. As the tie between open source and proprietary software models stretches, the need for an in-depth analysis of the ways open source software affects companies' IP policies (and vice versa) is becoming clear.

This article primarily aims at investigating problems arising from the use of what is designated here as the *hybrid* protection model by commercial companies. The term *hybrid* model refers to situations where companies incorporate both open source and IP protected code into the final proprietary software they release to the market. Both the respective advantages and disadvantages of proprietary and open source software are investigated. A practical analysis using company examples is conducted to expose some of the strategies that firms have used to mix open source and proprietary software features together. The analysis is based on a literature review and its intent is to shed light on the major legal challenges involved with *hybrid* models.

The second part of the paper composes of an empirical study in the form of a case study research. A case study analysis was chosen because an in-depth investigation was needed to provide a holistic understanding of the problems. To this end, the case study relied upon a qualitative-type of analysis. A quantitative technique would have probably obscured some of the important information that needed to be uncovered, such as whether the *hybrid* protection method is efficient and what kind of specific problems it involves in practice. The theoretical focus (i.e., the object) of the study was identified as the problem(s) encountered by commercial companies that implement *hybrid* models of protection for their software products. The subject of the study was portrayed by representative companies operating in

1 Open Source Software (OSS), Free and Open Source Software (FOSS), Free, Libre Open Source Software (FLOSS), are all slightly different alternatives used to describe software which can be used, modified and redistributed with little or no restrictions. For the purpose of this paper, however, these terms will be used interchangeably.

the software field. A multiple case study was conducted, as more than one case was available for replication. Several companies use the *hybrid* model and might encounter the problems identified in this study. The study relied upon two different sources of evidence: documents and interviews. The case study was relevant because it provided in-depth answers to the theoretical issues formulated in the first part of the paper. The empirical analysis shed light on the most concrete legal risks associated with *hybrid* models, and on the coping mechanisms used to navigate such challenges.

2. THE PROTECTION MECHANISMS

Proprietary and open source software models possess very different characteristics and work in very different ways. When both mechanisms are used in the same software package, the divergences can lead to unavoidable conflicts and legal risks. To understand where the potential controversies might arise, it is important to first understand the ways they function within this model.

2.1. The Proprietary Model

A major characteristic of proprietary software lies in the way that computer programs are developed. Specifically, the proprietary model favours a centralised, closed type of development where the product is fully "built in-house". As a consequence, the developing firm usually owns and retains all the rights over the software it produces.

Another important feature of the proprietary software model lies in the way software is distributed to users. Generally speaking, licensing is central to the exploitation of all types of intellectual property rights. In software licensing, the IP holders retain ownership, but grant the licensees rights to use the software subject to certain restrictions. The type of restraints placed on users differs between proprietary and open source software, and is one of the main points of contention in the closed-open software conflict.[2]

Proprietary software companies use various types of end user licensing agreements that provide the licensees with limited rights to use the software for specific purposes. For instance, both the copyright and the patent regimes tie the license price into the usage restrictions.[3]

Under the proprietary model (or "closed-code" model) IP owners do not make their source code available to end users, and the product is distributed only in object code form. The rationale behind this trend is that source code contains valuable trade secret information that cannot be protected under the copyright or the patent regimes, and therefore should be kept a secret. Neither of these two legal

2 D Evans, & A Layne-Farrar, "Software Patents and Open Source: the Battle Over Intellectual Property Rights" (2004), 9 *Virginia Journal of Law and Technology* 10.
3 M Välimäki, *The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry* (2005), Ch. 2, at 13-49.

definitions provides protection to the structure, ideas, and logic described in the source code.[4] Therefore, it is important to point out some issues surrounding the debate on the treatment of source code.

First, it should be noted that even though patent rules do not prevent disclosure of the source code, they don't require it either. Additionally, the special nature of software, as well as existing case law and legal dispositions, have all contributed in drastically curtailing the disclosure requirements for software-related patents.[5] Consequently, it has been a common practice to disclose only the minimum amount of information necessary in order to meet patent law requirements.[6] Due to the cumulative nature of computer software and the way the code is constructed, however, access to the source code (or at least to detailed interface descriptions) is often necessary for developing new programs and fostering innovation. This need is particularly evident when (as it is often the case) the software product is a development tool or a component that needs integration adaptability.[7]

The "closeness" of the proprietary model distinguishes it from the open source model, and represents a highly controversial aspect in the debate between proprietary and open source advocates.

2.2. The Open Source Model

The most distinctive features of the open source system are identical to those that distinguish the proprietary model: the development structure, the licensing scheme and its relationship to intellectual property rights, and the policy treatment of the source code.

On the development side, a FOSS peculiarity is the use of collaborative developing structures that extend beyond the boundaries of a single firm. Open source projects are controlled by a community of stakeholders and the software is usually developed by a group of self-organised collaborators.[8]

Nowadays, the term "open source license" describes several different licenses. In order to be recognised as FOSS, licenses need certification from the Open Source Initiative, a non-profit organisation dedicated to promoting open source software.[9] The two most frequently used FOSS licenses are the GNU General Public License (GPL)[10] written in 1989 by Richard Stallman for use of programs released as part of the GNU Project, and the Berkeley Software Distribution (BSD) licence[11], forged by Bill

4 See note 2 above.
5 See D Burk, and M Lemley, "Designing Optimal Software Patents", in R Hahn (eds) *Intellectual Property Rights in Frontier Industries: Software and Biotechnology* (AEI Press 2005).
6 See EPC, Article 83, and 35 U.S.C. § 112.
7 RM Ballardini, "The Software Patent Thicket: A Matter of Disclosure", (2009) 6:2 *SCRIPT-ed* 207, http://www.law.ed.ac.uk/ahrc/script-ed/vol6-2/ballardini.asp.
8 See note 3 above.
9 See http://www.opensource.org/.
10 The version of the license currently in use is version 3, released in 2007. See GNU General Public License, Version 3 (2007) at: http://www.gnu.org/copyleft/gpl.html.
11 See http://www.opensource.org/licenses/bsd-license.php.

Joy from the University of California at Berkeley. The GPL and the BSD have served as models for many other FOSS licenses. In particular, it can be said that they gave birth to two highly influential families of FOSS licenses: the "copy-left" family, derived from the GPL license, and the "academic families", derived from the BSD license.[12] Each FOSS licence is based on copyright law. Specifically, FOSS licenses set the relationship between the copyright holder and the users.[13]

OSS licenses will be analysed in more details later in this paper, however, it is important to note that these licenses clearly differ from any traditional type of IP license. While IP licenses impose restrictions on the use of certain products, the OSS licenses grant freedom to use the licensed software. Specifically, OSS licenses grant the royalty-free right to run, modify, distribute, and redistribute modified versions of the computer program.[14] Another essential characteristic of OSS licenses is the obligation for the licensor to make the source code "freely" available to developers and users but this "zero-royalty" feature of OSS licenses does not necessarily mean that OSS licensors cannot profit from selling the code. OSS licenses are non-exclusive: copyright owners might license their code under an additional license that provides additional services, such as a warranty to the users.[15]

Notwithstanding the common characteristics highlighted above, FOSS licenses differ in several respects. One of the most important distinctions lies in the extent the license allows commercial exploitation, especially with respect to the possibility to combine the FOSS code with a company's proprietary code. The essential nature of the GPL, for instance, is enshrined in a requirement called "copyleft":

> "You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License".[16]

Thus, under the GPL, licensees might be required to disclose their own source code under an open source license. On the other hand, copyleft licenses are not uniform in this aspect. For example the Lesser General Public License (LGPL)[17] and the Mozilla Public License (MPL)[18] are more permissive than the GPL.

Finally, the BSD and academic licenses in general allow more commercial exploitation and more flexibility in combining open source with proprietary software.

It is important to keep all this in mind because (as will be explained later) these differences might be crucial for firms attempting to profit from FOSS products.

---

12 See note 3 above.
13 *Ibid*.
14 *Ibid*.
15 *Ibid*. See also F Lévêque and Y Ménière, "Copyright Versus Patents: the Open Source Software Legal Battle" (2007), 4 *Review of Economics Research on Copyright Issues* 1, at 27-46.
16 See GNU Lesser General Public License, version 2, at http://www.gnu.org/copyleft/lesser.html, "Terms and Conditions for Copying, Distribution, and Modification", 2b).
17 *Ibid*, GNU Lesser General Public License.
18 See Mozzilla Public License Version 1.1. at http://www.mozilla.org/MPL/MPL-1.1.html.

2.3. Closed vs. Open Source

This section compares the proprietary and open source software models to shed light on the benefits and limits each model brings to software innovation. Both systems possess advantages and disadvantages with respect to software innovation, and therefore necessitate finding a balance.

2.3.1. Advantages and Limits of Closed Software

Commercial developers and vendors typically protect their software using many different IP mechanisms, with copyright and patents being the most popular. The reason for this multilayered type of protection lies in the pluralistic nature of computer programs.

Computer programs possess several elements, each of which could fall into different categories of IP laws. Software has a dual nature: on the one hand, it can be defined as a literal work under the form of source code and object code. On the other hand, once executed by a machine it is also a functional object. This configuration allows software to fit within the scope of many different types of IP tools.

The major advantage of proprietary software lies in the fact that a firm can control the destiny of its products. This provides companies with the sufficient profit motive to make their products highly valued by consumers. For example, vendors might try to make their products backwards compatible with earlier versions, so that consumers can experience a smooth transition from an older version to a newer one.[19]

Another strength of proprietary software is that it provides consistent platforms for running applications. Fragmentation is usually not an issue for proprietary software. In an industry like software, where network effects are very strong, this consistency is especially important. For example, developers of proprietary software, such as Windows, write their programs in a manner that allows them to run on all computers meeting their specific hardware and operating system requirements (e.g. Windows ME, Windows 2007, etc…).[20]

Each individual IP protection mechanism possesses both positive and negative aspects with respect to software protection. As previously mentioned, the focus here is on copyright and patents.

For many years, software has been considered a literary work and primarily protected by copyright law.[21] It is undeniable that the software code is expressed "in writing". To some extent, copyright law provides computer code with an adequate level of protection. Copyright prevents third parties from copying the software code

19 See D Evans, and B Reddy, "Government Preferences for Promoting Open-Source Software: A Solution in Search of A Problem" (2003), 9 *Mich. Telecomm. Tech. L. Rev.* 313. Available at: http://www.mttlr.org/volnine/evans.pdf.
20 *Ibid*.
21 See 17 U.S.C. §102(a) (1988); Software Copyright Directive, Art. 1; see also TRIPs Agreement, Art. 10.

(binary and source code) without permission, or using it as an input into a product of their own. On the other hand, however, copyright presents various shortcomings when applied to software. In this respect, the major reasons of concern lay on the copyright scope of protection.

Probably the biggest failure of software copyright is that copyright law extends to the 'expression' of the program in the form of either source or object code, but does not afford protection to the way the program works, i.e. to the program's functions ("idea-expression" dichotomy). On the one hand, by leaving the functional elements unprotected, copyright creates serious risks of under protecting software. On the other hand, there are risks that over-protection might arise, as drawing a border between the literal and functional elements of software is often difficult. This is especially due to the fact that in computer programs there is a significant degree of independence between literal and functional manifestations. The software functions are fully independent from the grammatical (i.e. literal) construction of the lines of code. In other words, even though the source codes of two programs might look completely different, such codes can perform the exact same function and produce the same (or a very similar) set of instructions. This configuration makes it difficult to discern the literal/expressive from the functional/utilitarian in software.[22] Additionally, other problems might arise with software copyright. These can include the long duration of copyright protection with respect to the software's short lifespan, the challenges involved with the definition of "originality" of the code, and various other difficulties deriving from new technological developments (for example, the increased use of modularisation and object-oriented designs in computer programming).[23]

Copyright shortcomings are partially responsible for the adoption of patent protection for computer-related inventions. Historically patent protection was not available for software based on the perception that computer programs were abstract concepts, and as such did not meet general patentability requirements. Current doctrine, however, recognises the patentability of computer software (in particular in the United States and, to some extent, in Europe).[24] By impeding competitors from writing code that includes any patented aspect of the software, patents constitute an efficient mechanism for blocking, or at least obstructing, attempts to duplicate a program's functionality.[25] Patents can protect the implementation of algorithms and other creative aspects of software design.

Notwithstanding these advantages, software-related patents possess various shortcomings. The fact that assessing patentability of abstract technologies such as computer programs raises quality concerns, and potentially leads to issuing of obvious, non-inventive patents in the industry. This can create patent floods, hold up

22 For a general discussion on the issue see RM Ballardini, "Scope of IP Protection for the Functional Elements of Software", in *In Search of New IP Regimes* (2010), Publications of IPR University Center, at 27-62. Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1599607.

23 See J Lipton, "IP's Problem Child: Shifting the Paradigms for Software Protection", (2006) 58 *Hasting Law Journal* 2, at 205-251.

24 For more information see note 7 above.

25 R Jordan, "On the Scope of Protection for Computer Programs under Copyright Computer Programs: The Patent/Copyright Interface", (1989) 17 *A.I.P.L.A. Q.J.* 3, at 199-214.

problems, royalty stacking, and blocking patents. It might actually discourage rather than promote innovation in the field.[26]

Another possible disadvantage of the proprietary software model refers to the treatment of the source code. Specifically, the fact that the source code is kept undisclosed does not allow a technically adept user to fix bugs himself or customize a proprietary program in ways the vendor has chosen to make the program available. In this way, the non-disclosure of the source code might impede direct and indirect competitors, as well as end users, to build upon the program in order to create further developments. Even though these activities do not usually affect the vast majority of home users they might be very important for large customers, and in particular, for promoting further software innovation.[27]

To summarise, the extent of intellectual property protection that computer software should receive is debatable. Proprietary models present various advantages, but also several disadvantages with respect to software innovation. At the same time, however, it is untenable to argue that no IP protection should be available for software.

2.3.2. Advantages and Limits of Open Source Software

Depending on the product, its usage, and the market constraints, OSS has specific properties that can be advantageous or disadvantageous for computer software.

The most remarkable strength of the FOSS model is costs saving. Using FOSS can save both license and development costs. Furthermore, it can save time for the component updates and corrections because more free labour is available to localize and correct defects. It should be pointed out, however, that open source software is not free software, and it often requires substantial investment in order to deploy it in the marketplace.[28]

Another advantage of widely used OSS packages is the generally high quality of the software. The fact that there is a large community behind the projects guarantees that bugs are found and fixed quickly.[29] In this sense, another important reason for using OSS is to attract developers to reduce costs related to having internal support services.

The fact that the source code is made available to developers and users is another remarkable advantage of open source software. Openness can enhance welfare in several ways: it allows others to correct defects and bugs and to customize the programs by adding more features to the software, designing around it, and

26 See M Lemley, and C. Shapiro, "Patent Holdup and Royalty Stacking" (2007) 85 *Texas L. Rev.* 7, at 1991-2050.
27 See note 19 above.
28 M Ruffin, & C Ebert, "Using Open Source Software in Product Development: A Primer", 21 *IEEE Software* 1 (2004).
29 See "Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defence" by *MITRE Corporations*, Version 1.2.04 (2003).

promoting further developments. Furthermore, the openness of the code makes it easier to adapt and reuse it, and ultimately helps software retain value rather than becoming obsolete.[30]

On the other hand, the fact that the code is open can also bring some unavoidable disadvantages, as developers have limited opportunities to earn monetary returns for their investments. Even though non-pecuniary rewards might provide some motivation[31], the limited economic benefits of OSS can reduce the supply of efforts devoted to these activities.[32] For instance, the limited pecuniary rewards in open source projects might reduce firms' incentives to perform costly consumer research into usability and consumer needs.[33]

Furthermore, it is worth noticing that proprietary companies tend to avoid digging into the source code of large OSS projects unless absolutely necessary. Widely used OSS packages usually include very large source codes, therefore making it difficult to try to change parts of such code (especially if the programmer is not very familiar with that particular software). For example, in dealing with bugs in the code most proprietary companies would look for fixes (e.g. whether there is a newer version of the software that has been fixed), or check whether the bug has been reported (and in this way they might find out whether the OSS community is already working on fixing the bug, or whether a fix has been scheduled for future releases). If these options are not available they might decide to find a workaround, which might not mean fixing the bug but rather not using the functions of the software that have the problem. There are cases where the openness of the code is extremely useful. For example, when a company wants to make the binary smaller by taking only part of the source code, or when a company needs to customize the software for internal use. In a nutshell, the fact that the code is open is definitely a positive characteristic of open source software, but it is not the most important one of the model.[34]

From a company perspective, another advantage of OSS is the recruiting process. The fact that extensively used OSS projects develop large communities means that there are many developers familiar with using the software tools related to such projects. Thus, incorporating widely used OSS projects can increase the overall efficiency of the company. If the recruited developers are familiar with the OSS software the company is implementing, they will certainly be more productive.[35]

Theoretically, a fundamental problem with the OSS model is market fragmentation due to the development of multiple, sometimes incompatible, versions of the same software. The extent to which open source users take advantage of their freedom to

30 S Maurer, and S Scotchmer, "Open Source Software: the New Intellectual Property Paradigm" (2006), *National bureau of Economic Research*.
31 See, for instance, K Lakhani, & R Wolf, "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects" (2005) in J Feller, B Fitzgerald, S Hissam, & K Lakhani, *Perspective of Free and Open Source*, MIT Press, Mass.
32 See note 19 above.
33 *Ibid*.
34 Interview conducted on 12 October 2011.
35 *Ibid.*

modify and customize code leads to fragmentation.[36] In practice, however, when OSS is a *de facto* standard component and has a large user community, it provides lasting solutions. The advantage is that developing an application on a proven *de facto* OSS standard provides companies with better protection against changes in a supplier's terms or conditions. With a proprietary solution, changing suppliers is often not possible because the mitigation costs are usually too high.[37]

3. THE RAISE OF THE *HYBRID* MODEL

The above analysis suggests that both open source and proprietary mechanisms are important to promote innovation in the software field. Indeed, the analysis exemplifies not only that each model possesses both positive and negative aspects, but also that the different and often conflicting aims of the proprietary and open source systems can hamper their peaceful coexistence when both models are embedded into a company's final proprietary software product. Discrepancies arise from their different perspectives, spanning from managerial, economic, and technology-related, to purely legal matters.

This article investigates what is here termed the *hybrid* protection model for computer software. In the *hybrid* model companies incorporate both open source and proprietary code into the proprietary software they release to the market. Specifically, the *hybrid* model takes advantage of the resources available through FOSS while adding proprietary features that generate revenue. Potentially, this could provide equivalent value to traditional commercial software at a lower cost and with better quality to the end users.[38]

Specifically, the focus of this paper lies on the legal challenges that can arise from the *hybrid* model and the possible coping mechanisms that can help navigate the risks.

The *hybrid* model includes different companies' trends and strategies. On the one hand, some typical proprietary companies are increasingly including OSS code into their proprietary software. On the other, traditional OSS companies are starting to incorporate intellectual property rights into their model. Furthermore, other companies were born as "pure hybrids" in the sense that they have included both OSS and proprietary features into their models from their beginnings.

Corporations have launched various strategies as part of these transformations. The following paragraph provides an overview of some of the mechanisms used by selected firms operating in the software industry. The companies chosen are highly representative because they are among the biggest corporations operating in the computer programs field and because they have been among the first to implement the *hybrid* model into their business. This analysis sheds light on the 'core' legal risks that might arise from using the *hybrid* model. It should be noted, however,

36 See note 19 above.
37 See note 29 above.
38 M Karels, "Commercializing Open Source" (July/August 2003), *QUEUE*, at 4755.

that there are several ways to develop *hybrid* models and the following samples represent but a small portion of them.

Three different groups of companies are identified:

1) Traditionally commercial (i.e. proprietary, "closed") companies that have modified their fully proprietary protection model in order to incorporate open source software. Within this group, the strategies implemented by International Business Machines (IBM) are analysed;

2) Traditionally FOSS firms that have started to include closed-types of software. Among this group the Red Hat's model is investigated;

3) Purely *hybrid* companies, i.e. software firms that have started as a mix of closed/open features. MySQL AB (nowadays owned by Oracle Corp.) is taken as a representative example for this group.

3.1.1. Closed Companies Going Open

Proprietary software companies have found that they could generate more profit and better satisfy their customers by including aspects from open source. Accordingly, they have embraced an approach based on honoring FOSS while, at the same time, relying on fees for the use of intellectual property.[39]

3.1.1. IBM: The Pioneer

IBM was one of the first proprietary companies to change its protection model and move to a *hybrid*, closed-open system.

In the 1980s, IBM was one of the most vigorous advocates of strong IP protection for computer programs. They believed that without strong IP rights, there would not have been sufficient incentives for firms to invest in software development. At that time, IBM was distributing programs merely in machine-executable form (i.e. object code form) and was using several proprietary protection mechanisms, such as copyrights, patents, trade secrets, trademarks, licensing agreements, and technical protection measures.[40] Certainly, at that time no one would have guessed that two decades later IBM would have embraced open source software.

However, IBM re-oriented themselves as an open source company, albeit to a limited degree. It currently contributes over one hundred million U.S. dollars a year to

39 See C Nosko, A Layne-Farrar, & D Garcis Swartz, "Open Source and Proprietary Software: The Search for a Profitable Middle-Ground" (2005). Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=673861.

40 See P Samuelson, "IBM's Pragmatic Embrace of Open Source", *Communications of the ACM* 49 (10), 2006. Available at: http://escholarship.org/uc/item/4xb4t1ps.

the development of Linux and other open source software projects.[41] With such a radical reversal, one cannot help but wonder what the possible motivations could have been. The answer, however, is not very straightforward and includes various considerations.

3.1.1.1. Possible Justifications

Probably the most obvious justification behind IBM's change of protection model relates to the relationship between IBM and its main competitor, Microsoft. Many companies give the "kill Microsoft" approach as a reason to pursue open source. Accordingly, companies are willing to invest in software development that might not generate monetary returns, as these efforts might impede Microsoft's ability to extract future profits by monopolizing markets.[42]

During the eighties IBM was the dominant firm in the software industry. When it entered the market for personal computers IBM decided not to build its own proprietary operating system, but to license it from Microsoft, who was a small firm at the time. In order to ensure a steady supply of programs for the PC platform, IBM required Microsoft to make interface information available to application developers.[43] The IBM PC was a huge success and soon became an industry standard.

This allowed other vendors to offer equivalent technologies, but all were required to interoperate with software created for the IBM PC. In other words: all the equivalent technologies were running Microsoft's operating system.[44] Taking advantage of this situation, Microsoft started to license its operating system to PC developers to encourage network economies. This enabled Microsoft to obtain monopoly power for its platform. Meanwhile, Microsoft started to develop Windows 3.x and soon launched Windows 3.0, which immediately became a phenomenal success in the marketplace. Microsoft's platform soon became a *de facto* industry standard.[45]

Since then, no operating system has been developed that could compete with Microsoft's offering(s). Linux is one of the first operating system with real potential to challenge Microsoft's position.[46] Therefore, investing in Linux not only allows IBM to be independent from Microsoft's licensing terms, but it also increases the chances that Linux will succeed in its competition with Microsoft in the operating system market.[47]

---

41 H Chesbrough, "The Era of Open Innovation" (2003), 44 *MIT Sloan Mgmt Rev.* 35.
42 See R Mann, "Commercializing Open Source Software: Do Property Rights Still Matter?", 20 *Harvard Journal of Law & Technology*1 (Fall 2006).
43 See note 40 above.
44 *Ibid*. See also H Chesbrough, "Open Innovation. The New Imperative for Creating and Profiting from Technology", *Harvard Business School Press* (2003), at 93-112.
45 See P Capek, S Frank, S Gerdt, and D Shields, "A History of IBM's Open Source Involvement and Strategy", (2005) 2 *IBM Systems Journal* 44, at 249.
46 Indeed, other players are present, for e.g. the Mac OS, the iPhone OS, the JavaME, and the Android operating systems. See Operation Systems market share, September 2011 at: http://www.netmarketshare.com/report.aspx?qprid=8&qptimeframe=M&qpsp=152.
47 *Ibid*.

IBM fully entered the software market in the mid-1990s[48]. However, IBM's success was undermined by various factors, not only Microsoft's platform dominance in key platform's markets, but also by the fact that software started to be mass-marketed and became increasingly commoditized.[49]

Acknowledging these problems, IBM began thinking of some alternative business models in order to succeed and quickly discovered that what customers wanted were open standards, interoperability, and customization tailored to their own needs.With these priorities in mind, IBM concluded Linux was a better platform to meet customers' demands than traditional proprietary operating systems. Thus, it embraced a semi-open protection model accordingly.[50]

At the base of this shift in strategy towards open innovation are some of the most common reasons associated with the use of open source software. Among those, monetary advantages are probably the most important: OSS is less expensive than proprietary software and thereby reduces the overall cost that customers pay for IBM's software.[51] Furthermore, through an open innovation strategy, IBM can distribute the costs of designing, developing and improving software among many contributors. Consequently, there is less need for internal support services. In OSS projects the customers can become part of the development team, as they are willing to invest time, money, and energy on improving the software, fixing bugs, and making the code more robust and extendable. This type of distributed collaborative development is particularly relevant in a technological field like software, where the increased complexity of the technology involved and the need to integrate programs from various sources is essential to create more efficient systems.[52]

Linux provides also a common platform on which IBM can build and sell special applications and services. Consequently, IBM currently focuses on selling complementary hardware and software running on top of Linux (and other open source programs), as well as integrating other customized services to enterprise customers.[53]

Moreover, IBM sells additional complementary proprietary software to the open source product.[54] The complementary software is usually a proprietary-type of software that works in conjunction with the open source software.

Finally, an open innovation strategy allows IBM to study others' innovations allowing them to perceive opportunities for building new technologies on the open source base.[55]

---

48 It should be noted that software was not at the core of IBM initial business. Instead, IBM initially focused on developing software merely in order to sell hardware.
49 See note 40 above.
50 *Ibid*.
51 J West, and S Gallagher, "Challenges of Open Innovation: the Paradox of Firm Investment in Open-Source Software" (2006), 3 *R&D Management* 36, at 319-331.
52 *Ibid*.
53 See note 40 above.
54 See note 39 above.
55 See note 40 above.

3.2. OSS Firms Implementing IP Features

The "hybridization" process of software protection not only involves major changes from proprietary firms, but it also affects the purely open source companies. Recent years have seen traditionally open source firms starting to incorporate IP mechanisms into their protection models. Whether their motives are based on enhancing profit or whether (as they often claim) they are only a defensive tactic implemented to cope with the increasing threat from proprietary companies is an open question. Some answers can be found by analyzing the type of mechanisms implemented in these companies' practices. To this end, the next section considers the example of Red Hat.

3.2.1. Red Hat: The Pinocchio Approach

Red Hat is an American company and a well-known Linux distribution vendor. Red Hat began as a purely open source software company, but has gradually changed its policy to incorporate proprietary features, in this way embracing a *hybrid*-type of protection model.

The goal for Linux distributors is to solve one of the major problems of the Linux development model: overcoming the inconvenience of modularity.[56] Distributors do not integrate and sell the Linux components as a single package like proprietary operating systems. Instead, the individual components "tend to float around". This is due to the fact that Linux (as well as most FOSS projects) does not have one single developer, but rather group contributors. Therefore, Linux distributors, like Red Hat, consolidate these pieces in a convenient package and sell them to the consumers.[57]

Originally, Red Hat concentrated on generating revenue through supporting services and packaged products containing a manual and a CD to facilitate installation. By paying for just one Red Hat Linux package the clients also acquired the right to install it on as many computers as they wished. This type of policy, however, brought a problem typical to many open source companies [58]: inadequate revenues. To overcome this problem, Red Hat drastically changed its strategy and implemented a model that combines features of the open source and proprietary frameworks.

In 2003 the company split distribution into two different products: the Fedora project[59] and the Red Hat Enterprise Linux (RHEL)[60]. Fedora is a traditional open source project, and is to run experiments and for outside developers to submit code. RHEL is a Linux distribution system produced by Red Hat and targeted to the

56 *Ibid*.
57 See R Gabriel, & R Goldman, "Open Source: Beyond the Fairy Tales" (May 2002), *Perspectives on Business Innovation*, Ernst & Young, Issue 8.
58 MandrakeSoft that distributed Mandrake Linux, and SCO-Caldera, that distributed OpenLinux, are but two of many examples of OSS companies that failed due to revenues issues.
59 See http://fedoraproject.org/.
60 See http://www.redhat.com/rhel/.

commercial market. The code developed in Fedora can be included in RHEL at Red Hat's discretion. Even though the RHEL's source code is made available, the code computers need to run the operating system is conditional on purchasing a support subscription. Additionally, following a typically proprietary model, a support subscription needs to be purchased for each computer.[61]

Red Hat has also started to use trademarks as a protection mechanism. A major problem for companies trying to generate revenue from software licensed under the GPL is that because the source code is freely available, third parties can easily "compile" it for computers to read, and then resell it without having to bear the development costs.[62] To reduce their competitor's ability to obstruct its business interests, Red Hat makes use of trademark protection: another company could rebuild RHEL from freely available source code, but it would have "to strip out" all references to Red Hat to comply with trademark law.[63]

Finally, another intellectual property mechanism that is incorporated into Red Hat's model is patent protection.[64] Examples of the patents owned by the company within the field of computer programs include: the European patent EP1312195 "Method and apparatus for handling communication request at a server without context switching", the EP1691276 "System and method for verifying compatibility of computer equipment with a software product", and the EP1659493 "Replacing idle process when doing fast messages". In the U.S., Red Hat's patents consist of US2011066672 "Transaction Sticky Load Balance Policies", the US2011067007 "Automatic Thread Dumping", and the US2011249013 "Plug-in Architecture for Dynamic Font Rendering Enablement" amongst numerous others.

This move came as highly unexpected, particularly considering the company's well-known objections to software-related patents. The company has tried to reassure the public by stating that these patents are mainly for defensive purposes and used as a 'trade off' with proprietary firms that are constantly threatening them for allegedly infringing their patents. Accordingly, their policy is not to enforce those patents upon open source developers. This message is clear in the company's statement towards software patents:

> "Red Hat has consistently taken the position that software patents generally impede innovation in software development and that software patents are inconsistent with open source/free software. [...] At the same time, we are forced to live in the world as it is, and that world currently permits software patents. A relatively small number of very large companies have amassed large numbers of software patents. We believe such massive software patent portfolios are ripe for misuse because of the

61 See note 57 above.
62 J Lerner and J Tirole, "Some Simple Economics of Open Source", 50 *J. Industrial Economics* 2 (2000), 197-234.
63 See note 39 above.
64 See also Red Hat press release (June 3, 2005), "Red Hat Calls for Intellectual Property and Patent Policy reform; Red Hat Commits Significant Resources Towards Fedora Foundation, Global Reform of Government Public Policy and Advocates as Patent Commons", at: http://investors.redhat.com/releasedetail.cfm?ReleaseID=355725.

> questionable nature of many software patents generally and because of the high cost of patent litigation. One defence against such misuse is to develop a corresponding portfolio of software patents for defensive purposes. [...] At the same time, Red Hat will continue to maintain its position as an open source leader and dedicated participant in open source collaboration by extending the promise set forth below. [...] Subject to any qualifications or limitations stated herein, to the extent any party exercises a Patent Right with respect to Open Source/Free Software which reads on any claim of any patent held by Red Hat, Red Hat agrees to refrain from enforcing the infringed patent against such party for such exercise ("Our Promise").[...]".[65]

This is but a promise without any legal force and it does not exclude *a priori* the possibility for Red Hat to change its strategy at any time. Perhaps (temporary) one can find security in the general assumption that if companies enforce patents for competitive reasons (as they often do), it is not in Red Hat's interest to enforce its patents. The company would attract too much negative attention compared to the potential benefits of such action. As already mentioned, however, conditions can change and so can Red Hat's policy.

3.3. Pure *Hybrid* Firms

3.3.1. MySQL AB: The Dual Licensing Approach

MySQL is one of the world's most popular open source database systems. Originally owned by the Swedish company MySQL AB (now owned by Oracle Corp.), MySQL's *hybrid* nature lies in its original licensing model. The company used a system of "dual licensing" by combining proprietary and open source licensing models.

Generally, the dual licensing model mixes together proprietary and OSS mechanisms, and offers the same software product under both a traditional proprietary license and an open source license.[66] Technically, only one core product exists, but two licenses are used: one for free distribution and use, and another for proprietary distribution.[67]

With dual licensing anyone can download the source code for free and redistribute it, just so long as the redistributed product is licensed under an OSS license. The second license removes the open source license's restrictions and allows purchasers to distribute it and integrate it with proprietary products. This second option obviously targets companies planning to customize the product for commercial purposes. Thus, both licenses allow developers to customize MySQL and redistribute it as part of

65 See Red Hat, Inc., "Statement of Position and Our Promise on Software Patents" at http://www.redhat.com/legal/patent_policy.html.
66 S Comino, and F Manenti, "Dual Licensing in Open Source Software Markets" (January 1, 2010). Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=985529.
67 M Välimäki, "Dual Licensing in Open Source Software Industry" (2003), 8 *Systemes d'Information et Management* 1, pp. 63-75.

a larger product. If the larger product is released as open source, no license needs to be purchased. If the product is distributed in proprietary form, it requires a commercial license.[68]

The dual licensing system (and models where OSS is fully embedded and distributed together with proprietary software in general) includes two major and interconnected legal risks. One danger is that the OSS license (in the case of MySQL, the GNU General Public License) can dilute ownership and even eliminate the possibility to dual license, and the fact that OSS projects with multiple authors can have conflicting copyright claims poses the other.

The ability to license the product with terms other than open source, however, requires full ownership of rights to the product. Thus, no hidden liabilities in the form of code contributions from unknown third parties should remain.[69] To clear the rights and avoid legal risks, MySQL adopted a strategy that allows the firm to fully own the copyright over both the product and over all its modifications: MySQL not only develop almost all code in-house, but it also rewrites all the bug fixes and further extensions and modifications offered by external developers.[70]

Another way MySQL profited from the dual licensing system was by taking advantage of the "network effects" typical of database systems. In most cases, databases need customization to meet the specific needs of buyers. The more people use a particular package, the more developers become trained in customizing the system, inevitably leading to more written documentation and the creation of more software add-ons. This makes the system more valuable to each user, since trained developers and convenient add-on packages are easier to find and cheaper to use.[71]

In the specific case of MySQL, the GPL licensing option encouraged network effects by creating a group of developers familiar with the product. At the same time this attracted the interest of commercial users willing to pay, at least in part, because of the network effects created by the free licensing for non-commercial use under the GPL. Even though some revenue is certainly lost, (because users who download the GPL version would have, if required, purchased a license) the lost unit sales can be recouped through the higher prices charged to commercial users. In this way, MySQL utilizes price discrimination to effectively separate users according to product usage.[72]

68 See note 39 above.
69 See note 62 above. The issue on ownership of rights is also explained in more details later in this paper, under section 4.1.3..
70 *Ibid*.
71 See note 39 above.
72 *Ibid*.

In a recent press release, Oracle Corp. announced the addition of certain close-type features to the MySQL code.[73] This is a clear sign that things are changing and the MySQL code is not any longer fully open source software.

4. THE *HYBRID* MODEL FACES ITS LIMITS. SOME EMPIRICAL EVIDENCE

The analysis on the companies' strategies thus far shows that the *hybrid* protection models can lead to several legal risks. Companies utilize open source software either internally (for internal use) or in commercial products or services. Indeed, the legal challenges increase when a company fully embeds and releases open source code together with proprietary software. Solely internal use of open source software, such as use as part of an internal tool, is usually safe from a legal perspective. Internal use means that there is no distribution and that no FOSS code is incorporated into the company's final product.

Internal use can still be compromised by something as simple as an external contractor having access to the code. Legal risks become real once the FOSS program is distributed to a customer, contractor, or otherwise made available. They become concrete once the open source software incorporates itself as a component or as a library of the proprietary software developed by the company. This situation is particularly interesting because it raises a series of challenging legal questions. Specifically, this article analyses both the legal risks related to the FOSS licenses (i.e., compliance and compatibility with the FOSS licensing terms, risks associated with the 'copyleft' clause, and ownership of rights related risks), and the risks related to intellectual property rights (i.e., patent infringement and litigation risks, dilution of the company's own patent rights, and lack of warranties and indemnifications on the FOSS side). The focus is on the risk(s) of the simultaneous use of FOSS and proprietary software by proprietary companies, and not on the risks posed by IPRs (especially patents) to the FOSS development. This last issue has been extensively discussed in literature, and therefore, it is not the intention of this paper to further reiterate it.

Even though this article addresses the legal challenges involved, it is worth mentioning that other risks can arise from using *hybrid* protection models. For instance, the technical risks related to the OSS code (like those related to the function of the software), the security risks, the availability of support, and the compatibility of files and formats. Another highly concrete risk is the so-called "open source community risk": if a company is perceived to "violate" (in a broader sense than legal infringement) certain OSS license conditions, or even the "spirit" of that license, or goes against the generally accepted conventions of the OSS community in another way, the community may react against the company itself. The community can include not only active participants of the open source movement, but also employees of the company, employees of the company's clients, or the suppliers of the company. Overall, the community risk can attract lot of negative publicity to the company.

73 See Oracle's MySQL Blog, "New commercial extensions for MySQL Enterprise edition", at: http://blogs.oracle.com/MySQL/entry/new_commercial_extensions_for_mysql.

An empirical study under the form of a case study research composes this section. A case study analysis was chosen because an in-depth investigation was needed to provide a holistic understanding of the phenomenon. To this end, the case study relied upon a qualitative-type of analysis. A quantitative technique would have potentially obscured some of the important information that needed to be uncovered, such as whether the *hybrid* protection method is efficient and what kind of specific problems it poses in practice. The theoretical focus (i.e., the object) of the study was the problem(s) encountered by commercial companies implementing *hybrid* models of protection for their software products. The subject of the study was exemplified by representative companies who operate in the software field. A multiple case study was conducted, the reason being that more than one case was available for replication: several companies use the *hybrid* model and may encounter the problems investigated in this study.[74]

The nature of the project and the type of research questions investigated justified an 'intrinsic' and 'collective' case study. In other words, a study where the researcher has a personal interest in the case, and where a group of cases or objects are studied. The case study intended to generate new understandings, rather than answer one (of a few) specific question.[75] The questions posed to the companies included:

- *General questions,* such as 'how' do companies combine FOSS code with proprietary software? 'What' are major advantages and disadvantages for incorporating *hybrid* models (FOSS in particular) with a company's proprietary model? 'What' are the reasons for choosing certain FOSS packages? 'What' are the reasons for a company's customers for specifically asking not to incorporate FOSS pieces of code into the company's proprietary software?

- *Specific questions* on the potential *legal risks* of using *hybrid* models, such as 'how' difficult is it to review and interpret the OSS licenses terms? 'How' do these difficulties challenge compliance with all licenses used by a company? 'How' concrete is the risk of "contaminating" a company's proprietary code that becomes associated with the 'copyleft' clause, and for 'what' reasons? 'How' concrete are legal risks from the fragmentation of rights (for example, copyright) in FOSS projects? Is FOSS' rights fragmentation an advantage or disadvantage for a proprietary company implementing *hybrid* models? 'How' do companies perceive the risk that certain FOSS packages might expand to include proprietary applications? 'How' much, and in 'what' way does the incorporation of open source code into a company's proprietary software affect the risk of IP (in particular patent) infringement and litigation? 'How' concrete is the risk for the dilution of a company's patent rights, and

74 For more information on case study research see: R Yin, *Case Study Research: Design and Methods*, Volume 5 of Applied social research methods series, Sage Publications, Inc., Forth edition (2008); R Yin, *Applications of Case Study Research*, Sage Publications, Inc., Third edition (2011). See also R Stake, *The Art of Case Research*, Sage Publications, Inc., First edition (1995). See also W Tellis, "Introduction to Case Study", 3 *The Qualitative Report* 2 (1997) at: http://www.nova.edu/ssss/QR/QR3-2/tellis1.html; W Tellis, "Application of a Case Study Methodology", 3 *The Qualitative Report* 3 (1997), at: http://www.nova.edu/ssss/QR/QR3-3/tellis2.html.

75 See, for instance, Stake (1995) above.

what factors require consideration when addressing the problem? 'How' relevant is it that several FOSS products do not provide warranties or indemnifications when a company decides to choose between a proprietary and FOSS version of a component, and what are the justifications behind each option?

- *Specific questions* on possible *coping mechanisms* to navigate the legal challenges associated with *hybrid* models, such as 'what' kind of policy procedure and processes should a company implement to review the licenses it plans to incorporate? To 'what' extent does the risk of contamination of a company's proprietary code associated with the 'copyleft' clause affect the decision of a company to incorporate GPLed code (or similar types of licensed code) or not? 'What' kind of coping mechanisms can (if any) a commercial company using *hybrid* models implement to reduce the risks associated with the fragmentation of rights in FOSS? 'How' can a company effectively monitor and prevent the possible infringement and litigation risks associated with the *hybrid* models in general, and with the incorporation of open software code into the company's final proprietary software in particular? 'How' can a company effectively monitor and prevent possible dilutions of its own patent rights when it releases software under FOSS licenses?[76]

To select the subjects of investigation, an information-oriented technique rather than random sampling was used. [77] The companies were chosen for their representativeness with respect to the overall purpose of the study's research objective (i.e., they were 'key' cases) and to maximize what could be learned in the period of time available for the study. Specifically, the study considered six cases, composed of one consultancy company and five software companies. These specific cases were chosen for the following reasons:

- The field of operation of the companies: they were all companies whose main business lay in software, i.e. companies whose innovation was based on the software they developed as opposed to companies that used software indirectly. The consultancy company operated specifically in the field of FOSS for proprietary companies, and therefore, considered a valuable source of insightful information from a more neutral perspective.

- All the companies implemented or dealt with *hybrid* models of protection.

- Key people working at these companies were well-informed experts on the research object and in possession of important knowledge.

The size of the companies and their geographical areas of operation were not regarded important factors in the selection of the cases.

76 For more information see Appendix V: "Interview Questions, Software Companies Implementing *Hybrid* Protection Models".
77 See note 74 above.

The study used two different sources of evidence: documents and interviews. Other sources of evidence usually considered in case study research such as, archival records, direct observation, participant observation, and physical artifacts were not relevant for this study.[78]

The documents used were mostly academic literature (legal and economics), case law, legislations, publicly available companies' policies in the fields of intellectual property and open source, companies' websites, and newspaper articles.

Interviews were the most important source of information for the study, and followed an open-ended format.[79] All the interviews were conducted during the autumn of 2011. Key respondents were asked to comment on the research questions from the perspective of their own company, but also and more importantly, based on their extensive knowledge of the field. The respondents were free to propose solutions or provide insights into the subject matter, as well as to corroborate evidence obtained from other sources. Indeed, this 'open' method expanded the depth of the relevant data gathered.

A draft report was written based on both the documents consulted and the answers received in the interviews.[80] All the participants in the study then reviewed the report to verify the accuracy in reporting their answers, as well as the overall conclusions and observations. Several external peers then critically challenged the results and provided relevant feedback in a discussion regarding the report. This process enhanced the accuracy of the case study.

The anonymity of the people interviewed and their respective companies was necessary due to the participants' consideration of the topic as being both controversial and confidential. As a compromise, a cross-case analysis was composed instead of a single-case report.[81] The case study report does not portray any single one of the companies interviewed, but rather a synthesis of the lessons learned from all of them. Accordingly, none of the cases are presented as single-case studies. Instead, examples from the cases are discussed under each research topic section (i.e. 'Compliance and Compatibility', 'Reciprocity of FOSS Licenses', 'Ownership of Rights', 'Patent Infringement and Litigation', 'Dilution of the Company's Own Patents', and 'Lack of Warranties and Indemnifications'). Furthermore, there are two sub-sections in each section: 'Concerns' and 'Coping Mechanisms'. Under the section 'Concerns' theoretical frameworks are discussed, while the 'Coping Mechanisms' section presents data from the empirical study. In addition, the research topic sections 'Reciprocity of FOSS Licenses' and 'Patent Infringement and Litigation' include two additional sections: 'Free and Open Source Software Litigation' and 'Patent Litigation on FOSS'. Relevant case law on FOSS and copyright and on FOSS and patents is discussed in these sections.

78 *Ibid*.
79 *Ibid*.
80 See Yin (2008), note 74 above, at 141-167.
81 *Ibid*, at 170-173.

4.1. Licensing-Related Concerns

A major set of legal problems associated with the *hybrid* protection model includes licensing-related issues. The following paragraphs focus on compliance with the OSS licensing terms and the compatibility among the licenses, the issues related to the 'copyleft' clause, and the ownership of rights-related problems.

4.1.1. Compliance and Compatibility

Concerns

One main point of concern with the *hybrid* model is the compliance and the compatibility between the OSS and the proprietary licenses' terms and conditions. Two separate sets of problems should be distinguished: on the one hand, problems might derive from the use of both proprietary (either third parties' proprietary software or the company's own proprietary licensed code) and OSS licensing models; on the other hand, incompatibilities might arise between OSS licenses themselves. Although the increase in the number of open source licenses has improved firms and project leaders ability to find models that better suit their needs, such a multiplication has also led to the creation of licenses that are incompatible with others. Ironically, this might distort the original purpose of open source software by limiting, rather than encouraging, the reuse of code.

Coping Mechanisms

The companies interviewed cited compliance with the obligations imposed by licenses to be the biggest and most concrete reason for concern. The basic difference between commercial licenses and FOSS licenses is the fact that with the FOSS licenses the terms cannot be changed, while in a commercial set up, there is usually some room for negotiation on the licenses' terms. To minimize the license-related risks all the companies agree that it is extremely important to build a solid design of the architecture and principles around the types of licenses the company plans to incorporate (e.g. where should the company include GPLed components and where it should incorporate components licensed under more permissive OSS licenses). In accordance with this consensus, all the interviewed companies affirmed to conduct some sort of compliance and compatibility checks of the licenses included in their models. The medium and large companies all had an internal procedure for assuring compliance of the licenses in use. Even though the start-ups and small companies interviewed did not have well-structured policies on the matter, they usually had at least one person in-house responsible for scrutinizing the licenses' compatibilities and compliance with the licenses terms.

According to some of the companies, a very important requirement for choosing certain OSS packages instead of others is the compatibility of the OSS licenses with each others, as well as third parties' proprietary code and the company's own proprietary software licenses. A common problem most of the companies reported in the licenses' review process was the difficulty in interpreting the OSS licenses'

terms. This was attributed to several reasons: the language of the licenses *per se*, the fact that multiple versions of the licenses can exist for the same piece of code, thereby making it difficult to detect which version actually applies, and the fact that the same OSS code is often licensed under several different OSS licenses, often stating different rules and making it challenging to understand the applicable principles.

Only some of the firms acknowledged conducting further checks on the licenses' terms every time the FOSS packages receive updates. One company was very aware of risks involved with possible changes to the licenses in the FOSS packages. While reviewing the licenses included in an FOSS package the company was using, the company discovered the package was extended with certain proprietary features. As the company was using these features, it found itself in a situation of either having to change the FOSS package or having to pay the license fees for the proprietary parts of the code. As the particular component was not an essential part of the company's product, the decision was made to remove the full package and substitute it with another FOSS component. The company reported that after this experience, they introduced a more thorough review of the licenses' terms.

4.1.2. Reciprocity of FOSS Licenses

Concerns

The license restrictions of the open source software have a clear impact on a proprietary company's strategy. One of the biggest risks is that a companies' proprietary code will be "forced" open. The risk appears particularly high with the 'copyleft' licenses in general, and with the GPL in particular.

The 'copyleft' licenses are typically quite restrictive when it comes to combining proprietary and open source types of software. According to the GPL, if a piece of code that in whole or in part contains or is derived from an OSS code or any part thereof is distributed or published together with another (proprietary) software, the source code of the entire final product must be made available and licensed under the terms of the GPL.[82] In other words, if a company combines GPLed software with its own developed proprietary software, the question comes down to whether or not the result is published "as a whole work". Technical issues will probably determine what is considered as "a whole", like how closely the programs interact, how they are linked together, and how the proprietary program loads with the GPL-licensed code. On the other hand, if the proprietary code that a company combines with the GPL code is apparent and recognizable as an independent and separate work, such code might be able to remain free of the GPL "taint".

82 See note 10 above.

Furthermore, the GPL includes also specific patent clauses[83] with the intent "to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary"[84]. In other words, incorporating code originally acquired under a GPL-type of license might dilute possibilities for commercialization and ultimately compromise the company's IP rights.

The BSD and academic licenses give more flexibility and opportunity for commercial exploitation. Under the BSD-type of licenses, licensees can generally distribute their derivative works without any obligation of source code disclosure. This means that licensees are free to integrate FOSS code (as well as modifications of the code) into proprietary software, and then redistribute the whole piece of software under proprietary licenses.

Free and Open Source Software Litigation

Problems related to the enforceability of the FOSS licenses have become a hot topic of discussion in recent years, especially with the uncertainty surrounding the application of the FOSS licenses' principles. In an already risky and difficult corporate environment, companies are attempting to minimize their potential liabilities, and one way of doing this is by reducing the companies' legal risks.[85]

Most GPL cases thus far have been on copyright issues focusing on failures to comply with the source code's distribution requirement. In general, the retention of the copyright for the original code allows the enforcement of the FOSS licenses: the original developer retains the copyright for the original program and subsequent developers retain the copyright(s) on their improvements. If one does not comply with the license it terminates and it becomes impossible to copy, modify, distribute, or redistribute the code without violating the owners' copyrights.[86]

In the US, court cases on the interpretation of the FOSS licensing terms started to appear in early 2000. For instance, *Palnetary Motion vs. Techsplosion* (2001)[87], *Progress Software Corp. vs. MySQL AB* (2002)[88], and *Computer Associates vs. Quest* (2004)[89] all centered on the enforceability of the GPL licensing terms. Even though they followed different paths, all decisions presumed that the GPL terms are binding.

---

83 For instance, see clause 7 of the GNU General Public License, version 2: http://www.gnu.org/licenses/gpl-2.0.html and clause 11 of the GNU General Public License, version 3: http://www.gnu.org/copyleft/gpl.html.

84 The preamble of the GPL version 2 explained the motivation behind the patent clause: "[...] every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free." See GNU General Public License, version 3.

85 See A Guadamuz, "Legal Challenges to open source licenses", (2005) 2:2 *SCRIPT-ed*, 301-308.

86 See GNU General Public License, Version 3 (2007), note 10 above.

87 *Palnetary Motion, Inc. vs. Techsplosion, Inc.* (2001) United States Court of Appeal for the Eleventh Circuit 261 F.3d 1188.

88 *Progress Software Corp. vs. MySQL AB* (2001), Civil Action No. 01-11031 PBS.

89 *Computer Associates, Inc., vs. Quest Software, Inc., et. al.* (2004) No. 02 C 7421.

SCO vs. Linux[90] are a series of legal disputes between the company SCO Group (a well known software developer of the UNIX related products) and several Linux vendors and users, including IBM, Red Hat, and Novell. Since 2003, SCO has claimed that these companies infringed upon SCO's intellectual property on the UNIX kernel. Despite the fact that outcomes for some of SCO's cases are pending, they certainly increased the financial importance of all 'copyleft' licenses.[91]

Many of the court proceedings refer to the Software Freedom Law Center (SFLC). Launched in 2005, it provides *pro-bono* legal representation and related-services to non-profit developers of free and open source software. Among the lawsuits filed by the SFLC are the BusyBox-related litigations: starting from 2007, the SFLC filed a series of copyright infringement suits against various defendants on behalf of BusyBox's principle developers, Erik Anderson and Rob Landley. They claimed violation of the GPLv2. Lawsuits were filed, among others, against Monsoon Multimedia Inc., Xterasys Corp., High Gain Antennas LLC, Verizon Communications Inc., and Cisco System Inc. All the cases centered on the failure of the infringer to distribute the source code under the terms of the GPL license. Most of the cases ended in settlements and with the defendants agreement to start distributing in compliance with the license and paying the corresponding fees.

A complex but interesting dispute is the *Jacobsen v. Katzer* case[92]. The dispute involved copyright and patent issues (the patent issue, however, was subsequently removed from the case), as well as Digital Millennium Copyright Act and 'cybersquatting' issues. The importance of this case lies in the CAFC strengthening artistic license agreements by affirming (for the first time in history) that such licenses' violations equates copyright violations. Even though the reasoning was limited to the artistic license and subsequent interpretations of each open source license will depend on its precise provisions, this decision has strong repercussions for FOSS licenses in general.

In Europe, Germany has developed what is perhaps the most comprehensive body of FOSS-related case law. This is partly due to one important project that was launched in 2004 by a German programmer, Harald Welte: the GPL-Violations.org[93] project. The purpose of the project was to track down and prosecute violators of the GPL. Since 2004, GPL-Violations.org claims to have enforced over one hundred actions that were successful in either settling or obtaining judgments. The most relevant cases include: the *Welte vs. Sitecome Germany* from 2004[94], the *Welte vs. D-Link*[95] from 2006, and the *Welte vs. Skype*[96] from 2008. The first two cases confirmed

90 These cases included: *SCO-Caldera vs IBM* (2003) US District Court District Court of Utah, No. 2-03-cv-294; *SCO-Caldera v DaimlerChrysler* (2004) Oakland Country Circuit Court, Michigan, No.: 2004-056587; *SCO-Caldera v AutoZone* (2004) US District Court District Court of Nevada, No.: 2-04-cv-00237-RCJ-GWF; *Red Hat vs. SCO-Caldera* (2004) US Federal Court District of Delaware, No: 1-03-cv-772; *SCO-Caldera vs. Novell* (2004) US District Court District Court of Utah, No.: 2:04cv0139. All the legal documents related to the cases can be found at: http://sco.tuxrocks.com.

91 See note 85 above.

92 *Jacobsen vs. Katze* (2008) Federal Circuit, No.: 2008-1001.

93 See http://www.gpl-violations.org/.

94 *In Re Welte vs. Sitecome Germany*, District Court of Münhen I , No 21 0 6123/04 (2004).

95 *In Re Welte vs. D-Link Germany*, District Court of Frankfurt am Main, No 2-6 0 224/06 (2006).

that failure to provide source code originally licensed under the GPL is a violation of the GPL's terms, and ultimately warrants legal action. In the Sitecome's case the court granted a preliminary injunction against Sitecome for failing to provide the source code under the terms of the GPL. Specifically, the Court rejected the defendants' claims that it had not agreed to the GPL and that the plaintiff had waived all rights to the code by distributing it under the GPL. In the D-Link case the court affirmed the validity of the GPL terms under German law and ordered D-Link to reimburse GPL-violations.org for the enforcement expenses. This was one of the first rulings on damages arising from a GPL violation. Finally, in *Welte vs. Skype*, Skype sold third-party hardware on its website unaccompanied by the source code or a copy of the license. Skype tried to remedy this by providing links to both the source code and the license, but this was insufficient. According to the Judge: "If a publisher wants to publish a book of an author that wants his book only to be published in a green envelope, then that might seem odd to you, but still you will have to do it as long as you want to publish the book and have no other agreement in place"[97]. In other words: full compliance with the GPL is needed.

As shown, all the existing case law has focused on the distributor's failure to make available certain FOSS code, and not on the allegedly "tainted" or "contaminated" company's proprietary code.[98] When companies have been found to distribute in breach of the GPL licenses' terms, they have been asked to either start distributing in compliance with the GPL (and pay some license fees) or to stop distributing but not to distribute their proprietary code under the GPL.

It should be pointed out that it appears to be a widely accepted procedure among FOSS right holders to initially attempt to find a reasonable deal with the alleged infringer. Usually the right holder would ask the company in violation to comply with the license before initiating any legal proceeding. For instance, In *FSF vs. Cisco*[99] FSF initiated legal proceedings after several attempts to elicit compliance with the license terms, and several refusals by Cisco in taking any action.

Coping mechanisms

The legal risks associated with the *reciprocity of the FOSS licenses* in general, and the *copyleft clause* in particular, have been reported as highly challenging and one of the most concrete risks associated with the *hybrid* model. To reduce such risks, companies have developed different coping mechanisms. Many of the interviewed companies shared a common practice: to avoid using OSS packages which includes GPL or copyleft-type of licenses. Some companies included GPLed (or GPLed-type) code, but only for OSS tools they used internally. These companies ensure that pieces of GPLed code are not distributed to the customers. One of the

96 *Welte vs. Skype Technologies SA*, Higher Regional Court of Munich (2008).
97 This English translation was provided in Harald Welte's blog: http://laforge.gnumonks.org/weblog/2008/ (08 May 2008).
98 One exception is the *Progress Software Corp. vs. MySQL AB* (2001), Civil Action No. 01-11031 PBS mentioned before in note 88.
99 For instance in *FSF vs. Cisco* (USA 2008) before initiating legal proceeding, the OSS right holder had asked Cisco for over two years to start distributing in compliance. The case was settled later in 2009.

companies also claimed to receive requests from its own customers to omit FOSS code from their products, due to risks associated with the 'copyleft' clause.

One company avoids incorporating GPLed software into its final product because of the so called "community risk". Specifically, the company believes that the OSS community has tools to detect possible violations of the OSS licenses' terms. If the OSS community considers the company non-compliant with the GPL, the company could attract a lot of negative attention.

Only one company distributes GPLed code with its final proprietary software. According to the company, interpreting something as unclear as the GPL licenses (and FOSS licenses in general) in a purely literal, legal interpretation is not accurate. Instead, it is essential to actively follow interpretations given by different sources, including the existing and evolving jurisprudence, the more recent interpretations of organizations like the Free Software Foundation (FSF), and the FOSS community. The company was well aware of the fact that none of the court decisions have focused on the possibility that GPL terms might "taint" a company's proprietary code, and that the primary aim of any FOSS right holder is to reach an agreement outside the courtroom. The company justifies this with the fact that the primary aim of the FSF and the FOSS community is to promote the philosophy and ideas behind the GPL and open source software. This goal has been achieved by seeking compliance of the source code distribution both in and out of the courtrooms. For example, there have been convincing court decisions confirming the ideas and philosophy of the open source movement, and this promotes the use of open source software in general. The company maintains that this success is attributable to the FSF and the FOSS communities focus on implications of the license, such as the "contamination" of the proprietary code issue. The company is not considering changing this strategy in the future, due to the long lasting acceptance of this interpretation of the 'copyleft' clause. Consequently, it does not possess any back up plan.

4.1.3. Ownership of Rights

Concerns

As mentioned earlier, another concern associated with *hybrid* models relates to the ownership of rights. To commercialise software, companies must have undisputed rights over the product. Open source licenses are copyright licenses, and not interpreted as the licensor relinquishing rights. In order to incorporate an OSS licensed piece of software into a proprietary framework, a company must carefully evaluate the conditions under which such a product had been licensed and acquire all the rights over it.

Ownership of rights issues might concern the proprietary companies that use OSS due to the collaborative way open source projects are developed and the resulting fragmentation of copyright rights. For instance, it might often be difficult to identify the right owners in a FOSS project, as there might be several holders for a single FOSS

component. Furthermore, FOSS components might include files without copyright notices, or notices without proper licenses.

Coping mechanisms

Theoretically, the *ownership of rights* concerns can be solved in two ways: 1) by fully re-writing the FOSS code (including re-write all the contributions to the code, for instance for bug fixes), or 2) by somehow acquiring all the necessary rights over the software.

The first option, adopted (for example) by MySQL, is safe from a legal point of view. Fully re-writing the code, however, might be prohibitively expensive as it may involve complex and costly R&D studies,[100] especially since software is constantly becoming increasingly more complex. As mentioned earlier, digging into big FOSS packages to change parts of the code can be very laborious and time consuming, particularly if a programmer is not very familiar with such software.

Another option to solve the issue of rights ownership is to try to obtain all necessary rights through a specific license or contract. Even though this offers a more affordable option, such an alternative might not clear all the legal risks. Problems can remain if the transfer of rights is incomplete if, for instance, because the code contributor has no authorization to withhold the necessary rights.[101]

Most of the interviewed firms had not thought the issue through. Some of them agreed that the ownership of rights poses a risk, but they did not consider it highly problematic in practice. None of the companies are taking any precautions in this respect. One of the companies affirmed that proper compliance with the FOSS licenses included in the packages suffice in clearing all the risks of ownership.

One company considered the fact that in many FOSS projects the fragmentation of rights is a benefit for the FOSS users (i.e. the companies). The fragmentation of rights limits the possibility for the copyright holders to act because it is too challenging for a single FOSS right holder to get consensuses to pursue certain claims. The company recognizes that this justification does not 'guard' against all the risks because problems with the ownership of rights are highly contingent upon jurisdiction. For example, in certain jurisdictions it is possible to pursue a claim even in the case of "partial" ownership.[102]

4.2. IPRs-Related Risks

Another set of interesting legal challenges relates to the intellectual property related risks. Specifically, the patent infringement and litigation risks, the dilution of the

100 See M Välimäki, "Dual Licensing in Open Source Software Industry", (2003) *Systemes d' Information et Management* Vol. 8, No. 1, pp. 63-75.
101 *Ibid*.
102 See, for instance, the case of Finland: Finnish Copyright Act (2010), Section 6. Available in English at: http://www.finlex.fi/en/laki/kaannokset/1961/en19610404.pdf.

company's own patents and the lack of warranties and indemnifications from the FOSS side.

4.2.1. Patent Infringement and Litigation

Concerns

Intellectual property law does not discriminate between proprietary and open source software for enforcement purposes. Open source is not inherently more likely to infringe upon software patents than proprietary software, and vice versa. Software faces potential IP infringements because it is highly complex, the patent rights in the field are innumerous, many of the patents available are either very broad, or not novel, or not inventive; therefore even a single small software package can infringe upon several hundred distinctive intellectual property holdings.[103] These problems are common to all types of software.

The actual risk of litigation (and the directly related risk of getting injunctive relives), impacts the open source firms in a different way than the purely proprietary and *hybrid* companies.

It has been argued that risks of violations are higher on the open source side: open source developers often operate outside the IP legal framework that dominates the proprietary software industry because most open source projects lack the infrastructure to properly monitor their code base. Furthermore, these organisations often accept code contributions from developers who are unknown to the open source community and therefore have little control over the origin of the code.[104]

Additionally, the fact that the source code is open in FOSS enhances the possibility for competitors to detect infringement. In software, third parties are not always able to perceive patent infringements from the outer function of the product; sometimes it might not be possible to detect violations without knowing exactly how the product works. Generally, the closer the patent is to pure software, the more difficult it is to see from the outside how the software works. In these cases, open codes can prove essential to detecting patent infringements.

On the other hand, the risk of litigation appears higher for both purely proprietary and *hybrid* software companies than for the purely open source companies. IP infringement by an open source software package is more likely to incur legal action from a commercial company, than vice-versa. Even though open source software development has been halted in specific areas of technology with known software patents (e.g. MP3 audio, LZW data compression present in the GIF graphics file formats, etc…) no FOSS organization has yet been subject to legal action for patent infringement. One important reason for this is that pure open source software companies are generally 'not worth suing'. Another is that widely used FOSS projects

103 For more info see notes 5 and 7 above.

104 On this concern, however, it is worth noting that the same problem is found also among proprietary companies. See, for e.g., M Lemley, "Ignoring Patents", 2008 *Mitch. St. L. Rev.* 19, 19-34.

with a large community are more aware about the potential to infringe on existing claims. In other words, the fact that many people use the same FOSS software lowers the chances for companies to be hit by a claim.

The issue of patent infringement is particularly complex for companies implementing *hybrid* models. One key reason that might increase the litigation risks for *hybrid*-type of companies relates to the fact that, as already mentioned, OSS packages often have several users. This means that if company X succeeds in proving that company Y is infringing on patents for a certain piece of OSS code, company X can confidently assume that it can conduct successful legal proceedings against other companies implementing the same OSS software package as company Y. This factor might increase the litigation risks for companies implementing *hybrid* models. The *SCO Group vs. Linux* controversies[105] launched in 2004 and all the cases starting from 2010 in the United States over the Java and the Android operating systems (that will be exposed later in the paper) represent emblematic examples of such risks.

The more competitive a company is and the bigger the market share it owns, the higher the risk of being hit by a patent claim. The reason why certain types of software (e.g. software in phones) are highly litigated is most certainly because they are associated with highly successful devices. Furthermore, if the devices (in particular OSS based devices) are delivered directly to consumers the litigation risks increase because the products become more visible. In other words, successful consumer products have an increased risk of litigation for companies using the *hybrid* protection model.

Finally, it is worth mentioning that in the case of *hybrid* models (in the same way as for proprietary software in general), litigation risks depend on the jurisdiction. For example, it is well know that litigation risks are particularly high in the United States, due to its long history of allowing the patenting of software. In Europe, the case law has limits under EPC Art. 52's exclusions of computer programs "as such" from patent law.[106]

Patent Litigation on FOSS

On the specific issue of FOSS and patents, the existing case law is relatively limited and mostly comes from the USA.

Barracuda vs. Trend Micro (2007)[107] relates to the infringement suit filed by the antivirus software vendor Trend Micro against Barracuda Networks for Barracuda's use of the open source ClamAV product in its network gateway protection devices. The claim is that ClamAV violates one of Barracuda's patents, filed in 1995[108]. Trend Micro accused Barracuda of infringing its patent directly, contributory, and by inducement. Barracuda went to a Californian federal court first and filed a lawsuit

105 *Caldera Sys., Inc. vs. Int'l Bus. Machs. Corp.* (2003) US District Court District Court of Utah , No. 03-CV-0294.

106 For more information see RM Ballardini, note 7 above.

107 *Barracuda, Inc., vs. Trend Micro, Inc.* (2007) USITC, No. 337-TA-624, 72 Fed. Reg. 74, 329.

108 US patent 5,623,600 - "Virus detection and removal apparatus for computer networks".

against Trend Micro[109], seeking to settle the controversy through a declaratory judgment declaring Trend's patents invalid. With the help of the FOSS community, Barracuda is now trying to gather prior art in order to invalidate Trend's patent.

In 2006 FireStar sued Red Hat for patent infringement in *FireStar vs. Red Hat*[110]. The dispute was settled in June 2008[111] with the agreement that Red Hat claims afford broad upstream and downstream protection for the whole FOSS community. *Microsoft vs. Tom Tom*[112], saw the software giant accusing Tom Tom's navigation products of infringing upon Microsoft's patents for the FAT32 file system, was settled in March 2009.

The series of legal disputes related to the Java and Android platforms require discussion. These lawsuits began in 2010 with *Apple vs. HTC*[113] (Apple claimed that HTC infringed upon twenty patents in the iPhone's user interface), with Apple claiming different patent infringements by the Android open source operating system. Other Android related lawsuits are *Microsoft vs. Motorola*[114], *Apple vs. Samsung*[115], *Microsoft vs. Barnes and Noble*[116], and *Oracle vs. Google*[117]. Specifically, in this last case filed in August 2010 at the District Court for the Northern District of California, Oracle claimed willful infringements of certain patents related to the Java programming language distributed on Google's developed Android software, and of some unspecified copyright rights. At the time of writing the cases are still pending.

Coping mechanisms

Theory

The academic literature suggests that to mitigate the risk of patent litigation, several coping mechanisms are available to companies. For instance: patent acquisition, re-engineering sections that allegedly infringe patents (when possible), collecting and keeping prior art information to invalidate patents either in-house or through big

109 *Barracuda, Inc., vs. Trend Micro, Inc.* (2007), US District Court Northern District of California (San Jose) No.: 3:07-cv-01806-MHP.
110 *FireStar Software, Inc., vs. Red Hat, Inc., et al.* (2006) US District Court Texas Eastern District Court, No. 2-06cv-258.
111 See Settlement Agreement, 6 June 2008, available at: http://www.redhat.com/f/pdf/blog/patent_settlement_agreement.pdf.
112 *Microsoft Corp. vs. Tom Tom NV and Tom Tom, Inc.*, (2009), US District Court Western District of Washington at Seattle.
113 *Apple Inc., vs. High Tech Computer Corp., a/k/a HTC Corp., HTC (B.V.I.) Corp., HTC America, Inc., Exedea, Inc.* (2010) US District Court District of Delaware.
114 *Microsoft Corp. vs. Motorola, Inc.* (2010), US District Court Western District of Washington at Seattle.
115 *Apple Inc., vs. SAMSUNG ELECTONICS CO. LTD, et al.* (2011) US District Court California Northern District (Oakland) No.: cv-11846.
116 *Microsoft Corp. vs. Barnes and Noble, Inc., et al.* (2011) US District Court Western District of Washington at Seattle.
117 *Oracle America, Inc., vs. Google Inc.* (2010), US District Court California Northern District (Oakland), No.: 3:10-cv-3561.

projects (like the Open Source as prior art project[118]), actively participating in projects that facilitate better examination and quality of patents (e.g. the "peer-to-patent" project[119], the "patent quality index" project[120]), or establishing patent pools with reasonable terms.

On a broader scale, there are several recent initiatives to turn patents into "open source intellectual property rights", granting use to all members of the community.[121] Commitments to encourage the development of OSS projects, such as patent holders unilaterally pledging not to enforce some of their patents against users of certain open source software, are one example of such initiatives. A typical example of this model was the agreement reached in 2006 between Microsoft and Novell, where Microsoft announced not to enforce its patents against the version of Linux distributed by Novell.[122] Another initiative relates to patent owners committing not to sue those adhering to a statement of permitted use. A plan put forward by IBM in 2005 was formulated along these lines, where the firm announced that it would release 500 of its patents into a "patent commons" available for the open source community.[123] Other initiatives aim at coordinating and encouraging unilateral commitments. For instance, following the aforementioned decision by IBM, several companies have made similar patent pledges. These companies include the Open Source Development Lab (OSDL), a non-profit institution financed by large commercial companies, and dedicated to the promotion of Linux among firms that now host their patents in a "patent commons".[124] In other words, the OSDL provides a central location for patent pledges and software patents. Nokia, for one, has committed not to assert all its patents against the Linux kernel.[125] In 2007 the OSDL merged with the Free Standards Group (anon-profit consortium chartered to specify and drive the adoption of open source standards) to form The Linux Foundation. Both organisations narrowed their focus to promoting Linux in competition with Microsoft Windows.[126] Theoretically, patent commons do not only benefit the OSS developers by providing them with a shelter, but they also reduce the litigation costs: companies participating in the commons cannot benefit from the protection offered by the commons unless they agree not to sue other firms or beneficiaries for infringement.

118 See http://www.linuxfoundation.org/programs/legal/osapa.
119 See http://peertopatent.org/.
120 See http://www.law.upenn.edu/blogs/polk/pqi/faq.html.
121 See, for instance, note 118, 119 and 120 above.
122 See Microsoft News Center, "Microsoft and Novell Announce Broad Collaboration on Windows and Linux Interoperability and Support" (Nov. 2, 2006), at: http://www.microsoft.com/presspass/press/2006/nov06/11-02MSNovellPR.mspx.
123 See "IBM Statement of Non-Assertion of Named Patents against OSS", at: http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf.
124 See www.patentcommons.org.
125 See Nokia Corp. press releases, "Nokia Announces Patent Support to the Linux Kernel", (May 25, 2005), at: http://press.nokia.com/2005/05/25/nokia-announces-patent-support-to-the-linux-kernel/.
126 See http://www.linuxfoundation.org/.

It is worth mentioning that in recent years web initiatives such as Groklaw[127] and the Foundation for a Free Information Infrastructure (FFII)[128], have been launched. Their goal is to increase communal awareness on the issues concerning IPRs and OSS.

Practice

None of the interviewed companies considered that the use of open source software in their final products enhance (or reduce) the risk of patent litigation.

According to some of the companies, one essential characteristic of the OSS packages they use is to have a large OSS community as support. Highly used OSS packages assure companies that the community of users would promptly detect possible patent claims. Furthermore, these companies were confident that in the case of a potential patent infringement, a large OSS community would find ways of dealing with the issue, such as by re-writing the code in question. Most companies thought that the risk of being hit by IP claims (in relation to the OSS code they incorporate) is higher if using OSS packages that are, in one respect or another, unpopular.

One company considered the fact that the open FOSS code does not necessarily lead to higher litigation risks. According to this company, the fact that in proprietary software the actual code is 'closed' (i.e. not publicly disclosed) can trigger more litigation. The infringements based on analysis rather than facts (based on the source code) leaves more room for interpretation.

Most of the companies were not concerned about facing patent claims on the FOSS code they use. Some companies relied on the fact that their biggest and most successful competitors will make better targets in any patent claims regarding their use of FOSS code. These companies affirmed that in such a situation, they would immediately change the problematic FOSS component. This operation can be highly expensive and time consuming or pretty simple, depending on the importance of the component. Only one of the interviewed companies, however, admitted to regularly monitoring the FOSS packages they are using for potential IP claims. Another company only follows the biggest infringement cases related to FOSS and patents.

One company considered that the incorporation of FOSS software into a company's model can play a positive role in an alledged infringement case. In legal disputes, the open source community can gather prior art to defeat claims on the defendant's behalf. The active participation of the community depends on the specific case and on the position and reputation of the alleged infringer.

Most of the interviewed companies shared the opinion that the patent litigation risks for FOSS within the framework of the *hybrid* protection model are relatively low: taking into account that almost every company in the world that produces software uses certain portions of FOSS, the amount of IP litigation is very minimal. They generally agreed that the litigation risk is higher when a company delivers its

127 See http://www.groklaw.net/.
128 See http://ffii.org/.

products directly to the consumers and is very successful. Some of the companies, however, were not convinced that the litigation risk is different for companies that implement *hybrid* models than for those that use only proprietary software or FOSS as a strictly internal tool.

One company considered the area of industry where the company operates as a key feature when it comes to patent litigation. The company specifies the telecommunications sector as the riskiest area of technology for patent litigation.

Another company identified a company's jurisdiction as an important factor for patent litigation risks. The United States was considered much more dangerous than Europe in this regard. For instance, this specific company does not currently own any patent on the software it produces, but would not consider operating in the USA without filing some patent application..

Finally, one company opined that patent litigation is an incentive in commercial disputes; disputes that have their own logic and do not relate to open source software. The size of companies' patent portfolios and the commercial situation among competitors usually plays a part in triggering patent litigation. FOSS does not play any active or important role in this scenario.

Overall, none of the interviewed companies considered the risk of patent litigation in particular relation to the OSS part of the code they use, as very concrete and, accordingly, do not implement any specific coping mechanism on this respect.

4.2.2. Dilution of the Company's Own Patents

Concerns

Another problem with *hybrid* models is the dilution of the company's own patent rights. Dilution may result from an explicit or implied patent license under the applicable FOSS license. Determining the patent portfolio's exposure to the FOSS licensing model may be difficult. For instance, both the GPLv3 and the BSD license (i.e. the most used FOSS licenses) fail to mention patents. Notwithstanding the absence of an explicit patent grant, however, FOSS licenses may include implied patent grants. These provisions state the right holder (or the distributor), under his/her authorization, may implicitly grant a license to the recipients of the components to practice any right holder's patent claims covered by such component.[129]

Implied patent licenses can be deduced from any conduct of the right holder that induces reasonable belief in the existence of such a license. These can include any written statement of the patent holder (e.g. the wording of the open source license) and/or the way the patent holder acts. For instance, the fact that the BSD license grants the right to "use, modify, copy, create derivative works and distribute the software" might induce reliance based on the statements of the right holder.

129 See A Pugh, and L Majerus, "Potential Defences of Implied Patent License Under the GPL", *Fenwick and West LLP* (2006).

Furthermore, releasing the code under any FOSS license (which imply that the software is "free" for everybody to be used, copied, distributed and redistributed), might, in effect, induce reasonable reliance on an implied patent grant based on the acts of the patent holder.[130]

Indeed, the interpretation of an implied patent license is regional, and not all jurisdictions recognize such a concept as part of their legal regimes. Thus, the place where the software is released and/or where the patent is granted should also be taken into account when evaluating the risk of dilution.

Coping mechanisms

Only one of the interviewed companies released products under FOSS licenses. The company did not consider the risk of dilution as highly problematic. The company was confident that an accurate and careful internal review on both the explicit and the implicit patent grants included in the FOSS licenses used, as well as consideration of the national interpretation of the laws of the country of operation of the company, successfully minimize risks associated with dilution of patent rights.

4.2.3. Lack of Warranties and Indemnifications

Concerns

The fact that most FOSS providers do not offer the same warranty protections typically given to commercial products might represent an additional source of concern when companies consider implementing the *hybrid* protection model. Some OSS organizations have proposed such warranties. Hewlett-Packard, has announced that it will offer legal protection (albeit with rather strict conditions) for its version of Linux. [131] Similar programs have been implemented by other OSS companies, such as Red Hat[132], Novell[133], Hewlett-Packard[134], and JBoss[135].

Coping mechanisms

Most of the interviewed companies feel this risk is not something specific to open source software. Generally, companies did not feel safer by using proprietary

130 A Haapanen, "The is No Such Thing as Free Lunch – Implied Patent Grant Under Open Source Software Copyright Licenses", *Law in The Internet Society* (2008), at: http://moglen.law.columbia.edu/twiki/bin/view/LawNetSoc/AnnaHaapanenPaper1.
131 See Hewlett-Packard Company Website – Terms of Use and Legal Restrictions, at: http://www8.hp.com/us/en/privacy/terms-of-use.html.
132 See Intellectual Property warranty, Red Hat, Inc., at: http://www.redhat.com/legal/open_source_assurance_agreement.html.
133 See Linux indemnification program Novell, Inc., at: http://www.novell.com/licensing/ntap/.
134 See Linux indemnification for HP customers at: http://h71028.www7.hp.com/enterprise/cache/328211-0-0-224-121.html.
135 See JBoss indemnification program at: http://www.jboss.com/pdf/press/indemnification0405.pdf.

software over FOSS alternatives because they felt the availability of specific pieces of software does not depend on whether the software is proprietary or open source. More specifically, companies considered that in the event some third parties' component or tool is hit by a claim, their situation would be the same with an open source or proprietary component. Some companies indicated that even if the use proprietary products, companies would be unable to replace specific products if they were to suddenly become unavailable. Companies in the software field tend to move on very quickly. In the event this situation occurs, both proprietary and open source software users will face the same challenges as the other.

One company considered the lack of warranties on FOSS products to be a cause for concern, but it is certainly not the only reason for disregarding open source software and opting for proprietary versions. The company maintained that there are risks associated both with FOSS and with proprietary software. However, depending on circumstances, there are several reasons for choosing one alternative or the other: the lack of warranties and indemnifications on the FOSS side is but one of those. Reasons for companies to opt for proprietary software could be to get some warranties, but also for maintenance and support. A company's architecture can also play an important role in their choice. For instance, if the company's system is so inflexible that it cannot change to incorporate FOSS software, they are more likely to opt for proprietary software. In summary, the interviewed company opined that there is not a straight answer to the question of whether the lack of warranties on FOSS is a risk or not..

5. Concluding Remarks and Future Research

The substantial investment made by proprietary software firms in open source indicates that the nature of competition in the software industry has radically changed during the past two decades. Not only it is evident that the largely volunteer software movement has altered the basic nature of the software industry, but also it appears clear that the FOSS phenomenon has undergone a significant transformation from its free software origins to a more mainstream, commercially viable form.

Open source developments have experienced so much success that proprietary companies now incorporate open source strategies into their protection models, and very successful open source projects have had business models created around them. Despite the success of open source software models, it is highly unlikely that the proprietary software will wither away and die. The recent incorporation of proprietary features within some of the most prominent open source companies is clear evidence that IP rights have not decreased in importance during the past twenty years. On the contrary, it appears more likely that the proprietary and open source models will continue to co-exist as they have for a long time.

However, what this analysis shows is a shift in the way companies employ these models in the software industry. Although it remains clear that open source and proprietary software models will remain distinct, the data provides evidence demonstrating that in the future software will not receive licenses at either extreme.

The reason for this trend is simple: none of the currently available protection mechanisms individually succeed as a stand alone source of protection for computer programs. Instead, a balance of several protection mechanisms, including strictly IP models and FOSS, can meet the specific, customized needs of individual companies.

The *hybrid* model carries a large amount of potential legal risks. By combining legal theory and empirical research, this paper has provided a comprehensive analysis of the 'core' legal challenges surrounding the implementation of the *hybrid* model in the software context, and the coping mechanisms companies can implement in order to navigate such risks. Specifically, the empirical study confirmed the existence of all the theoretically formulated legal risks in practice, even though some were considered more concrete than others. The empirical research showed that the most challenging risks are those associated with the open source licenses, the compliance and compatibility with the licensing terms, and the risks associated with the 'copyleft' clause being the most concrete. These risks applied to both large and small companies. The least risky problems for the interviewed companies were associated with the lack of warranties and indemnifications on the open source side. Generally, the companies considered this feature equally problematic for closed and open source software. The problems associated with the ownership of rights were issues where companies did not have strong opinions and had not given it much consideration. The infringement risk was reported quite heterogeneously: even though all the companies agreed that incorporating open source into their final products does not enhance or reduce the risks of patent litigation, they provided different opinions on the reasons and consequences of the infringement and litigation risks (e.g. how widely the OSS package is used, field of technology, jurisdiction, etc...). Finally, only one company released software under FOSS licenses and, was the only participant that could comment on the dilution of the company's own intellectual property rights. The company was confident that the risk is minimal provided there is a thorough review of both the code and the FOSS licensing terms internally before releasing the final product.

The study showed that none of these methods is fully foolproof *per se*. Due to uncertainties surrounding the FOSS environment and the interpretation of the FOSS licenses, a well structured internal procedure and a solid design of the architecture and principles surrounding the overall model the company plans to incorporate is essential to reducing the legal challenges. Accordingly, all the interviewed companies conduct legal reviews and checks, either through an internal structured procedure or by delegating such responsibilities to one (or several) in-house staff members.

Generally, companies did not welcome the idea of a legislative entity to solve these problems, mainly due to the too slow legislative process in relation to the fast trends of the software industry. On the other hand, however, most companies felt the need for more case law and court interpretations.

The specific field of research at the centre of this essay has not been extensively investigated in previous literature. As mentioned earlier, even though several studies on the issue of intellectual property and open source software exist, they mostly

focus on FOSS as a phenomenon or proprietary software individually, and neglect to investigate the relationship and interaction between the two, the ways open source software affects companies' IP policies, and vice versa. Consequently, this case study relies on a number of theoretical assumptions formulated in the first part of the paper. Reliance on hypothesis was further enhanced by the still 'grey' area surrounding the interpretation of the FOSS licenses terms. One of the main difficulties in researching areas with ill-defined legal principles is the lack of clear-cut interpretations of the norms at hand. It is not realistic to claim that the findings of this study are applicable to all the companies that implement *hybrid* types of protection models. This was not the original purpose of the study nor does it represent its main contribution. Instead, the aim of the study (both the theoretical and the empirical part) was to generate new knowledge for strengthening the general understanding of the problems surrounding the use of *hybrid* models in the software field. The case study was relevant because it provided in-depth answers to the theoretical issues formulated in the first part of the paper. The empirical analyses shed light both on the most concrete legal risks associated with *hybrid* models, and on the coping mechanisms that are used to navigate such challenges.

The exercise conducted in the article was an explorative undertaking about the risks associated with *hybrid* protection mechanisms in the software industry. Specifically, this article limited itself to the legal problems surrounding the model. There is still much about software *hybrid* protection models that remains a mystery and warrants further research. Perhaps the more essential avenue for work in this area lies in the replication of similar studies among different participants to test the replicability of the results achieved. Another interesting avenue for research would be to explore different aspects of the open-closed business model in the software environment. For instance, the economic or managerial aspects of the problem could be investigated. The research conducted in this paper constitutes a solid basis for any further study that might aim at investigating or addressing problems related to the *hybridization* of other fields of technology. The biotechnology sector, for example, is another field where the closed-open innovation model is prominent but remains unexplored by researchers.