An Investigation of How Commercial Software Companies Should React to Open Source Software

UW CSEP 590 TU Course Project

December 10, 2004

Patrick Haluptzok, Bipin Karunakaran, Rodrick Megraw, Magdalene Tatum, James Welle, and Song Xue

Abstract

Open source software began as the hobby of a small number of programmers and has developed today into a worldwide phenomenon that is a viable economic alternative to proprietary software. As a proprietary software company, it is essential that you understand open source and how to interact with this type of software. You may choose to compete with open source or embrace it into your business model, but ignoring open source is not a viable option. We present this paper as a guide to open source software. First, we discuss what open source is and explain its history. We also explain the general advantages and disadvantages of open source software. Second, we explain the myriad of open source licenses and how they work. Next, we discuss how your workforce can safely and effectively coexist and interact with open source software. In the fourth section of the paper, we discuss the different open source business models and the benefits of each. Fifth, we discuss the specific legal issues involved with open source software and we highlight the recent SCO litigation. Finally, we present a case study on Microsoft Windows and other proprietary platforms and investigate how they have reacted to the open source phenomenon. We also investigate possible open source strategies for the Windows operating system.

0 Table of Contents

Introduction

Open Source Software (OSS) was born in universities and research labs and today is spreading to the commercial world. Many companies are adopting OSS strategies and licensing models in whole or in part. Some of the most successful OSS initiatives to date include Linux, Eclipse, Apache and Mozilla. For companies looking to define their software strategies for the coming years, ignoring OSS would be a mistake. In this paper we try to trace the origins of the OSS movement, highlight pros and cons, and discuss alternatives to pure OSS. We talk about different licensing models which exist to serve different segments of the industry and investigate how commercial software company employees can co-exist in the mixed world of proprietary and open source software. Legal issues have had a huge impact on the growth of OSS, so we take a closer look at the legal risks by studying the SCO case. Finally, we end with a case study on open source strategies for Microsoft Windows and other platforms.

1 Open Source Definition, Benefits, History, and Alternatives

1.1 Source Code Defined

Before we begin discussing open source in depth, it is important that we have a good understanding of the differences between the source and binary versions of a computer program. Software consists of the source code and its corresponding binary executable. Source code is a set of instructions that tell a computer what to do. The source code is an easily readable format that specifies how the computer is to respond to user input, how to process data to achieve the intent of the user, and how to present and store the results of computation. Source code is what software engineers write and modify, the naming conventions and comments in the source code explain what the program does. Source code by itself usually can't be run on a computer directly. Source code is converted by means of a compiler into a binary executable which can be run on a computer. The binary executable is a more efficient form of the instructions for the computer to execute, and binary executables are very difficult to impossible for humans to read. The exact original source code can't be derived from the binary executable, and attempting to convert the binary executable into source code is illegal under most software license agreements.

The binary executable is what a customer traditionally buys when purchasing proprietary source software. The source code is kept private and usually only released under extremely restrictive non-compete and non-redistributable licensing agreements. Further details on the business motivations and practices of proprietary software companies are discussed in Chapter 3.

1.2 Open Source Defined

The open source movement has been gaining momentum for the past twenty years, mostly in universities and technical areas such as the internet and the worldwide

web. Now it has breaking into the commercial arena as a powerful platform in combination with proprietary software, often even as alternative to proprietary software. As the name suggests the source code is available for viewing and modifying unlike propriety software where only binary files are available to the user. Also the program and the code are freely available on the internet for download. More formally open source as defined up on opensource.org as software to which there is access to the source, in addition the software complies with the following criteria:

1.2.1 Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

1.2.2 Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

1.2.3 Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

1.2.4 Integrity of the Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

1.2.5 No Discrimination against Persons or Groups

The license must not discriminate against any person or group of persons.

1.2.6 No Discrimination against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

1.2.7 Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

1.2.8 License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

1.2.9 License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

1.2.10 License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

The definition is broad and leaves a lot of room for interpretation and provisions exist where the chain of being a good open source citizen can be broken; we will talk about this in more detail later in the paper.

1.3 Why Open Source

The basic idea behind the open source movement -- more people look at the code on the internet, more people read it, more people adapt it, more people redistribute it, more people fix bugs and slowly you have this huge community of people which cannot be employed by a single corporation, all working for the same project and churning out new software at speeds which never could be achieved by conventional software development team.

Cost is another factor. OSS is available for download freely. As compared to proprietary software, the initial cost of OSS is zero which is very attractive for startups and small businesses. Other advantages include reliability and a large number of software engineers working round the clock at no cost to the user. The contributors to the project could be distributed at different corners of the world making support and fixes available round the clock.

The other reason for using OSS could be purely based on principles that open source is an ethical business model, all the source code is in the open and none of it is concealed and is good for the client or the user of the software to modify to fit them to their needs rather than depend and wait for the seller of the product to make changes.

1.4 Open Source Advantages

Reliability is one of the most important reasons for adopting open source. The argument towards increased reliability is basically about more number of users see the code and more they look at the code the more they find problems and fix them. In addition there is a larger base of people who report problems than just a bunch of select testers who work at a conventional software company. A paper has been written by D.

Bosio, M. J. Newby titled “Advantages of open source project for reliability: clarifying the issues”. The authors discuss the improvement of reliability by considering open source software as a development model like conventional development model and use mathematical models to predict why reliability of open source is greater or less than conventional software development model. Major findings of the paper are

- Reliability Growth during use: Reliability increases as more people use the source code to enhance or to look at it just from the standpoint of understanding it.
- Diversity is useful for improved reliability: Diversity is a good thing: all things being equal, it is better for users to have diverse demand profiles than for them to have the same profile. The community dealing with open source software is more diverse than the one which deals with conventional software.
- Cost reduction is achieved by not having to pay for the propriety software. Developers work on development of the product for free. The number of people working on the project could be significantly great considering there is no single company employing them. This leads to faster development and if a significant number of people get interested in the modifications that a company is making to the core software, the company making the modification might be able to leverage coding efforts from the open source community, reducing the cost of development. The cost factor has been heavily debated in the industry considering support costs paid to third party companies like Red hat and IBM.
- Inventory/tracking free, open source software need not be tracked for licenses and avoids extra money, time and effort spent on creating an inventory system.
- Rapid feature addition and bug fixing, if there is a problem in the open source software you don’t have to wait for the vendor to fix the problem for you. You have the source code and you probably can fix it yourself, assuming you have the technical expertise on board.

1.5 Open Source Disadvantages

The advantages of Open source could very well be the disadvantages if considered from a different perspective

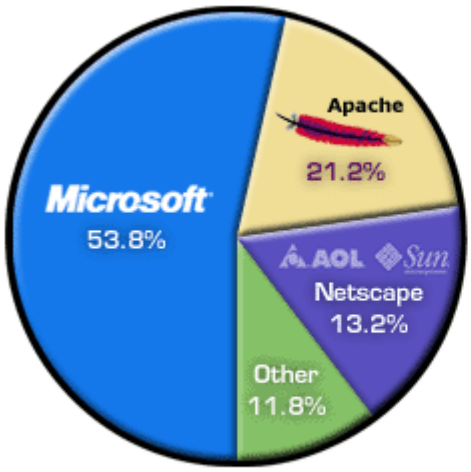
- Reliability: Is open source really reliable? The argument for open source reliability is based on a simple argument Eric Raymond's "many eyeballs" maxim: with many eyeballs, all bugs are shallow (The Cathedral and the Bazaar 1997). If more people look at the code more issues will be unearthed and more issues will be fixed. Is it necessarily true? If the eyeballs looking at the code were real experts in the software field and had several years of experience in software development then that would be the case. It could be a possibility but not a guarantee that the eyeballs looking at the code are mature experts. Reliability is no guarantee but a possibility like in any other software.
- Cost reduction of open source software is a highly debated topic in the industry. The initial cost of the software is zero, support and maintenance costs could be higher than proprietary software. If the client is huge enough for a proprietary

software company, often maintenance and service is free. If a bug halts production for a few days and the in-house expert is unable to solve the problem there is no paid support which could bail out the company from crisis and could cause millions of dollars in lost production and credibility.

- Licensing and legal issues (intellectual property and patent infringement) is the biggest problem with OSS. The client company using the Open source software is not guaranteed of not being sued due to some intellectual property right being violated. The source and the authenticity of the developers contributing to the Open source software is unknown and this could lead to patented software making its way into the open source software that is being developed. OSS developers and distributors make no intellectual property warranties.

1.6 Open Source Today

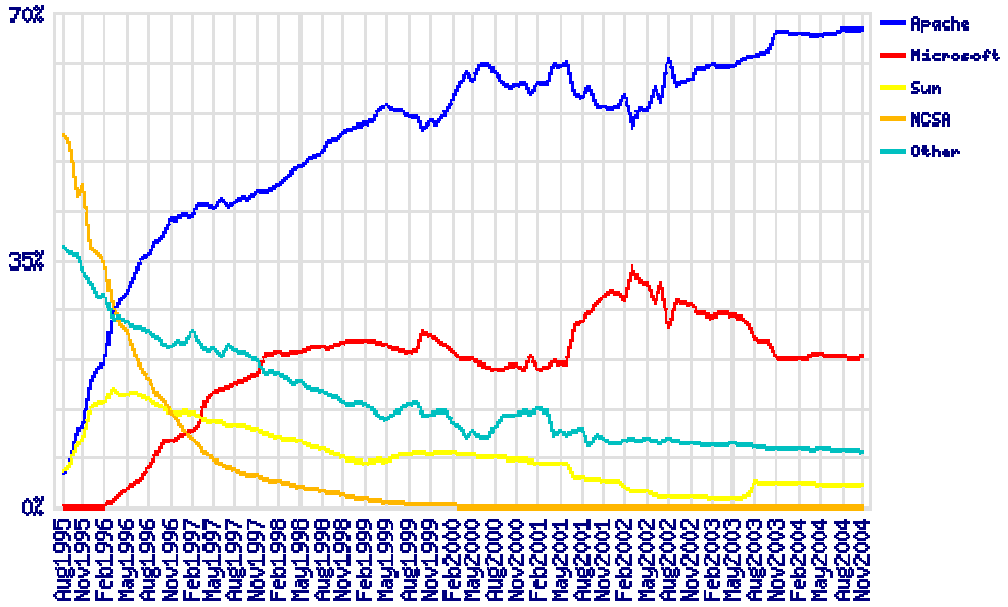
From Dell saying that Linux is too complex for desktop at one point to selling laptops and desktops preinstalled with OSS Linux. OSS has come a long way, from being shunned to being a threat to proprietary software like Microsoft. On the other hand a study shows that a majority of the fortune 1000 companies use IIS server over apache to host their sites. Microsoft's IIS server was not even in the market when Apache was gaining ground. Obviously OSS is a great idea with lots of potential but with its disadvantages, companies are really thinking hard before going the OSS route or playing it safe by sticking to proprietary software.



Microsoft IIS	**53.8%**
IIS 5.0	40.9%
IIS 6.0	9.0%
IIS 4.0	3.9%
Netscape	**13.2%**
Enterprise 6.0	5.6%
Enterprise 4.1	5.5%
Enterprise 3.6	1.4%
Enterprise 4.0	0.4%
Enterprise 3.5.1	0.2%
Other	0.1%
Other Servers	**11.8%**
Sun One	2.0%
Lotus Domino	1.5%
IBM	0.8%
Weblogic	0.7%
Websphere	0.7%
Zeus	0.1%
Other	2.9%
Unknown	3.1%

Apache	**21.2%**
Apache 1.3.27	3.8%
Apache 1.3.26	1.2%
Apache 1.3.29	1.1%
Apache 1.3.31	1.1%
Apache 2.0.50	1.0%
Apache 1.3.28	0.7%
Apache 1.3.20	0.6%
Apache 2.0.46	0.6%
Apache 1.3.19	0.5%
Apache 1.3.12	0.4%
Apache 2.0.48	0.4%
Apache 2.0.49	0.4%
Apache 1.3.9	0.3%
Apache 2.0.40	0.3%
Apache 2.0.44	0.2%
Apache Coyote	0.2%
Apache Tomcat	0.2%
Apache Other	4.5%
Apache Variants	3.7%

Microsoft IIS is the Top 1000 Corporations' Web Server Leader with 53.8% of market share



Market Share for Top Servers across All Domains August 1995 - November 2004 (www.Netcraft.com)

As the two graphs show overall Apache servers lead in market share, but in the top fortune 1000 companies IIS is clearly a leader.

1.7 Open Source History

Source sharing has been around from almost the same time that software has existed. Computer programmers in universities and research facilities across the world shared software. Software actually became copyrighted only in the 1980's; with the rise of software copyright the notion of sharing software started falling.

Richard Stallman is the father of free software. His story and contribution to the OSS movement begins when he was a professional programmer at the MIT Artificial intelligence lab. They had received a printer from Xerox Corporation as a gift and Stallman sent a job to be printed to the printer, when he reached the printer he realized that his 50 page document hadn't printed at all, other user's print job had jammed the printer. The AI lab staff soon realized that it was a problem with how Xerox had built the printer basing their ideas on the Copier machines that they were so successful with. Stallman had fixed a similar problem with the earlier printer that the AI lab had, by opening the software module that modulated the printer on the PDP 11 machine. He couldn't fix the jam problem but he had inserted a software program that would report back to the PDP 10 machine the labs central machine that no one users print job would bring down the entire print job queue. This was one of the characteristics of the smart programmers at the AI lab that prompted companies like Xerox to donate machines, so that they could bring back such clever fixes into their products and incorporate them into the newer versions. This time Stallman wanted to make the same fix to the Xerox printer

and he was surprised to learn that Xerox had only included binary files for its printers. Stallman slowly started hunting for the source code for the printer and he found Sproull who had worked at the Xerox center and who had recently joined Carnegie Mellon's AI lab. It was not long before he realized that he was under a NDA to not disclose the source code for the Xerox printer. That set Stallman thinking about free software as in "freedom". Stallman says "Without the printer incident his life might have followed a more ordinary path".

Stallman first started thinking about licenses for free software when he was working on "Text Editor and Corrector"(TECO). He wanted to make sure that all modifications made on the software reached others; he put in a "Terms of Use" statement in the source code. "Users were free to modify and redistribute the code on the condition that they gave back all the extensions they made" he wrote. Stallman dubbed it the "Emacs Commune." In 1985, Stallman created the Free Software Foundation, a tax exempt charity, to support his work and that of his collaborators.

In 1989 Stallman wrote the first version (1.0) General Public License(GPL). He changed the license from the original Emacs license, in the original Emacs license he said all derivative works must be published, in the GPL he only asked works that would be distributed in the same way that Stallman distributed, should be published. The GPL had its problems and evolved over the years as any the software itself (Stallman).

Eric Raymond was also one of the contributors to the GNU project, but had distanced himself from the project due to what he termed as Stallman's "micro management". Raymond also was known for his short temper which played a part in Raymond and Stallman not working together. In 1997, Eric Raymond published an essay entitled "The Cathedral and the Bazaar". In the essay, Raymond articulated the reasons why he believed that open source licenses--licenses that allowed anyone to freely view, modify, and distribute the code--resulted in higher quality, less expensive software. The essay spread quickly through the programming community In 1991 Linus Torvalds posted an article on comp.os.minix, his experimental kernel was running bash and GCC and he would post the source code soon. In late 1995 Peter Salus a member of free software foundation invited these three people Raymond, Stallman and Torvalds to a conference on free software, this was the beginning of a strong force in the direction of Open source Software. Even though there is a slight difference in the Free Software model that Stallman is credited to pioneering and the Open Source Software movement that Raymond is credited to having started, both share the same underlying principles.

Netscape was one of the companies that attended this conference. Netscape's CEO stated that Raymond's essay "Cathedral and the bazaar" was one of the driving forces for the company's decision to go open source. Shortly afterward, a coalition of individuals, led by Eric Raymond, Bruce Perens, and Tim O'Reilly, decided that the free software community needed better marketing. They formed the Open Source Initiative to a) promote the pragmatic benefits to the business community, and b) certify free/open source licenses that meet the Open Source Definition.

The Open Source Initiative's evangelism paid off. Following Netscape's announcement, several additional vendors announced support for Linux, including Oracle, IBM, and Corel. Intel and Netscape invested in Red Hat, the largest English language Linux distributor.

A statistically insignificant presence in 1997, the popularity of Linux and the free/open source software movement exploded. The International Data Corporation (IDC) estimated that Linux has 25% of the server market, second only to Windows NT which has 38%. With 4% of the market, Linux is also the third most popular desktop after Apple. Moreover, IDC estimated that commercial shipments of Linux will grow at a compounded annual growth rate of 25% from 1999 to 2003, compared to 10-12% growth rates for other operating systems. (Note, however, that Linux's installed base was quite small--it's much easier to have high growth percentage rates when your starting absolute numbers are small.)

In August of 1999, Red Hat Linux went public. The stock price soared to $72 dollars the day after the IPO, giving Red Hat a market capitalization of $4.8 billion--a remarkable valuation for a company with a $5,787,945 net loss on $33,031,682 million in revenues for the fiscal year ending in February 1999. VA Linux, a vendor of hardware with Linux pre-installed, netted the largest first day run-up in IPO history, giving VA Linux a $7 billion dollar market capitalization. Other successful Linux IPO's include Cobalt Networks ($3.1 billion) and Andover.net ($712 million) (Scannell, 1999).

1.8 Other More Recent Successes

Recently OSS has become more of a mainstream product with many companies adopting it in their offices. OSS market share has also grown rapidly making many companies sit up and take notice of the OSS phenomenon.

- IBM recently announced that the company would devote almost $1 billion dollars to support Linux. (Burke, 2000)
- Forrester Research estimates that more than 55% of the world's 2,500 biggest firms use open source software, with almost a quarter using the software in production systems. (Connor, 2000)
- Sun released Star Office, an office suite similar to Microsoft Office, under the GPL license.

To be sure, free/open source software still faces challenges. Both Red Hat and VA Linux, two of the most prominent corporate supporters of Linux, still lose money.

1.9 Open Source Alternatives

Code sharing is one of the alternatives suggested. In this method a known entity takes responsibility of sharing the code under certain confidentiality agreement with another entity. Like Microsoft decides that it wants to share its code module related to Processor bus access with Intel. This would help both Intel and Microsoft as Intel could understand how Microsoft is using its bus technology and could provide suggestions to improve the code to better utilize its architecture on the other hand Microsoft engineers would have an opportunity to look at Intel’s low level processor bus utilization code and make suggestions or bring in experiences they have had with customers to improve on the architecture. The only thing that each company cannot do is freely distribute whatever code they have of each other to other entities. Microsoft does code sharing with universities and other companies.

This approach has the advantage over OSS, it brings with it guarantees that no intellectual property rights are violated. The source of the development is known to be a certain company or entity and can easily be traced back to an individual owner. In the OSS world there is a set of unknown people who contribute to the code and some of them might be lifting patented code and contributing to the OSS development effort. It also means that support is readily available in case something goes wrong.

Microsoft has various code sharing licensing methods under “Shared Source Initiative” program which details around 22 different licenses tailored around different industries and products. The Enterprise Source Licensing Program (ESLP) is a no-cost program that licenses Microsoft® Windows® source code to enterprise customers and state and local government organizations in eligible countries. Any organization that meets the specified criteria—and signs the source licensing agreements may access Microsoft Windows 2000, Microsoft Windows XP, and Windows Server™ 2003 operating system source code. Benefits of the program

- Providing insight and a deeper understanding of Windows
- Facilitating security and privacy audits and maintenance of customer's computing environment.
- Enhancing performance tuning, thereby allowing customers to adjust and optimize their own systems and related applications.
- Enhancing the pre-deployment engineering process for enterprise environments.
- Improving internal support and troubleshooting capabilities of deployed Windows systems.
- Improving the feedback mechanisms that ultimately contribute to the development of better Microsoft customer solutions and tools for the future.

All of the other licensing programs like “MVP Source Licensing Program” or “Government Security Program” includes pretty much the same benefits as stated above. This has been a recent initiative from Microsoft, the company recognizes some of the benefits of the OSS movement and how it could be used to its advantage by walking a middle ground.

In an article about code sharing an example about Citigroup summarizes in short the benefits of code sharing. Scott Preble, director of enterprise architecture and advanced technology at CitiMortgage Inc., foresees a day when a repository of objects that governs functions ranging from a login procedure to credit scoring can be snapped together to build applications at a fraction of the time and cost of starting from scratch.

Those are important benefits for Citigroup, which frequently acquires new businesses. If CitiMortgage, Citibank, Salomon Smith Barney, Travelers Insurance and the other businesses within the $112 billion giant shared Java components, the units would be better equipped to cross-sell products, Preble said. Citigroup and other big financial firms uniformly justify their large mergers by touting the potential of cross-selling.

The other alternative approach is the one taken by Novell, to mix proprietary software and OSS. Novell has taken various steps to make use of pre-existing OSS and scaled back development of its own proprietary software that competes with OSS.

Apache web server is the default web server for Netware 6.5. Novell participates in most of the OSS initiatives like MYSQL, rsync file utility, Perl scripting language etc.

From being a company which was forced out of the leadership position in network OS business by Microsoft. Novell has bounced back by making OSS a part of it strategy. Novell's strategy is not to make or compete with products that already have an OSS alternative, integrate it with its Netware products to give those extra features for half the price of what could be possible if Novell developed all the components in-house. Novell is trying to win back the market by giving the consumers OSS products combined with its own proprietary software.

The rapid announcement of Novell Linux Desktop 9 is very much a case of Novell putting its money where its mouth is. The OS is being positioned as an option for use in call centers, or at service desks, offering organizations a way to avoid lock-in of their desktop systems to Microsoft Windows. It comes pre-installed with the Novell Edition of OpenOffice.org, and Mozilla Firefox browser software.

Novell is also banking on its immunity from prosecution because it sold UNIX to SCO in the first place. Both IBM and Novell have made significant investments in Linux and open-source software over the past few years. Novell has also proposed to open up its patent portfolio to protect its consumers from litigations.

Novell has come up with a unique approach of using its position in the industry and trying to overcome the litigation scare that consumers usually face when going the OSS route by opening up its patent portfolio to make the customers at ease. Novell has said that In the event of a patent claim against a Novell open source product, Novell would respond using the same measures generally used to defend proprietary software products accused of patent infringement. Among other things, Novell would seek to address the claim by identifying prior art that could invalidate the patent; demonstrating that the product does not infringe the patent; redesigning the product to avoid infringement; or pursuing a license with the patent owner.

This is what Novell quotes on its site about patents and intellectual property rights "As appropriate, Novell is prepared to use our patents, which are highly relevant in today's marketplace, to defend against those who might assert patents against open source products marketed, sold or supported by Novell. Some software vendors will attempt to counter the competitive threat of Linux by making arguments about the risk of violating patents. Vendors that assert patents against customers and competitors such as Novell do so at their own peril and with the certainty of provoking a response. We urge customers to remind vendors that all are best served by using innovation and competition to drive purchasing decisions, rather than the threat of litigation."

Novell has previously used its ownership of UNIX copyrights and patents to protect customers against similar threats to open source software made by others. In the process of integrating OSS and proprietary software the company has brought down costs and successfully packed some punch in its products which were rapidly losing market share over the years.

Different licensing models set different rules on how the OSS can be used. A single licensing model might not suit every company's needs. In the next section we provide a detailed description of different Licensing models.

2 Open Source Licensing

In this section we examine the open source licensing quagmire from the perspective of a commercial software company. Once the decision to start an open source project has been made, a firm must decide on the license under which it will release its software. This is a complex decision that can have long-term and unforeseen implications on the project's success.

Open source licenses differ greatly from commercial software licenses in that they must not only set policies for use of the source code provided, but must also set policies for derivative works created from the source code. How an open source license sets these policies is of critical importance since open source licenses are alike. The choice of an open source license will reflect the philosophical position of an enterprise as much as its business strategy. As such, the license decision should be well informed.

We begin by examining the common characteristics of open source licenses. We then describe the major classes of licenses and give examples from each class. We discuss non-open source licenses that may be confused with open source. We examine compatibility between open source licenses. We discuss the process of creating a new license. We finish by discussing the decision making process that a commercial enterprise must undergo when choosing an open source license.

2.1 Characteristics of Open Source Licenses

There are a large number of open source licenses in existence, but most of these licenses share some common characteristics: clauses on derived works, statement of no warranty, statement of no liability, patent clauses, commercial use clauses, policies for interaction with other modules, and attribution statements (OSI). Not all open source licenses exhibit all of these characteristics, but nearly all contain a significant subset. In this section we discuss each of the common characteristics and discuss how they affect the behavior of a license.

2.1.1 Derived Works

The single most important characteristic of an open source license is how it addresses derived works. Derived works are source code or binaries that are based on modifications of the original source code. A key element of an open source license is that it permits the code recipient to modify the source. Licenses vary with respect to how derived works can be distributed. Some licenses may allow the recipient to keep a derived work only for their own use, but force them to distribute the modified source code in the event that they distribute a derived work or attempt to use it commercially. A clause that requires disclosure of modified source code under some conditions is known as a "forcing clause" (LeClair).

Licenses may contain ambiguities that make it difficult to interpret when the forcing clause is triggered. For example, if a company modifies open source software for its internal use it may not be clear if that company is under obligation to redistribute the modifications to the community. Other license may not have a forcing clause at all, allowing code recipients to make modifications to source code and commercially distribute the binaries without ever releasing the modified source code. Such licenses

often require that derived works include an attribution statement the work to the original open source distributor. Attribution statements are discussed in more detail below.

Lastly, a license may or may not have a viral nature with respect to derived works. The term “viral” indicates that the terms of the original license will also apply to any derived works. Non-viral licenses allow derived works to exist under any license of the modifiers choosing.

A commercial software company’s stance on derived works will be strongly influenced by its business model. Open source business models are discussed in detail later in this paper.

2.1.2 No Warranty

Most, if not all, open source licenses explicitly state that there is an absence of any warranty on the source code. A typical claim is that the code is provided on an “as is” basis and that there is no guarantee of quality. The distributor or the original source code assumes no responsibility for fixing bugs or for problems that arise due to defects in the software (OSI).

An open source license with a warranty leaves the door open for potentially unlimited liability. If a company wishes to provide a warranty for its source code, a commercial license is a better option. In no case should a commercial software company distribute open source licensed code without this type of clause.

2.1.3 No Liability

Most, if not all, open source licenses explicitly state that in no case shall the distributor be responsible for damages resulting from use of the code. This clause is necessary to release the distributor from any legal responsibility for what might occur as a result of using the code or its derivatives.

The absence of this clause in an open source license exposes a company to unlimited liability. While also common in commercial software licenses, this is an absolute must-have for an open source license.

2.1.4 Patents

Some open source licenses explicitly address the issue of patents. Often this is in the context of patent infringement litigation. A licensor of open source software may wish to prevent a licensee who has patented software derived from licensed code from going back and suing them or other licensees for infringing on their patent. They can do so by including a clause that terminates the license in the event that a licensee asserts any patents against the licensed code. This is a recommended clause for any open source license issued by a commercial software company.

Robert Gomulkiewicz, an associate professor of law at the University of Washington School of Law, points out that some open source licenses have a more activist stance on patent litigation, asserting that a license will be voided if the licensee asserts any patents against any code whatsoever, not just the licensed code. Because its scope includes code under other licenses, this type of a clause reflects a philosophical stance on patent litigation and is not necessary for most commercial enterprises. It is a more common clause in licenses issued by academic organizations.

Some open source licenses issued by patent holders or commercial software companies, such as IBM, include a clause that gives the licensee a royalty-free license to any patents that may cover the original source code. This is necessary to protect the licensee from claims that it owes license fees to the licensor. A company with its own patent portfolio may wish to include this type of patent clause to make it's software more acceptable to potential licensees.

Yet another type of patent clause is designed to absolve the licensor of any liability for patent claims brought against the licensed code. This is essentially a special type of "no liability" clause, discussed above, the forces licensees to take full responsibility for patent claims brought against the code. This is also a recommended clause for any open source license issued by a commercial software company.

2.1.5 Commercial Use

Many open source licenses address the issue of commercial use. Clauses on commercial use may be worded to either explicitly forbid or permit commercial use of the source code. Licenses may vary in whether they allow the code recipient to sell the original code, derived works, or related services to third parties.

The Free Software Foundation states that a "free software" license cannot prohibit commercial use (GNU). Many popular licenses fit this description and allow any time of commercial use of licensed code.

A smaller subset of open source licenses state that licensed code cannot be used for commercial purposes. This may include commercial distribution and/or the selling of related services.

A clause restricting commercial use can be helpful in preventing competitors from using open source code in competing products and services. IT can also be a serious impediment to the popularity of an open source project. These considerations must be taken into account when choosing whether or not to prohibit commercial use in an open source license.

2.1.6 Interaction with Other Modules

A number of open source licenses include clauses that explicitly permit code to be tightly coupled with other code or software that is not covered under the license without extending the license coverage to the uncovered software. This type of clause grew out of the desires of recipients of open source to be able to make calls to open source libraries without subjecting their own software to the viral nature of some open source licenses.

Licenses with interaction clauses are useful in situations where some properties of viral license are desired, such as a forcing clause, but where exemptions are also needed to allow licensees to integrate licensed code with unlicensed proprietary modules. The usefulness of this type of clause will depend on the particular type of software involved.

2.1.7 Attribution statement

A significant number of open source licenses require derivative source code to include a statement crediting the original creators if the code is to be distributed in any way. Most often such a license will give a specific statement that must be included.

This type of clause is a good idea of the original author wishes to receive some recognition or credentials based on their contributions. For a commercial software company it serves a purpose as a marketing tool, but should not be a requirement.

2.2 Classes of Open Source Licenses

Open source licenses fall into one of four major classes: strong copyleft, weak-copyleft, non-copyleft, and hybrid (Fishman). In this section we discuss these classes and provide specific examples of licenses within each class. The following table shows the four classes of licenses along with some examples of licenses from each class.

Table of Major License Classes

License Class	Examples
Copyleft	GPL
Weak-copyleft	LGPL, Mozilla Public License (MPL)
Non-copyleft	BSD License, MIT License, Apache License
Hybrid	Artistic License, Academic Free License

The table at the end of this section lists several popular open source projects along with their license and license class.

2.2.1 Copyleft Licenses

Copyleft licenses typically are the most free form of open source licenses and reflect the views of Richard Stallman, discussed in the previous section. Such licenses contain a forcing clause to ensure that derived works are shared with the community. A viral clause requires that derived works also be covered under the same license as the original source code.

The GNU General Public License (GPL) is probably the most popular open source license overall and is by far the most prominent example of a copyleft license.

2.2.1.1 GNU General Public License (GPL)

According to the Wikipedia (Wikipedia), as of April 2004 the GPL accounted for 74.6% of the 23,479 open source projects listed on Freshmeat.net. The GPL also accounted for 68.5% of the 52,182 free or open source projects listed on SourceForge.net. The popularity of the GPL is a reflection of it being one of the original open source licenses and its place as the definitive copyleft license. As a copyleft license, the GPL is viral and has a forcing clause. It does not prohibit commercial use.

From the perspective of a commercial software company, an advantage of the GPL is that it prevents other companies from modifying your work and profiting from it without disclosing the modified source code. A competitor can attempt to sell products that use GPL license code, but any competitive advantage will be neutralized by the forcing clause.

A disadvantage of the GPL is that the language is a bit ambiguous as to what activities trigger the forcing clause. For example, it is unclear if distributing modified

GPL licensed code within an enterprise constitutes distribution from the standpoint of the GPL. A commercial software company may view this ambiguity as a significant risk and a reason not to use the GPL.

Please see the appendix for a copy of the current GPL text.

2.2.2 Weak-copyleft Licenses

Weak-copyleft licenses are equivalent to copyleft licenses with some exemptions for derived works. Weak-copyleft licenses permit code recipients to maintain ownership of some types of derived works. The exact nature of the exemptions is specific to a particular license. Earlier in this section we discussed license clauses that address integration between licensed code and other modules. This type of clause is found in weak-copyleft licenses.

A derivative of the GPL called the GNU Lesser General Public License (LGPL) and the Mozilla Public License (MPL) are widely used weak-copyleft licenses.

2.2.2.1 GNU Lesser General Public License (LGPL)

The LGPL is very similar to the GPL except that it includes a clause that exempts tightly coupled code that calls LGPL licensed libraries from the viral clause and forcing clause of the GPL. This license should be used rather than the GPL if some licensees require exemption from the forcing clause and viral clause to maintain ownership of proprietary code used with LGPL code or libraries.

2.2.2.2 Mozilla Public License (MPL)

The MPL is a weak-copyleft license with a greater number of restrictions pertaining to patent infringement and intellectual property matters. It is less copyleft-oriented than the LGPL in that it offers more exemptions that allow licensees to maintain ownership of derived or integrated works.

2.2.3 Non-copyleft Licenses

Non-copyleft licenses are open source licenses that have unrestrictive policies towards derivative works. There is no legal obligation to share works derived from non-copyleft--licensed source code. Derivative works can be used and sold commercially, and even re-licensed under different terms. A typical requirement of non-copyleft licenses is that they require derivative code to include an attribution statement, described earlier in this section, crediting the source code’s original creators.

Prominent examples of non-copyleft licenses include the Berkeley Software Distribution (BSD) License, MIT License, and Apache License.

2.2.3.1 Berkeley Software Distribution (BSD) License

The BSD license is the most prominent non-copyleft license. It originated at the University of California at Berkeley and is associated with a number of high profile software programs that originated at that university, including the BSD UNIX derivative and the Berkeley Internet Name Domain (BIND) software. According to the license text, it permits “redistribution and use in source and binary forms, with or without

modification" subject to clauses about including a copyright notice and attribution statement in derived works.

Prior to 1999 the BSD contained a controversial clause that required derived works to place an attribution statement in any advertising materials. This was a controversial clause due to the fact that many enterprises found it a burden when attempting to commercialize BSD licensed software. It was removed in 1999.

Please see the appendix for a copy of the current BSD License text.

2.2.3.2 MIT License

The MIT license is very similar to the BSD license. The two licenses differ primarily in the advertising clause found in the BSD license prior to 1999. Other than that they are equivalent for practical purposes, though slight differences in wording exist (Laurent).

2.2.3.3 Apache License

The Apache license is essentially the same as the MIT license with minor differences in wording (Laurent).

A more complex and detailed update of the Apache License, Apache License v2.0, appeared in January 2004. This license is functionally very similar to its predecessor, bit attempts to resolve ambiguities with more detailed treatment of derived works and patent rights. Among other clarifications, v2.0 makes it clear that derivative works can be licensed under terms that differ from the original license.

2.2.4 Hybrid Licenses

A number of license exhibit a mix characteristics of copyleft and non-copyleft licenses and other restrictions. Examples include the Artistic License, first used by Perl, and the Academic Free License, a newer license aimed at academic licensors.

2.2.4.1 Artistic License

The Artistic License allows free distribution of the standard version of source code. The "standard version" is defined as the source code in unmodified form. It also allows free distribution of source code containing bug fixes and such code is still considered the standard version. Any other modifications must only be used within the modifiers organization, posted to the community, or the modified executables must be renamed to not conflict with the standard version executables.

Software in executable form may be distributed if it is the standard version, or if it accompanied with the modified source code if it is a non-standard version, or if non-standard executables are renamed and distributed along with the standard executables and documentation of the differences.

This license seems unnecessarily detailed in some respects and unnecessarily vague in others. It behaves like the GPL in that in some cases it requires you to share derivative source code with the community. But it behaves like the BSD license in that you can freely distribute derivative executables as long as they are renamed and not to be confused with the standard version.

For a commercial software company this type of license seems to try to do too many things at once. It would be better to focus on one set of objectives and choose a copyleft or non-copyleft license accordingly.

2.2.4.2 Academic Free License

The Academic Free License is similar to the non-copyleft licenses in that it offers the code recipient the rights to use the code as he or she sees fit. It differs in that has an additional clause that terminates the license if the code recipient files a lawsuit claiming patent infringement by other parties contributing to the software. There is also a clause that prohibits use of trademarks from the original work to promote derived works. These clauses keep the Academic Free license from being classified as a pure non-copyleft license.

2.2.5 Dual Licenses

An increasing number of open source projects that have commercial ties are licensed under dual licensing arrangements. In dual licensing situations, the same source code may be provided to multiple recipients under different licenses. The license granted to a particular recipient will depend on the characteristics of the recipient and the recipients intended use of the source code. For example, a dual license arrangement may permit the hobbyist and research communities access to source code under a less restrictive license than the one offered to commercial users.

Exmaples of projects using dual licenses include MySQL and the Open Office project. Please see the section in this paper on business models for a discussion of dual license business models.

2.2.5.1 MySQL

MySQL offers a dual license arrangement consisting of the GPL and a commercial license. With dual licenses MySQL can charge commercial customers to use code in proprietary projects without releasing modifications to the community. They can also reap the benefits of a GPL-style project, with many non-commercial users choosing the GPL license where it is less important to keep modifications out of the public view.

2.2.5.2 Open Office

Open Office offers a dual license arrangement consisting of the LGPL and Sun's Sun Industry Standards Source License (SISSL). In this case licensees of Open Office are offered a choice between two weak-copyleft licenses. The SISSL is less copyleft-oriented than the LGPL and is more appropriate for commercial licensees seeking to use the source code in proprietary projects.

2.2.6 License Disjunctions

Some open source software projects have chosen to license themselves under license disjunctions. This means that a code recipient has a choice between two licenses, and can choose to follow whichever one they prefer. Licensing disjunctions can be confusing and it is often not clear which license should be chosen.

An interesting observation about software covered by license disjunctions is that both of the most prominent examples are scripting languages. The prominent examples of projects using license disjunctions are Perl and Javascript.

In most cases disjunctions are confusing to the licensee and are not recommended for commercial software. It would be preferable to choose a single license that best meets the objectives of your enterprise.

2.2.6.1 Perl

Perl is license under a disjunction of the GPL and the Artistic license, both of which are discussed above. In cases where the user is seeking to use source code in a proprietary project, the Artistic License would be preferable. In cases where the licensee is happy to disclose source code modifications, the GPL would be suitable.

2.2.6.2 Javascript

Netscape's Javascript is licensed under a disjunction of the GPL and the Netscape Public License (NPL). The GPL is discussed above. The NPL is similar to the Mozilla Public License (MPL) with an added clause that permits Netscape to use your modifications in their proprietary versions of the program (GNU). The NPL almost seems written to discourage licensees from making significant modifications. The GPL would seem to be preferable to most licensees, except those who require an exemption to maintain ownership of tightly linked code. Please see the earlier discussion on weak-copyleft licenses for more details on the MPL and similar licenses.

Table of Popular Open Source Software Projects and Their Licenses

Open Source Project	Description	License	License Class
Apache	Web server	Apache License	Non-copyleft
BIND	DNS server	BSD	Non-copyleft
Emacs	Text editor	GPL	Copyleft
FreeBSD	Operating system	BSD	Non-copyleft
GIMP	Graphics tool	GPL	Copyleft
JBoss	Application server	LGPL	Weak-copyleft
Linux	Operating system	GPL	Copyleft
Mozilla	Web browser	Mozilla Public License (MPL)	Weak-copyleft
MySQL	Database	Dual (GPL/Commercial)	Copyleft/Non-Open Source
Open Office	Office suite	Dual (LGPL/Sun Industry Standards Source License (SISSL))	Weak-copyleft/Weak-copyleft
Perl	Programming language	Disjunction (Artistic License/GPL)	Hybrid/Copyleft
Sendmail	Mail server	BSD	Non-copyleft

2.3 Non-Open Source Licenses

In the recent hype surrounding open source software, some non-open source licenses may appear to be similar or offer the same benefits of open source licenses. A major example of this type of license is Microsoft's Shared Source license.

It is important to note that the Shared Source license does not give the recipient freedom to distribute or modify source code. It is a much more restrictive type of license and may be specific to particular product. The goal of Shared Source is to expose some aspects of source code to a select group of recipients while protecting intellectual property. It is more like a restricted commercial license for source code than an open source license.

The Shared Source license is discussed in detail in later sections of this paper that discuss business models and Microsoft's open source initiatives.

2.4 License Compatibility

License compatibility is an important issue to consider if software may be linked to or combined with software under other licensing arrangements. Not all open source licenses are compatible, making it illegal in some cases to link projects with incompatible licenses. In some instances license compatibility is required to attract developers to your project who may have a preference for a particular license or class of licenses. The Free Software Foundation maintains a list of popular open source licenses and comments on each license's compatibility with the GPL: http://www.gnu.org/philosophy/license-list.html. The LGPL, for example, is compatible with the GPL but the BSD and MPL licenses are not.

The importance of license compatibility is of interest to your company if you are seeking to attract a group of developers with or customers with specific open source license preferences. It is best to analyze the specific details of the licenses that interest you to determine their compatibility with other licenses.

2.5 Creating a New License

Experts recommend against writing your own license for legal and for marketing reasons. Customer authored licenses may have compatibility problems with existing licenses. Fear of an unknown license can make it more difficult to attract developers and make customers nervous. For these reasons we recommend that you first examine your business model and attempt to use an existing license from one of the major classes that most meets your needs. Try to determine if your needs are better met by a copyleft or non-copyleft license. Examine the licenses chosen my companies that are similar to yours. If a single license does not work for your enterprise, consider a dual licensing arrangement.

In the event that you must create a new license, it is best to base it on an existing license and clearly highlight how it differs from that license. This makes the license easier for potential licensees to understand. We recommend against using a license disjunction because it may also confuse potential licensees.

2.6 Making a Decision

This section has discussed the common characteristics and classes of open sources licenses, given specific examples of licenses, and discussed compatibility and the process of creating new licenses. When deciding on a license for your project, there may be both philosophical and business issues influencing your decision.

Your personal or corporate philosophy may align you more closely with a copyleft or non-copyleft approach to licensing. Is software something that should be shared with all, or something that should be kept secret? Your choice of a license will indicate your position on this issue to the open source community and will determine the developers and customers that will choose to ally themselves with your project.

The business component of your license decision should be driven by a thorough examination of your business model. The license you choose should fit the business model you plan to pursue. For example, if you plan to make money from support services, a non-copyleft license that gets your software into the hands of large corporate customers may be appropriate. If you are giving away the software to make money from selling compatible hardware, a copyleft license may make more sense.

The table below lists three examples of open sources licenses issued by commercial software companies.

Table of Open-Source Licenses Issued by Commercial Software Companies

License	**License Class**
IBM Public License	Non-copyleft
RealNetworks Public Source License	Weak-copyleft

In the case of IBM, the IBM Public License (equivalent to Common Public License) encourages commercial development of software based on the original source code. This in turn drives demand for IBM's hardware and services. A non-copyleft license best fit this need.

In the case of RealNetworks, the RealNetworks Public Source license was authored to foster a development community of media player software that would drive use of its proprietary codecs and server software. At the same time, it did not want to give an advantage to its competitors who may have used the code in competing proprietary player products. Weak-copyleft was most appropriate for this purpose.

These cases illustrate the fact that the license decision is closely tied to your business model. See the section of this paper on business models for a thorough discussion of open source business models. Your final choice of a license will reflect both your business model and your stance on open source software. With the information in this section and some careful thought you should be prepared to move forward with choosing a license.

The next section discusses complications that may arise when open source and proprietary projects interact.

3 Commercial Software and Open Source Coexistence

Open source is a viable option for some commercial companies, but there are a number of reasons commercial software companies will continue to move forward with proprietary source code bases. Proprietary source software companies need to take extra steps to protect their intellectual property and source code with the popularity of open source movement. Proprietary source code companies need to adopt policies for employees to follow to protect their source code, as well as take extra care and perform more due diligence in acquisitions that come with source code. Open source and proprietary source companies will co-exist and proprietary source code companies need to understand the risks involved and the options they have to protect the patents and copyrights for their proprietary code base.

3.1 Protecting Proprietary Source Code Copyright

Commercial software companies traditionally take great measures to protect their source code from being leaked externally. As mentioned in (Gomulkiewicz), "mass market software developers often treat the source code of their core products as the crown jewels of the company. They do this because their customers seldom need or want source code and competitors might gain free rider advantages from access to it." Customers pay large sums of money to have the license rights to use commercial software, over 178 billion US dollars in 2003 and estimated to be 189 billion in 2004 (Sharma).

3.1.1 Competitive Advantage of Proprietary Source Code

In general the price a company can charge for a product depends on what customers are willing to pay and what competing products are priced at. This basic business principle is true for software, for any commercial software product if the source code was available freely under open source or if competitors offered similar functioning programs, the price a company could charge would be greatly constrained and the number of customers captured by the company would be reduced by sales made by the competition to customers. Many software companies sell software where the sole revenue derived is in the initial license price; service related revenue is not a significant part of the business model. These types of commercial software companies are highly motivated to make sure their source code and the IP knowledge to produce that source code don't end up in a competitor's hands or an open source project.

3.1.2 Value of Proprietary Source Code

Proprietary source code with exclusive rights is usually very expensive to obtain and is an asset that is bought and sold between commercial software companies. Historically commercial software companies develop the source code for their commercial products from a combination of acquisitions of source code from other companies and by writing the source code in house. Computing a value on source code

in acquisitions is tough as often source code acquisitions usually come with other assets such as patents, an engineering and sales staff, and physical assets. Deals for billions dollars are common, as in Microsoft's acquisition of Great Plains and Visio or Oracle's current acquisition of PeopleSoft. The production of source code in house is also a very expensive enterprise – Microsoft itself will spend over 8 billion or development in 2004.

Proprietary source code is protected because of its revenue stream potential and value as an asset. Proprietary software is expensive for competitors to duplicate and having a unique offering of functionality allows a commercial software company to derive a substantial revenue stream. In order to protect that asset commercial software companies have developed practices to retain the privacy of their source code.

3.1.3 Traditional approaches to protecting proprietary source code

Even prior to the relatively recent popularity of open source commercial software companies had policies in place in order to protect the ownership of the corporate source code assets. In 1980, the extension of copyright protection to computer software was made explicit by amendments to the Copyright Act (Copyright Office). A number of standard policies that software companies require their employees to follow are:

- Signing an employment agreement stating all inventions, ideas and software written while in the employment of the company belong to the company.
- Agreeing to guidelines forbidding the distribution of source code or trade secrets which describe how difficult problems are solved to anyone outside the company.
- Enforcing access control policies to limit access to the source code to only those who need it in their job function.
- Employment agreements stating that a former employee must not work at a competitor doing similar work for a specified time period after leaving the company, usually one year, and will never divulge trade secrets after leaving.

In addition to enforce their copy right often a corporate license enforcement team is tasked to find pirated software copies and unlicensed users of software and then force them to pay or if they are unwilling then bring legal action against them to extract payment.

3.1.4 New risks with respect to open source

The current popularity of open source has produced a new set of issues for commercial software companies to address in protecting their source code ownership.

3.1.4.1 Preventing proprietary source code from leaking out

The most obvious risk is that an employee would place corporate source code into an open source project, or reproduce trade secrets in a new implementation for an open source project. Commercial software companies typically have clarified that contributing

corporate source code or implementations of corporate trade secrets into an open source project is forbidden, treating open source projects in the same manner they treat competitors. Some companies such as Microsoft have forbidden all employees from contributing to any open source project that isn't specifically sponsored by the company. This restriction can be seen as a clarification that of the employment agreement under which the employee agreed that all source code and intellectual property the employee produces belonged to the company. Since anything contributed to an open source project would not be owned by the company, it would be a violation of the basic contract.

3.1.4.2 Preventing GPL open source code from leaking in

In addition to protecting the leaking of private source code out, commercial software companies need to prevent the leaking of open source code into the commercial source code base. One type of open source license style often referred to as the viral, copy-left, or GPL license require that any incorporation of the open source code requires that the entire derivative product be released as open source software. Clearly open-source software released under this license would risk invalidating a commercial software company's entire ownership if it was incorporated into the corporate code base by one of its employees. Forbidding the incorporation of source code from an open source project that exists with the GPL license is an additional restriction all commercial software companies adopt if they want to protect their unfettered ownership rights.

3.1.4.3 Risk from incorporating BSD open source code

The other open source license style known as the BSD style license allows much more generous rights for corporations that produce derivative works from it. Under a BSD style license the source code can be modified or combined with private source code and the resulting source code can be kept private and the ownership rights retained by company producing the derivative work. Commercial software companies have to make a business decision with regards to building on or incorporating BSD style open source code into their private corporate source code base. The risk is the licenses offered for BSD software typically offer no warranties that the source code doesn't contain infringing IP, or wholesale copies of copyright code from corporations or from a GPL open source project. The contributors for an open source project usually aren't closely monitored and operating under an employee agreement, but rather are a large collection of people working from different locations making it difficult to verify all the source code is unencumbered. The reward for building on or incorporating a BSD style open source code base into a private commercial source code base is the free functionality gained at little cost.

3.1.4.4 Business trade-off when using BSD source code.

Large companies with large revenue streams for established software products need to be careful not to risk polluting their source code with infringing IP. Companies such as Microsoft forbid the incorporation of any open source code into their code base.

Only source code that has a warranty to have been produced with proper legal supervision so there is no risk of ownership or infringing IP can be used.

However a smaller company with limited resources may consider it a reasonable business risk to incorporate open source code with a BSD style license into their proprietary code base. The risk and reward needs to be weighed on a case by case basis.

3.1.4.5 Commercial software company risk for shipping open source products

Some of the legal risks for a commercial software company basing its product on open source are the possibility that the open source violates copyrights by having unauthorized copies of code in it, or that the code infringes patents. A recent example of the risks of basing your commercial software on open source is illustrated by many recent public copyright and patent infringement cases. Recently in September 2004 a copyright dispute between Furthermore Inc and the Miro/Mambo Group arose (Vaughan-Nichols). The Miro/Mambo Group was shipping a product based on an open source code base which Furthermore claims infringes its IP. Allegedly a contractor which worked for Furthermore placed code written for Furthermore into the open source Mambo code base, violating Furthermore's copyright. Furthermore has since contacted Miro and customers of Miro demanding payment and to sign a license. The fallout in the public press certainly hasn't helped Miro's business prospects for making future sales, independent of whether the claims are true or not. A more famous example is discussed in Chapter 5, detailing the SCO and IBM battle over Linux.

4 Open Source from a Business Point of View

In recent years, the open source movement has enjoyed phenomenal growth and gained enough credibility to become a viable alternative to commercial software in even the most mission-critical scenarios. In fact, the much publicized NASA Mars Rover project is operated using a suite of open source software (NASA). Even Microsoft admitted that it saw the open source software movement as a threat to its commercial business model (Microsoft). On the other hand, from a business point of view, not all open source projects are successful; many open source software companies have ironically switched away from pure open source models. At the end of the day, the question still lingers: "Should I open source or not?" In this chapter, I will try to make a business case out of commercial open source software, including the motives behind the adoption of open source, the impact of open source licensing, and the choice of open source business models, in an attempt to help you answer such a question.

4.1 Do I Really NEED to Open Source?

Before a company considers possible open source strategy, it first needs to evaluate open source from a business point of view, with clear advantages and disadvantages relative to the traditional model. Open source is not something you adopt because it is cool and everybody else seems to do it. If your business is doing absolutely fine in the sense that there are no existing issues that cannot be solved and no future opportunities that cannot be grasped within the traditional software development model, then the open source is probably not for you. With that said, businesses usually face challenges for which open source model can help.

4.1.1.1 Resource constraint

Businesses usually do not have unlimited resources devoted to their endeavors. It is the responsibility of the management to optimize the allocation of limited resources to maximize profit. Constraints of resources surface in various forms. It prevents a great idea from turning into reality; it hampers the quality of the product due to the lack of proper quality assurance; it forces companies to continue sustained engineering for years to come instead of targeting new markets and developing new products. Open source strategy allows you to enlist an army of outside developers and testers to work for you "for free".

4.1.1.2 Strategic competition

When well managed, open source projects can change the landscape of the competition in a way the traditional software cannot. First, there is the "free" factor. By offering a piece of software for free, the usage increases. Then there is the "peer" factor, which further increases the popularity of the free software. When something is popular, wonderful things happen. What you end up with is a product that packs increased functionalities, meets the customer requirements more precisely, is more reliable and integrates well with the rest of the world. In the end, your product competes better among peers, which in turn improves the standing of your company and motivates your employees to produce even better products.

If your business faces similar challenges and these benefits are of interest to you, there are a few aspects of open source software that you need to evaluate in order to come to the conclusion whether open source is the right solution.

4.2 Engineer a Successful Open Source Project

In order for an open source project to succeed, you need to enlist and sustain a group of outstanding people to develop and test the project for you "for free" (In some open source projects, a small set of core staff members are actually paid either full-time or part-time by supporting companies (Mozilla). However, the majority of the volunteers are not). This approach appears counter-intuitive, against traditional commercial software practices. While true altruism exists in this world, there is no assumption that the developers and testers at large are altruistic. Initially, they are often attracted to an open source project because it attacks a problem that they think is important and worth solving. Over time, some are further motivated by the commercial potentials of the project, such as fee based consultation, customization and support. Others expect to gain personal knowledge and prestige among the peer in the industry. In order to retain these resources, it is important to create a fair environment for them to work in. Those who contribute to the project should be properly acknowledged and rewarded with future monetary potentials. At the same time, those who did not contribute should not be able to take the software and make a quick profit off it. These desires are expressed in the form of a license for the open source software. The license specifies the terms and conditions under which the open source software can be used, modified and distributed.

4.3 Open Source Software License

The open source software licenses have been explained in details in the previous chapter. I will not reiterate here. It is important to realize that the choice of an open source license is not an independent decision. It needs to be made together with the choice of business model. In fact, the license is determined by the business model and it helps to make a business model work. Each type of open source license has its strengths and weaknesses. They reflect the designers' desire to "free" the software while ensuring the fairness and prevent misuse. If none of the open source licenses seem to fit your specific needs, do not be afraid to write one yourself.

What follows is a discussion of various open source business models together with the licenses that work for them. I will start by briefly look at the traditional software business model.

4.4 The Traditional Software Business Model

In traditional software model, revenue is realized when a proprietary software product is sold. In fact, what is sold to the customer is the right to use the software, instead of the ownership of the product. This is true even when a customer buys a shrink-wrapped box of CDs and manuals. The source code that the product is compiled from and the underlying technologies and techniques are intellectual property owned by the developer and protected by copyrights and patents. The developer controls the use, modification and distribution of the product through legally binding and enforceable contracts, commonly known as End-User License Agreement (EULA).

The right-to-use licensing has its advantages and disadvantages. On the one hand, right-to-use license is flexible and can be tailored for different customers. Once a license is sold, the incremental cost of selling additional licenses is low. And unlike some other types of revenues, the license fee can be recognized immediately after the product is sold. On the other hand, because right-to-use license is flexibility, it often becomes overly complicated and incurs substantially increased administrative cost for both vendors and customers. The immediate recognition of revenue for accounting purpose also means potentially rocky revenue stream.

4.5 Open Source Business Models

The ultimate goal for a business is to make profit. The goal of commercial open source software should be two-folded. First, it should provide increased values to customers. Second, it should bring more revenues and profits to the business in return. Open source software is usually licensed royalty-free to its users, allowing modifications and redistributions possibly under certain restrictions. This being the case, how can a company possibly make money with open source software? The short answer is that they have to look elsewhere. Over the years, many companies have tried different open source business models and people have proposed a few more that are of theoretical interest. Let us take a look at each of them in details. The names and basic definition of the business models are courtesy of OpenSource.Org (OpenSource.Org) and Hecker.Org (Hecker.Org).

4.5.1 Pure Open Source Business Models

4.5.1.1 Support Sellers

In “Support Sellers” model, software revenue comes not from the traditional software license fees, but from the packaging, distribution and after-sale services such as custom development, training, consulting and support. This is the original free software business model advocated by Richard Stallman in the GNU Manifesto (Stallman) and is still one of the most popular models adopted by companies involved in open source.

Walnut Creek CDROM is an early example. Back in the days when slow dial-up was the predominant method of access to Internet, Walnut Creek CDROM gathered public domain and freely available software, compiled them and sold them in CDROM format. Although the same software was also available for download, the slow Internet access speed made it impractical. The value of this model at that time was the convenience of compiled package and easy access.

Nowadays, the most notable example is Red Hat / Fedora. Red Hat is the leader in development, deployment, and management of Linux and open source solutions for Internet infrastructure. The Red Hat Linux distribution generates no license revenue for Red Hat. Instead it makes money in maintenance and services in the form of subscription. For example, the enterprise Linux subscription features Red Hat Network, support, training, consulting and custom engineering. According to Red Hat, the subscription model is the most effective way to deploy, manage and evolve the rapidly innovative open source technology (Red Hat). Other companies such as Novell and Covalent have similar business model.

This was also the business model adopted by a lot of the early open source companies. All hoped to make money by giving away source code and offering services and support. While a few have succeeded thus far, most others have failed. The reality is that there are just not that many lucrative support contracts to sustain the businesses. A good example is VA Linux. With revenue for Linux support not meeting expectation, they switched away from Linux and even changed its name. It pays to conduct thorough business analysis before adopting such a model.

Most "Support Sellers" companies use GNU General Public License (GPL), but most if not all open-source licenses would work for this model.

4.5.1.2 Loss Leader

In "Loss Leader" model, a usually free open source software product serves as a loss leader for other more traditional software offerings such as extensions or professional versions. The purpose of the open source product is to improve the acceptance of the product base and to increase the market for the commercial ones, which generates improved revenues.

For example, Sendmail is noted for its popular mail transport agent (MTA) software. It maintains an open source Sendmail MTA and makes money by providing a commercial version as well as commercial support services. Of course, the commercial version is based on the open source Sendmail MTA, with additional third party proprietary components. Sendmail also sells an easy-to-use administrative console, spam filters, e-mail retention modules and other enterprise level enhancements.

Unlike "Support Sellers" model, the choice of license needs to be more careful here. If the open source product shares source code with the proprietary offerings, then the choice of license should allow both the distribution of such source code with the open source product and the use of such source code in commercially licensed proprietary product. As a result, GPL like licenses should be avoided due to its viral nature. A BSD like license should suffice. If more control is desired than MPL-like license can be considered as long as there is clear separation (per license definition) among the components.

4.5.1.3 Widget Frosting

"Widget Frosting" model applies to companies that are primarily in hardware business with software as an integrated part of the offering. They adopt open source model to improve the quality of the software while reducing the cost. In the extreme case, the whole software platform is replaced with an open source one for additional strategic appeal. It helps increase sales of hardware and boost revenues.

IBM is the leading example of the companies that follow this model. It offers Linux on its entire line of hardware, with annual sales exceeding 1 billion dollars. In addition to the increased sales, there are strategic motives. IBM pushes for various standards which speed up adoption and help it compete in more entrenched markets. It seeks to commoditize desktop and server operating systems, eliminating major revenue streams of companies such as Microsoft and SUN. Most major original equipment manufacturers (OEMs) have adopted this strategy to some extent.

4.5.1.4 Accessorizing

The “Accessorizing” model is for companies that indirectly benefit from open source movement by distributing books and other materials and promotional products such as T-shirts and mugs (as opposed to products and services in “Support Seller” model). These companies usually do not directly participate in the open source projects.

O'Reilly Associates began to print existing open source manuals. In the mid-1980s it began a successful series of original books about open source packages. Lately, it also expanded into training services via conferences and tutorials.

4.5.1.5 Service Enabler

In the "Service Enabler" business model a company creates and distributes open-source software primarily to support access to revenue-generating on-line services. To make this model successful, the back-end services need to be unique and valuable and can not be easily duplicated by competitors. The client can then be licensed under open source to increase the popularity of the services. Netscape originally used this model and made its open source browser client access Netcenter services which generated revenue from advertising. Slashcode from Slashdot is an open source project that allows readers to access Slashdot’s subscription service.

The license that works well with this model should prevent others from unfairly covert open source into proprietary software and profit from it. GPL and MPL are good candidates.

4.5.1.6 Sell It, Free It

In "Sell It, Free It" model, a software company would release a piece of software first as a traditional commercial product. Then at the point when the benefit of open source out-weights the development cost and revenue from licensing fees, it is converted to open source product. Besides cost consideration, the newly open sourced product may be strategically positioned as a loss leader, which benefits the rest of the offerings from the same company. The process can iterate over and over again, each time with another product.

After both Netscape browser (Project Mozilla) and Sun StarOffice (Project OpenOffice) lost in their respective competitions to Microsoft Internet Explorer and Office Suite, they were converted into open source projects with the hope that the reincarnations would be better positioned to compete with the archrivals from Microsoft. Interbase / Firebird from Borland was another example of “Sell it, free it” model.

4.5.1.7 Brand Licensing

In the "Brand Licensing" model a company makes the software product itself open source but retains the rights to its product trademarks and intellectual property, and charges other companies for the right to use those trademarks in creating derivative products distributed under the exact same brand name.

JBoss is licensed under Lesser Gnu Public License (LGPL). Companies and individuals are entitled to use and bundle JBoss freely. However, they do not have the right to brand their products “JBoss” or use the JBoss trademark in any way. JBoss offers special licenses for purchase to do just that.

4.5.1.8 Software Franchising

If a company with good reputation wishes to expand its business, it is possible for it to grow not through direct hiring and acquisition but rather through franchising. In other words, the company would authorize other developers to use its brand names and trademarks in creating associated organizations doing open-source support and custom software development in particular geographic areas or vertical markets.

4.5.2 Hybrid Business Models

The business models discussed in the previous section are based on pure open source. In practice, many companies take the best of both open source and traditional business models and come up with hybrid models that target their markets effectively. The strength of hybrid model is to be able to generate revenue from license fees instead of indirectly through other means. The flip side is that, since it is no longer pure open source, it can no longer enjoy the full benefit that open source brings.

4.5.2.1 Shared Source

If customers decide that having access to source code is a requirement, vendors may make it available, but without necessarily allowing the customer to publish or make changes to the code without consequences, provided this is all that is needed to address the needs and concerns of the customers. In the old mainframe days, IBM routinely gave big customers source code. Microsoft, in order to pitch a more trustworthy platform, solve high priority technical issues, and in general respond to the challenges from open source competitions, makes its Windows and Office source code available to a select group of customers and government entities.

4.5.2.2 Dual Licensing

Many widely known and well respected open source projects, including MySQL, Qt and OpenOffice use dual licensing as a key part of their business model, they give away their software as open source under the first license while generating revenue by charging clients for the same software under the second license. For the developer, the assumption is that some customers need the second license because they want to modify the software for competitive reasons and keep the source code of the modifications secret.

The biggest motivation for dual licensing is making money in the form of licensing fees which is a more effective revenue stream than services and support. Another important reason is to ensure compatibility with other open source licenses. Dual licensing doesn’t work on all software projects. For example, if your project contains source code that is GPL licensed, then you are not permitted to make your project proprietary and charge for it. You need to have control over the intellectual property by either owning it or licensed from third-party. For the free license, GPL is the most commonly used because it prevents someone else from making money off the project. Since the crux for the dual licensing model is to separate the two group of customers, those who do not pay and those who are willing to pay, it is important to choose the commercial license based on the types of uses of those potentially paying customers. MySQL commercial license grants the right to distribute modified software without

releasing the source code to the general public. OpenOffice licenses the source code and documentation separately.

Because the open source license is based on GPL, it also means that you cannot simply take code out of the open source tree and apply it to the proprietary tree (See "Commercial Software and Open Source Coexistence" chapter for details). As a result, you may get into a situation where the open source version has better quality than the proprietary one because the open source version tends to be scrutinized by more pairs of eyeballs. Therefore it is wise to invest resources to evolve the proprietary tree aggressively and closely monitor the code change in proprietary tree to make sure there is not tainting.

In theory, dual licensing may lead to other companies starting a competing development project using the GPL-ed or other open-source software. However, in practice, the potential gains from the use of dual licensing as a marketing tool for widespread adoption may outweigh this risk.

4.5.2.3 Commercial Value-added

As discussed above, the open source models in which software is given away and money is made by providing packaging and distribution and after-sale services has indeed been demonstrated to work. Small firms have been making a living this way. However, it's hard to scale up in revenue with this type of models. Instead, many larger commercial software companies favor offering proprietary middleware or applications that run on open source software such as Linux as if they were ordinary (not Open Source) software. Oracle's Linux involvement began in 1999 with the first commercial database on Linux. Today, Oracle collaborates with and provides first-line support for Red Hat, Novell, and Asianux, and all Oracle products are available on Linux platform (Oracle).

4.5.2.4 Commercial Enhancement of Open Source

In this model, the base open source software is extended commercially by making modification to it. Such use of modified code is obviously subject to the licensing of the base software. It works well with BSD style licenses. However, under GPL such commercial derivative is not permitted. However some companies push the limits of GPL by claiming that the additions are sufficiently separate from the original GPL software. For example, some vendors provide commercial products that are dynamically linked to the GPL-ed Linux kernel or are shipped as binary modules for their customers to link into Linux.

One example of this model is BSD/OS operating system. BSDi developed the product using the Net/2 and 4.4BSD-Lite releases from Berkeley. It filled in missing components and made many other modifications that were licensed; added other commercial products; and provided a complete, supported release sold with a standard commercial license.

4.6 Conclusion

Open source software is, from a business point of view, nothing magical and mythical. It follows the same set of business rules as traditional proprietary software. It is not a panacea for everybody. If implemented properly, it gives you a competitive

advantage that is hard to match by traditional software. Open source software business models vary. However, one thing is in common: There is no free software. There are all kinds of costs that go into developing open source software, such as time contributed by the developers and the subsequent cost of integration, patches, upgrades and technical support. Such effort can not be sustained without the generation of revenues somehow.

The open source software approach can be both difficult and strategic. Many customers have a misconception of the open source movement and have unrealistic expectations. At the same time, opportunities abound as most customers judge a solution by its quality, not whether it is open source or not. If neither commercial nor open source business models seem to suit your case, then a hybrid model may be the way to go. Remember it does not have to be all or nothing when it comes to open source. One may take the strengths of open source and combine with traditional software revenue and create a niche market for the success of your business.

5 Case Study: SCO v IBM and Open Source Legal Risks

5.1 Introduction

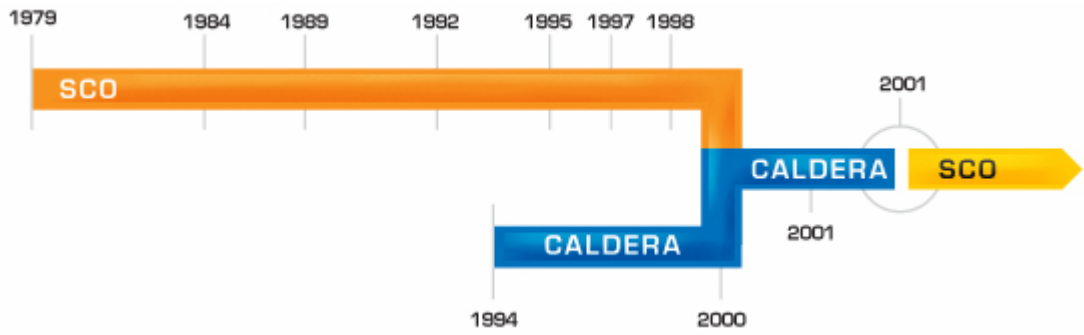
This section will highlight the legal and market implications of open source on commercial software companies. Primarily focused upon open source licensing legal risks and the SCO Group lawsuit against IBM. How will upcoming legislation from the SCO v IBM court case affect other companies towards their use of open source? What are the imperative issues and conditions of the case? What is SCO business model? What are the various reactions of companies based on the results of the cases?

5.2 Open Source Licensing Legal Risks

There are many legal risks and legal problems that take place when commercial software companies want to use open source licenses and strategies. The validity of the licenses and "their applicability under national laws" creates many enforceability concerns. Copyright protections and patents approach the problem of the "author's exclusive rights, along with international treaties (WIPO), Berne Convention (Paris Act), and the Trade Related Aspects of Intellectual Property Rights (TRIPS), but presents debatable concerns when delivering rights to "original source code", that can be altered, distributed with out compensation to the authors. The "Finnish Copyright Law" attempts to give economic rights for the author, but does not assign moral rights. Many software patents are "infringed upon" due to similar source codes used for open source programs that are available in a free use format. Commercial software companies have to determine an assessment tool to weigh the legal risk of open source licensing and innovations techniques for future consumers. Evaluating the legal risks for open source licenses and strategies is very complex. A dual conclusion is surmised for protecting author rights, and to ensure that if any software is used for the open source community to analyze, and recreate the author rights are waived.

5.3 The SCO Group v IBM

Doug and Larry Michels founded the Santa Cruz Operation (SCO) in1979 that began as a UNIX system porting and consulting company. Providing small businesses with the first affordable "business critical computing system". SCO creates a 4 billion dollar market working with application developers, computer manufacturers, resellers, and distributors to run SCO server software on Intel processors. In 1995, SCO acquires UNIX system and Unix Ware from Novell Corporation, who acquired it from AT&T Laboratories. The most advanced operating system was developed for an Intel processor titled Unix Ware 7.1.3 that was formed from these acquisitions. Caldera systems acquire the SCO's server software in 2002 with Darl McBride as CEO; they change their name to The SCO Group (SCO). SCO is one of the fasting growing Northern American Technology Company in the world. On March 6, 2003 they filed a civil lawsuit



against IBM. SCO is “citing misappropriation of trade secrets, tortuous interference, unfair competition and breach of contract”. UNIX is “ a computer operating system” SCO/UNIX is a modification of UNIX (System V Technology) and other related software developed by SCO. SCO provides licenses to various software vendors. The licenses that are granted “ impose restrictions and obligations on the licenses designed to protect the economic value of UNIX and SCO/UNIX. The core of this case is for SCO rights in regards to their “proprietary software misappropriated and misused in violations of its written agreements”. Therefore, IBM has misused their licensing agreement with SCO in the following manner:

- misusing and misappropriating SCO’s proprietary software;
- inducing, encouraging, and enabling others to misuse and misappropriate SCO’s proprietary software; and
- incorporating (and inducing, encouraging, and enabling others to incorporate) SCO’s proprietary software into open source software offerings

The SCO business model consists of inexpensive Intel processor chips, to add processors or servers when increase processing is needed, and for adding or changing functionality. The combination enabled SCO to garnish great relationships with business partners “such as CitiGroup, K-Mart, Cendant, Target Stores, Texas Instruments, BMW, Walgreens, Merck, Sherwin Williams, Radio Shack, Auto Zone, British Petroleum, Papa John’s Pizza, Costco”(Ref. 2), which in turn created an atmosphere for great economical progress for reporting data of high volume transactions and information simultaneously. The business model has a simple approach, to enable massive vendors to incorporate a functional transactional process with an Intel processor chip, while keeping exact records of transaction data.

IBM approached SCO “to jointly develop a new 64-bit UNIX-based operating system for Intel-based processing platforms” this venture is known as Project Monterey. IBM with no knowledge or expertise to run UNIX on an Intel chip was strictly confined to its “Power PC chip”. SCO accepted the venture and therefore invested into a development team, and money. In return IBM engineers would learn “valuable information and trade secrets with respect to architecture, schematics, and design of UnixWare and the UNIX Software Code for Intel-based processors”(Ref. 2). Once the final “technical aspects” of the project was completed, all that was left was the marketing and branding. IBM pulled out the venture and refused to go forward, but tried to destroy all of UNIX “marketplace value” with the business partners that were mentioned earlier,

by creating a “revenue model based on licensing of software rather than sale of services”. IBM’s business model presents to undermine the value of Unix consumers by presenting this revenue ideology instead of sales approach that they both were working towards. Based on its value in the marketplace, UNIX has become the “most widely used and widely accepted operating system for enterprise, institutional and manufacturing applications”(Ref. 2) throughout the world, and IBM is trying to step in and influence the market. Therefore, IBM entered into an agreement with Red Hat, Inc to bundle their Linux based software. IBM’s VP Robert LeBlanc states their perspective on their agreement with SCO, *“Project Monterey was actually started before Linux did. When we started the push to Monterey, the notion was to have one common OS for several architectures. The notion actually came through with Linux, which was open source and supported all hardware. We continued with Monterey as an extension of AIX [IBM UNIX] to support high-end hardware. AIX 5 has the best of Monterey. Linux cannot fill that need today, but over time we believe it will. To help out we’re making contributions to the open source movement like the journal file system. We can’t tell our customers to wait for Linux to grow up*”. Project Monterey plays an important role in the SCO v IBM case because it presented the information to IBM, which they misused the UNIX Software Code, which makes them in breach of contract. SCO has plenty of enforcement rights, which are as follows,

- Rights under trade secrets and developer agreements involving SCO Open Server;
- Rights under customer licensing agreements involving SCO Open Server;
- Rights under trade secrets and developer agreements involving SCO UnixWare;
- Rights under customer licensing agreements involving SCO UnixWare; and
- Rights under all other original UNIX licenses issued by AT&T Technologies and its successors.

Many events were happening in the open source community that makes the legislation outcome of this case impact how open source is viewed in the future. IBM misappropriated the trade secrets from SCO server, and UnixWare in their shared libraries for their own benefit that devalues SCO in the market. The damages from this misappropriation are unforeseeable. The conduct of IBM presents an unfair competition of business practices. Eric S. Raymond an anthropologist, and hacker specialist views this case as “ tragedy proceeded farce” and that “Caldera doesn't have a case. Its public statements are not just full of lies; they're full of lies that can easily be falsified just by looking at Caldera's own past press releases. A number of these are in the Open Source Initiative's position paper on the lawsuit.” The open source community should wait patiently for the outcome because the SCO v IBM case is still pending. The information technology world will determine if using open source code for the art of expression and change, or thus for profit and innovation so the good in technology as a whole will or will not be with the justification of this case. There are many reactions to this case, and a close eye will be on the outcome. It’s great that SCO is pushing the envelope of open source licensing agreement, trade secrets, and the manipulations of the judicial system.

6 Case Study: Microsoft Windows and Open Source Platforms

6.1 Introduction

Now that we have examined the different advantages and disadvantages of open source software as well as the different licensing models, we will take a look at a specific company, Microsoft, and study how it is has reacted to open source software. We will focus on the operating system market and examine how the open source phenomenon has affected the Windows operating system as well as other proprietary operating systems. Finally we will look to the future and discuss how Microsoft may change its licensing model to continue to maintain its market dominance in the operating system market.

6.2 Microsoft's Historical Position

Microsoft has always viewed open source software as a competitive threat, but its public position on the subject has changed quite substantially over time. This public position has also been quite different from internal thinking by the company's leaders, as has been evidenced by leaked internal memos. In the late 1990s, Microsoft was very aggressive in its attempts to negatively characterize open source software and the public relations (PR) division of the corporation attempted to portray open source software as generally inferior to its closed source or proprietary counterparts. However, the leak of several confidential memos at the end of October 1998 showed that internal thoughts on the subject were much different than what was being externally espoused. These memos, dubbed "The Halloween Documents", were published and analyzed by Eric Raymond. His analysis highlighted several key quotes from the documents:

> OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.
>
> Recent case studies (the Internet) provide very dramatic evidence that commercial quality can be achieved or exceeded by OSS projects.
>
> OSS is long-term credible ... FUD tactics can not be used to combat it. The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale (Raymond 1).

These memos caused quite a stir in the press and in the technical community because Microsoft was forced to acknowledge that many of its public claims about open source were invalid and that open source was a real threat and competitor. However, these events didn't stop Microsoft PR from attempting to spread fear, uncertainty, and doubt about open source software. In May of 2001, Steve Ballmer famously referred to Linux as "a cancer" (Greene 1). Whether comments like this were simple mistakes or part of a concerted effort is unknown, but at the same time initiatives were being started

inside the company that showed Microsoft had different feelings on open source software. In December of 2001, Microsoft issued a press release to announce that the Austrian Ministry of the Interior was granted access to the Windows source code as part of the Shared Source Initiative ("European Ministry" 1). Microsoft claimed it had been licensing source code since 1991, but this press release marked the formal announcement of the Shared Source Initiative. The Shared Source Initiative (SSI) is a program that gives selected institutions access to the source code of Microsoft products. Participants in the program are allowed to view the source, but are not allowed to modify the code or share it with others. Today, the SSI has over 1 million participants and Microsoft uses this initiative to compete with open source software. The participants in the SSI are mostly governments, academic institutions, and large customers. Although the most common use of the SSI grants an institution private access to current and past versions of the Windows operating system, other Microsoft products have also been added to the SSI over time. The most recent addition is the Rotor project. The Rotor project is an implementation of the Common Language Infrastructure, which is the backbone of Microsoft's .NET development platform. The source code for Rotor is freely available to the public and Rotor is the first computing platform that Microsoft has released to the general public under the SSI (Jepson 1). The release of Rotor through the SSI was also interesting from a platforms standpoint because Microsoft's two major platforms, the operating system and the development platform of the future, are now available in some form through the SSI.

Although Microsoft was developing and extending the SSI in 2001, it was by no means embracing open source. Press releases were issued by Microsoft which complained about the "viral" nature of the GPL and which described the benefits of the more limited SSI over other open source licenses ("Microsoft Approach to Source Code" 1). Microsoft stuck to this strategy of advertising the SSI as superior to open source licenses and attempting to inform customers and the press about the negative aspects of open source licenses fairly constantly during the early 2000s.

More recently, Microsoft has begun experimenting even more with true open source software and different licensing models. In April of 2004, Microsoft shocked much of the technology community when it released a new product to the public under a true open source license. The product, WiX, is a tool that helps developers write installation programs for Windows. WiX was published on sourceforge.net, one of the premier open source websites, and the source was released under a license called the Common Public License (CPL). The CPL is a BSD-style license that grants users the rights to use, modify, and distribute modifications of the code. Even though WiX is a small product, releasing it as open source was a huge deviation from Microsoft's past comments on open source and even from the SSI (Bishop, "Microsoft Notebook" 1). Soon after WiX was released, Microsoft released two more products, FlexWiki and WTL, also as open source products under the CPL. WTL is very similar to WiX in that it is a development tool, but FlexWiki is an actual product aimed at end users (Foley 1). This was a major milestone for Microsoft in that it released a non-developer product with some revenue potential under an open source license. It is interesting to note that Microsoft's motivation for releasing these products as open source was primarily to further the development of them and increase their adoption (Foley 1). In the case of the tools, Microsoft has a desire to provide very high quality tools for Windows developers

in order to further the platform and to make it more attractive to consumers. On the other hand, the reason for open sourcing FlexWiki seems more altruistic. Wiki technology is relatively new and the release of FlexWiki as open source appears like a genuine effort by Microsoft to improve and increase adoption of the technology for the technology's sake.

Next, in September of 2004 Microsoft agreed to allow customers to view the source of its Office productivity suite in addition to the operating system through the SSI. This move was seen by many analysts as a direct attempt to compete with the rising popularity of open source productivity suites, especially in developing countries (Linn 1). The effect of the move was important; today the source code of both of Microsoft's flagship products, Windows and Office, as well as that of its next generation development platform is available under the SSI.

It would be difficult to argue that Microsoft has not responded to market pressures and increased the access it grants to the source code of its products. To date, this access has mostly been aimed at large institutions, especially the type of institutions that can help or hinder Microsoft's bottom line. On the other hand, the release of several small products on sourceforge.net has shown that Microsoft is now willing to utilize open source licenses. The company is definitely more aware of and open to source code access that is has been at any time in the past.

6.3 Other Proprietary Platforms

Microsoft is not the only proprietary software corporation that has had to develop a response to open source software. In fact, almost all of the other major players in the operating system market that began as pure closed source organizations have made some changes to their business and licensing models in order to better compete with open source software. Most notably, Apple and Sun Microsystems have adapted their existing operating systems, both in licensing model and in the actual architecture of the software in reaction to open source. There have also been completely new operating systems, such as Symbian, which have used pieces of the open source model in a creative way.

6.3.1 Apple

Apple has released its latest operating system, OS X, under an open source license. In 1997, Apple decided to use Mach and elements of BSD Unix, both of which are open source, as the basis of its next operating system. Mach would provide the kernel of the operating system and BSD Unix would supply the file system, networking capabilities, and other basic building blocks. Four years later in 2001, Apple shipped Mac OS X which is based on that open source core. At the same time, Apple released the open source components of OS X as a standalone entity called Darwin. Apple's release was interesting, in that it shared the core source of its operating system while keeping the upper layers of the code as proprietary. This basic diagram from Apple shows how the open and closed source pieces of OS X fit together.

Copyright © 2000 Apple Computer Corporation

Darwin provides the core of the operating system and both Apple developers and individuals can perform work on it. The pieces of the operating system that are most important to Apple, like the new graphical user interface Aqua, are kept proprietary and closed (West 1269-1271). By adopting this type of architecture, Apple has improved its position with users running Linux or with those who merely favor open source products while still protecting some of its most valuable intellectual property (Story 1). This architecture is not without faults, however. Even though Darwin can be considered a full operating system, it is very limited. It does not contain a full graphical user interface and it provides no functionality or features that can't be found in BSD. Because of this, the Darwin project has not taken off as swiftly Apple would have hoped. As of December 2004, the Darwin project has only 95 registered contributors (OpenDarwin.org 1)

6.3.2 Sun Microsystems

Recently, Sun has been making open source news with its Unix-based operating system, Solaris, but the company actually has a long history of open source involvement. Sun has always been a company centered on networking and as such has always leaned toward open standards. Initially, Sun chose to keep most of its actual source code closed, but that position changed slowly over time. In 1999, Sun decided to release the source code to its Java programming platform under a partial open source license. Java is very similar to Microsoft's CLI; they are both cross-platform programming environments. The license is called the Sun Community Source License (SCSL) and it allows users to modify and redistribute the source code. However, the license requires that royalties be paid to Sun if the redistribution is part of a commercial product. Next in 2000, Sun released the source code to its office productivity suite, StarOffice, as well. Sun took a slightly different approach with StarOffice. The source code was released and a new organization, OpenOffice.org, was created. The source code is public to use or modify. Anyone, including Sun, can take the code from OpenOffice and use it in a commercial or non-commercial product (West 1266-1277). Thus, by the end of 2000, the source code for Sun's programming platform as well as its office productivity suite was available but its operating system was still very much proprietary. This has begun to change very recently as Sun has made several announcements in late 2004 regarding plans to open

source Solaris. First, in June of 2004 Sun's COO Jonathan Schwartz made the initial comment that Solaris 10 would be released under an open source license.

"I don't want to say when that will happen," Schwartz said in a press conference in conjunction with the company's SunNetwork conference. "But make no mistake: We will open-source Solaris." (Kait 1).

Later, in an interview with eWeek, Schwartz hinted that Solaris would be released under a license that was a hybrid of the GPL and BSD styles (Galli 1). Sun has also made an announcement that they will offer legal protection from patent infringement lawsuits to users of Solaris (Shankland 1). As of this writing, Solaris 10 has not been released and the final licensing model is still unknown. However, Solaris is planned to be released before the end of the year and by 2005 one more major platform will be open source.

6.3.3 Symbian

The final platform we will look at is Symbian. Symbian is the operating system that powers many common mobile phones and portable devices. It is interesting for several reasons. Symbian was created in 1998 as a private company owned by Ericsson, Nokia, Motorola, and Psion. The main purpose of Symbian's formation was to create an alternative OS to Microsoft's Windows Smartphone platform (Krazit 1). Today Symbian is owned by Ericsson, Panasonic, Nokia, Samsung, Siemens and Sony Ericsson. Symbian is not a true open source operating system. The source code for the system is not available to the public in any way, shape, or form. However, the fact that Symbian is owned by several large cell phone makers essentially means the operating system is open source to that select group. Secondly, Symbian has a very extensive partner program. A Symbian partner is usually someone who is writing an application or service for the Symbian platform. There are two levels of partnership, the affiliate and the platinum level. What is interesting is that partners at the platinum level have access to some of the Symbian OS source code. Currently, there are 212 platinum partners.

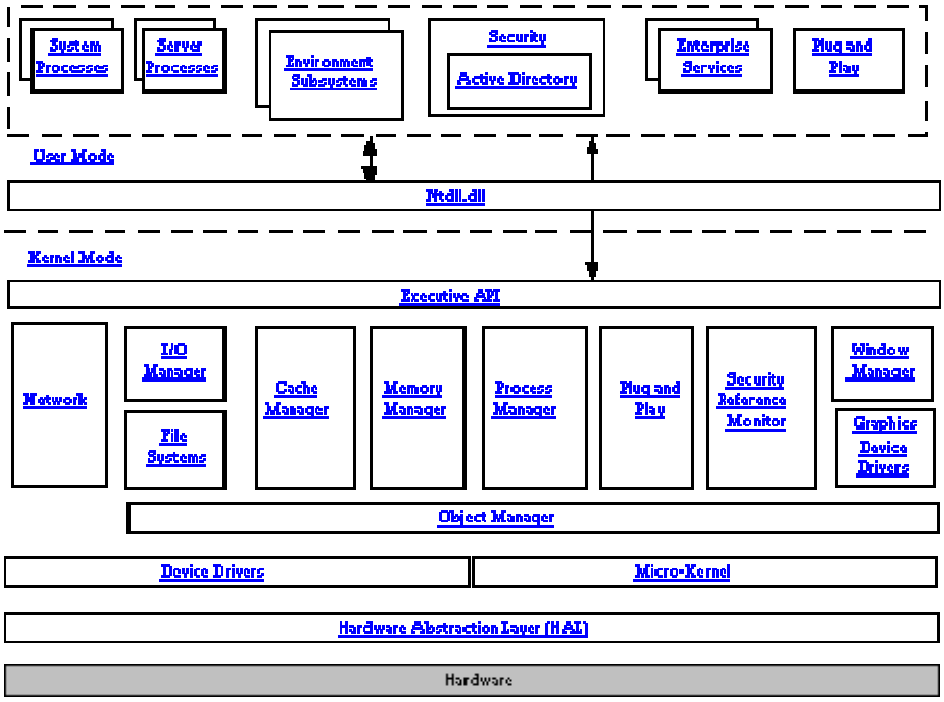
6.4 Possible Open Source Strategies for Microsoft

As we have seen, most major platform vendors offer some component of their platform as open source. Apple has the Darwin core. Sun has Java and plans to license Solaris 10 as open source. Symbian allows many partners to view the source code of the operating system. Given all this, is it possible that Microsoft will move even more in the direction of open source? Microsoft definitely has more open source momentum lately, so let's examine the possibilities of what Microsoft could do with Windows.

6.4.1 The Windows Architecture

First, we must look at the architecture of Windows and examine and how Microsoft could open source the operating system. The obvious suggestion is to take the route of Linux or Sun and open up the entire operating system with a true open source license. This seems unlikely given that Microsoft is still extremely protective of its intellectual property and that the business model of selling Windows as a proprietary operating system has been extremely profitable thus far. However, when you look at the structure of the Windows operating system, it becomes clear that Microsoft has several

possibilities in terms of granting access to the source code. The following diagram from Microsoft shows two major aspects of the Windows architecture. Windows is divided into layers with essential, low-level tasks at the bottom and more abstract, higher-level processes built on top of those lower layers. In addition to these layers, Windows is also divided into functional subsystems. Each subsystem supplies a relatively isolated piece of functionality and the subsystems communicate through well-defined interfaces. These two key features of the architecture give good clues as to how Microsoft could control access to the Windows source code.



Copyright © 1999 Microsoft Corporation

6.4.2 An Upper Layers Strategy

The first option is to open up the upper layers of Windows. One can imagine Microsoft open sourcing the user-mode code of Windows along with the executive API that sits on the user-mode kernel-mode boundary. There are several reasons for doing this. First of all, this code is what most application developers and users of Windows interact with directly. By doing this, the code that makes up the meat of the entire Win32 API would be exposed and developers would have the type of access to Windows of which today they can only dream. This would be extremely helpful to those writing or debugging applications on Windows. Secondly, this would protect some of the core kernel code that differentiates Windows from Linux such as the NTFS file system and the Plug and Play kernel subsystem. If Microsoft were afraid that these technologies would be cloned or copied if the code was exposed, then this would allow them to protect the intellectual property associated with those core technologies. This approach would be the opposite of Apple's approach and as such Microsoft would not realize any of the gains that Apple did. One key advantage they would lose would be free development

resources. Since Apple uses an open source core for OS X, it can leverage the work of everyone who contributes to Mach, BSD, and the Darwin project. This means that Apple can maintain a relatively small staff of operating system developers and still ship a very high quality and full featured system.

6.4.3 A Lower Layers Strategy

One the other hand, Microsoft could choose to follow Apple's lead and open source the core of its operating system. This seems less advantageous from Microsoft's perspective because it does not have a best-of-breed user interface to protect like Apple. A quick survey of the major Linux distributions shows that Linux has already successfully cloned the Windows user interface (Horowitz 1). In addition, the markets for most of the functionality provided by these upper layers have been commoditized. The web browser, media player, and email client are all given away for free and Microsoft generates no revenue from these components. However, one possible advantage of this approach would be that some of the core pieces of the kernel that serve as attack surface, such as the networking stack, could be reviewed by 3rd parties for security threats. Whether or not Microsoft would allow external programmers to work on the core of its operating system and whether anyone would want to is debatable. Thus, Microsoft would most likely not realize benefits similar to Apple by choosing this approach.

6.4.4 A Component Based Strategy

Finally, Microsoft could take an even more granular approach and do something similar to what Symbian does with its partner program. Since Windows is divided into functional subsystems, Microsoft could grant or deny access to subsystems depending on the need presented by the party wishing to access the code. For example, one can imagine that an antivirus software maker might be interested in some of the file system code while a firewall vendor would be interested in the networking stack. An academic doing operating system research might be granted access to the kernel itself. Most application developers would probably be interested in the user-mode and executive API layers of the operating system. This system would be similar to the shared source initiative except that it would be much more open. Instead of granting access to only very large institutions, the source would be open to anyone who could demonstrate a need to view it. This could lead to a problem of code sharing among these partners, but Microsoft already has this same problem with the SSI and has taken steps to prevent this type of code sharing. When a portion of the Windows source code appeared on the internet in February of 2004, Microsoft quickly identified the third party that was responsible for the leak (Wagner 1).

6.5 Microsoft's Licensing Model

If Microsoft were to open source Windows, the licensing issue would surely be of great concern to Microsoft. Any license used with Windows would have to be much more restrictive than both the BSD and GPL licenses. As we saw in the section on licensing in this paper, many of the common characteristics of a true open source license would severely damage Microsoft's business model. The derived work characteristic

alone would be enough to stop Microsoft from adopting such a license. It simply does not make sense from a revenue or support perspective for Microsoft to allow derived versions of Windows to exist. Microsoft also currently provides forms of liability and warranty protection with its software license which is explicitly forbidden in a true open source license. It is highly unlikely that Microsoft would give this up as well. In fact, Microsoft has recently announced it will offer indemnification protection to consumers in an attempt to further differentiate itself from Linux (Evers 1). What would an open source license for Windows actually have to look like? Essentially, the license would have to be read-only. Licensees would be granted access to view the source, but would not be granted access to modify or distribute the code in any way. Such a restrictive license would most likely not be deemed to be truly open source by the Open Source Initiative, but that would be of little concern to Microsoft. If Microsoft were to choose to open source Windows, they would make the decision for business reasons. Opening up the system could help to sell more copies of Windows, but allowing others to modify the OS or give it away is definitely not aligned with Microsoft's interests. Thus, any open source style license by Microsoft would have to be very similar to the current license used by the SSI. Does this mean that Microsoft could not open up Windows any more than it is currently? The answer is a definite no. On the contrary, Microsoft could stand to benefit by expanding and improving its shared source stance and we will show how this would be possible in the succeeding discussion.

6.6 The Effects of an Open Source Style Windows License

The earlier section on open source business models gave us some great insight into the many different models a company can use with open source. However, due to Microsoft's market dominance and history, it is highly unlikely the corporation would adopt any of those models in the near future. Microsoft will most likely remain as a company that sells licenses to use its operating system and other software unless markets change drastically and the company is forced to adopt a different model. For the purposes of this discussion, we will assume Microsoft will stick to its core business model in the near future. We should also note that although we use the term "open source" fairly liberally in the following sections, we have not changed our earlier position on licensing issues. An open source license by Microsoft might actually technically be called "shared source" and that should be kept in mind during the discussion.

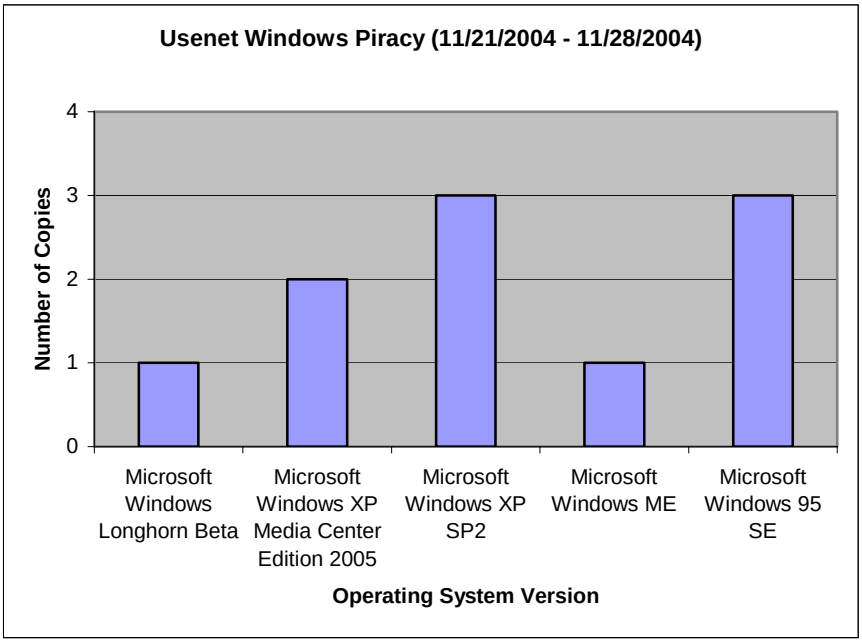
6.6.1 Revenue

After Microsoft released all or some portion of Windows as open source, what would be the outcome? One issue that comes to mind immediately is how such a move would affect Microsoft's revenue. The answer is that it would most likely not affect revenue at all. Microsoft's revenue for its client operating system division in the 2004 fiscal year was $11,546 million ("Microsoft Fourth Quarter 2004", 1). Although the information is not made available publicly, many in the technology press have reported that the "vast majority" of this revenue comes from large OEM licensing (Hruska 1). We feel that it is a conservative estimate that 75% of that revenue is derived from OEM licenses. Given the size of such OEMs and the penalties that would be associated with

violating Microsoft's license, it would be very unlikely that any OEM would use the available source code of Windows to violate any type of source code licensing agreement. This point becomes even more apparent when you consider that the major OEMs already have access to the Windows source through the existing SSI. Open sourcing Windows could lead to more technically savvy consumers using the software illegally, but the effect on Microsoft's bottom line would be miniscule.

6.6.2 Piracy

Another issue that arises when discussing the possibility of open source Windows is that of piracy. Once again, the point proves to be moot as piracy would also most likely not be affected. Even with a closed source operating system, Microsoft has not been able to stop those who wish to pirate Windows from doing so. An examination of a popular Usenet tracking website shows the following versions of Windows being trafficked over a period from November 21 2004 to November 28 2004.



This graph shows that Windows is pirated at will in the United States by those who desire to do so, and the situation is even worse abroad. It is estimated that over 90% of software copies are pirated in China and 36% of software copies are pirated world wide. Microsoft and the software industry lost close to $30 billion in worldwide sales last year due to piracy (GlobalPinoy.com 1). The fact of the matter is that pirates have been extremely successful pirating Windows by bypassing the embedded binary protection schemes and the fact that the source code became available would not change the state of software piracy in the world.

6.6.3 Security

One area where Microsoft could stand to gain a long-term advantage by licensing Windows as open source would be security. Microsoft has made security a top concern

in response to customer demand and today is working internally to increase the security of its products. By making the code available for review by third parties, Microsoft could move further away from its security through obscurity approach and also invalidate the claim by Linux advocates that Linux is more secure because it has more developers examining the code for security defects. On the other hand, licensing Windows as open source could initially create a host of short-term security problems as well. Allowing hackers access to the source code would give them the advantage of finding existing security holes and creating exploits before they could be found by Microsoft or its partners. Microsoft could be forced to release the source code to independent third party experts in security before releasing it to the public in order to have the code reviewed. However, even if Microsoft took such action there would most likely be a period of time after the release of the code where security vulnerabilities were found and exploited at a rate that dwarfs what we see for Windows currently. On the one hand, this seems like it would cause extremely large amounts of damage and financial loss. However, in the grander scheme of the IT world, this could actually have a positive effect by greatly increasing the rate at which security of the system is improved. Enduring a period of intense pain in order to improve the future of security in Windows over the long run could definitely be worth the discomfort.

6.6.4 Competitive Advantage

Another point that must be considered is that of competitive advantage. Would an open source Windows license harm any competitive advantage that Microsoft currently has? In the company's latest executive e-mail, Steve Ballmer identified total cost of ownership, security, and manageability as advantages of Windows over open source alternatives (Ballmer 1). Whether or not the software is open sourced seems to be orthogonal to these issues. These advantages stem more from having an intelligently designed system and listening closely to customers needs than from the fact that Microsoft guards its source code tightly. Another theme of Microsoft's current competitive strategy is that of "integrated innovation" (Wilcox "Microsoft's Integrated Innovation 1). Windows and Office are very mature products and Microsoft can no longer convince customers to upgrade to a newer version simply by adding new features and updating the UI with a fresh look (Wilcox "Office Looks Beyond" 1). To combat this, Microsoft is attempting to integrate many of its key products and to make the system more seamless as a whole. With an open source license, one could argue that Microsoft's competitors would be able to make their products work better with Windows as well, but the fact of the matter is that open source competitors are doing this already. Microsoft's advantage in this space stems more from the fact that is it a corporation with one unified goal of increasing shareholder value and simply because the people working on products like Office and Word are physically closer and more connected with one another. This area may be one of the few areas where the distributed nature of open source software is actually a disadvantage. Microsoft's competitive advantage in these areas would not be harmed by an open source license.

6.6.5 Community

One final issue to address is that of community. Microsoft has always been a company that has stressed the importance of its partner and developer ecosystem

("Microsoft Partner Ecosystem Flourishes", 1) and one of the primary reasons that Windows has been so successful is because of the large number of applications that are developed for the platform. Allowing these partners and developers much greater access to the platform could only server to strengthen this community. Recently, Microsoft has been attempting to strengthen this community in many ways. Corporate webloggers such as Robert Scoble and Raymond Chen have given a human face to the company. Raymond Chen's weblog is particularly interesting because he commonly discusses the architecture and design of the OS and why things are the way they are with his readers. This dialogue could be improved even further by allowing him to reveal even more details about the internals of the OS. Entire teams at Microsoft have also attempted to make their processes much more transparent. The team creating Visual Studio 2005 has opened up its bug database to the public for the first time. Initiatives like this are great, but allowing developers and partners greater access to the source code of the platform would be an even greater stride forward.

We can also look at the larger IT community or even the general public and see advantages to an open source version of Windows. Today, many people in IT and computer science see open software as superior to closed software, either for technical, philosophical, or religious reasons. Microsoft could greatly increase its favor with this group and with the general public as a whole by making such a move. Creating an open source version of Windows could help to alleviate much of the evil associations that people have with Microsoft without harming Microsoft's successful business model.

6.7 Conclusion

Over the past ten years, the open source software movement has greatly affected Microsoft. Microsoft has changed from a company that has attacked open source software to one that has accepted and even adopted some of open source software's philosophies. At the same time, other platform providers have embraced open source even more enthusiastically than Microsoft and today Microsoft is still one of the staunchest supporters of closed source software. While it seems unlikely that Microsoft will adopt a true open source license for Windows, it is clear that the company and the technology industry could benefit from allowing customers, developers, and partners greater access to the Windows source code.

7 Open Source Summary

7.1 Conclusions

In recent years, open source has enjoyed phenomenal growth and has evolved from a hobby to a viable alternative to traditional closed source software development. Many have ventured into the realm of open source with as many failures (VA Linux, Miro) as successes (Linux, Apache, OS X). It is clear that there is no sure-fire way to succeed in the open source business. There are many crucial questions that you need to ask yourself before you start the journey. Why would I want to open source? Do I have a viable open source business model? What is the right open source license that complements my business model? How do I minimize the potential legal risks? We have shown that these questions are answerable and that there is a promising path for those who know how to walk it.

8 References

8.1 Section 1

Novell's OSS shift analysis, a Butler group's opinion wire.
http://www.unisysworld.com/opinionw/art.php/158

Microsoft's shared source initiatives
http://www.microsoft.com/sharedsource

Novell patent policies and guidelines an introduction.
http://www.novell.com/company/policies/patent/

Open source definitions and movement "definition and benefits"
http://www.opensource.org A brief history of open and free source software:
http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html

Analysis of IIS and Apache market shares:
http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2862549,00.html

Survey of top 100 web servers:
http://www.port80software.com/surveys/top1000webservers/

Netcraft survey of web server market:
http://news.netcraft.com/archives/web_server_survey.html

D. Bosio, M. J. Newby titled "Advantages of open source project for reliability:
clarifying the issues"

8.2 Section 2

Fishman, S. (2004). Open Source Licenses Are Not All the Same.
http://www.onlamp.com/pub/a/onlamp/2004/11/18/licenses.html

GeodSoft (2004). Comparing Commercial and Open Source Licenses.
http://geodsoft.com/opinion/LicenseIssues.htm

GNU (2004). Various Licenses and Comments about Them.
http://www.gnu.org/philosophy/license-list.html

Laurent, A. M. S. (2004). The MIT, BSD, Apache, and Academic Free Licenses.
Understanding Open Source and Free Software Licensing, O'Reilly.

LeClair, J. (2003). Commercial Developer's Guide to Open Source Licenses, OpenEnterpriseTrends.com. http://www.oetrends.com/news.php?action=view_record&idnum=224

OSI (2004). Open Source Initiative. http://www.opensource.org

Wikipedia (2004). GNU General Public License From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/GNU_General_Public_License

8.3 Section 3

Dinesh C. Sharma, "IDC: Software sales to hit $189 billion", http://news.com.com, Nov 5, 2004

R. Gomulkiewicz, "De-Bugging Open Source Software Licensing," (http://www.law.washington.edu/Faculty/Gomulkiewicz/Publications/debugOpen Source.pdf) University of Pittsburgh Law Review 64:75 (2002)

R. Gomulkiewicz, "How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B," (http://cyber.law.harvard.edu/is99/Copyleft.htm) 36 Hous. L. Rev. 179 (2002)

Steven J. Vaughan-Nichols, "Copyright Battle Erupts over Open-Source 'Mambo' Code" (http://www.eweek.com/article2/0,1759,1648592,00.asp) September 21, 2004

U.S. Copyright Office, "Copyright Basics", http://www.copyright.gov/circs/circ1.html, December 2004

8.4 Section 4

Hecker.Org. "Setting Up Shop: The Business of Open-Source Software". Available from http://www.hecker.org/writings/setting-up-shop.html. Accessed 1 December 2004.

Microsoft. "Microsoft: Open source threatens our business ". Available from http://www.linuxworld.com/story/32628.htm. Accessed 1 December 2004.

Mozilla. "The Mozilla Project and mozilla.org". Available from http://www.mozilla.org/editorials/mozilla-overview.html. Accessed 8 December 2004.

NASA. "Open Source and NASA's Mars Rover". Available from http://www.onlamp.com/pub/a/onlamp/2004/08/02/oss_mars_rover.html. Accessed 8 December 2004.

OpenSource.Org. “Open Source Case for Business”. Available from http://www.opensource.org/advocacy/case_for_business.php. Accessed 1 December 2004.

Oracle. “Linux Technology Center”. Available from http://www.oracle.com/technology//tech/linux/index.html. Accessed 1 December 2004.

Red Hat. “Red Hat Enterprise Linux Features and Benefits “. Available from http://www.redhat.com/software/rhel/features/. Accessed 1 December 2004.

Stallman. “The GNU Manifesto”. Available from http://www.gnu.org/gnu/manifesto.html. Accessed 1 December 2004.

8.5 Section 5

History of SCO “The SCO v IBM” Caldera. com. Available from SCO Website www.caldera.com

History of SCO “The Power of UNIX” SCO.org. Available from http://www.sco.org/history

Raymond, Eric. “Tradegy to Farce The SCO vs. IBM lawsuit.” Techupdate.zdnet.com Available from http://techupdate.zdnet.com/techupdate/stories Accessed 21 May 2003.

OpenSource.Org. “Open Source Case for Business”. Available from www.opensource.org

8.6 Section 6

Ballmer, Steve. “Customer Focus: Comparing Windows with Linux and UNIX.” Microsoft Executive email, 27 October 2004. Available from http://www.microsoft.com/mscorp/execmail. Accessed 21 November 2004.

Bishop, Todd. “Open source's threat to Microsoft is growing.” *Seattle Post Intelligencer*. 19 November 2003. Newspaper online. Available from http://seattlepi.nwsource.com/business/148915_msftlinux19.html. Accessed 22 November 2004.

Bishop, Todd. “Microsoft Notebook: Open source at Microsoft!” *Seattle Post Intelligencer*. 12 April 2004. Newspaper online. Available from http://seattlepi.nwsource.com/business/168652_msftnotebook12.html. Accessed 22 November 2004.

Bishop, Todd. "Microsoft to open source." *Seattle Post Intelligencer*. 24 June 2004. Newspaper online. Available from http://seattlepi.nwsource.com/business/179256_msftopen25.html. Internet. Accessed 21 November 2004.

Evans, Bob. "Business Technology: Microsoft and Its Blind Spot: Linux." *Information Week*, 1 November 2004. Magazine online. Available from http://www.informationweek.com/story/showArticle.jhtml?articleID=51201701&tid=5999. Accessed 21 November 2004.

Evers, Joris. "Microsoft Offers Users Legal Protection." *PC World*, 10 November 2004. Magazine online. Available from http://www.pcworld.com/news/article/0,aid,118558,00.asp. Accessed 21 November 2004.

Foley, Mary Jo. "FlexWiki: Microsoft's Third Open Source Project." *eWeek*, 28 September 2004. Magazine online. Available from http://www.eweek.com/article2/0,1759,1657278,00.asp. Accessed 23 November 2004.

Galli, Peter. "Sun's Schwartz Opens Up on Solaris Plan." *eWeek*, 12 July 2004. Magazine online. Available from http://www.eweek.com/article2/0,1759,1622378,00.asp. Last accessed 28 November 2004.

GlobalPinoy.com "Microsoft, software makers lose $29 billion to piracy." 9 July 2004. Available from http://www.globalpinoy.com/news/business/07092004/busi5.htm. Accessed 1 December 2004.

Greene, Thomas. "Ballmer: 'Linux is a cancer.'" *The Register*, 2 June 2001. Magazine online. Available from http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer. Accessed 21 November 2004.

Hruska, Joel. "Taking the Time to Get it Right: Why Delaying Longhorn is a Smart Move." *Sudhian Media*, 1 March 2004. Magazine online. Available from http://www.sudhian.com/showdocs.cfm?aid=513. Accessed 30 November 2004.

Jepson, Brian. "Uncovering Rotor -- A Shared Source CLI." *O'Reilly OnDotNet.com*, 4 March 2002. Available from http://www.ondotnet.com/pub/a/dotnet/2002/03/04/rotor.html. Accessed 23 November 2004.

Joyce, Erin. "Microsoft, Open Source Claim XML Success." *InternetNews.com*, 26 November 2003. Available from http://www.internetnews.com/ent-news/article.php/3114271. Accessed 21 November 2004.

Kiat, Ong Boon. "Sun warms to open source for Solaris." *Cnet News Asia*, 2 June 2004. Magazine online. Available from http://news.com.com/Sun+warms+to+open+source+for+Solaris/2100-7344_3-5224473.html?tag=nl. Accessed 28 November 2004.

Krazit, Tom. "Symbian Updates Smartphone OS." *PCWorld*, 25 February 2004. Magazine online. Available from http://www.pcworld.com/news/article/0,aid,114932,00.asp. Accessed 28 November 2004.

LaMonica, Martin. "Open Source Solaris to debut this year." *CNet News*, 13 September 2004. Magazine online. Available from http://news.com.com/'Open+Source+Solaris'+to+debut+this+year/2100-7344_3-5364052.html. Accessed 28 November 2004.

Legard, David. "Microsoft: Open source threatens our business model." *Computerworld*, 5 February 2003. Magazine online. Available from http://www.computerworld.com/softwaretopics/software/story/0,10801,78203,00.html. Accessed 21 November 2004.

Lin, Allison. "Microsoft opens door to Office." *Seattle Times.* 20 September 2004. Newspaper online. http://seattletimes.nwsource.com/html/businesstechnology/2002040559_microsoft20.html. Accessed 22 November 2004.

McMillan, Robert. "Solaris steals Linux's clothes." *Computer Weekly*, 17 November 2004. Magazine online. Available from http://www.computerweekly.com/articles/article.asp?liArticleID=135131&liArticleTypeID=1&liCategoryID=6&liChannelID=9&liFlavourID=1&sSearch=&nPage=1. Accessed 22 November 2004.

Microsoft Corporation. "Microsoft Approach to Source Code Sharing Balances Accessibility with Responsibility." 3 May 2001. Press Release online. Available from http://www.microsoft.com/presspass/features/2001/may01/05-03csm.asp. Accessed 23 November 2004.

Microsoft Corporation. "European Ministry Provided with Microsoft Source Code for the First Time." 3 December 2001. Press Release online. Available from http://www.microsoft.com/presspass/press/2001/dec01/12-03SharedSourcePR.asp. Accessed 23 November 2004.

Microsoft Corporation. "Microsoft Partner Ecosystem Flourishes." 24 May 2004. Press Release online. Available from http://www.microsoft.com/presspass/press/2004/may04/05-24EcosystemFlourishesPR.asp. Accessed 1 December 2004.

Microsoft Corporation. "Microsoft Fourth Quarter 2004 Earnings Report." 22 July 2004. Press Release online. Available from http://www.microsoft.com/msft/earnings/FY04/earn_rel_q4_04.mspx. Accessed 30 November 2004.

Mundie, Craig. "Prepared Text of Remarks on The Commercial Software Model at The New York University Stern School of Business." Microsoft.com. 2001 May 3. Available from http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp. Accessed 21 November 2004.

OpenDarwin.org. "OpenDarwin Developers." Website online. Available at http://www.opendarwin.org/en/developers.html. Accessed 30 November 2004.

Raymond, Eric. "The Halloween Documents." OpenSource.org. Available from http://www.opensource.org/halloween. Accessed 21 November 2004.

Pope, Justin. "Microsoft may face upheaval in open-source policy." *Seattle Post Intelligencer*. 20 October 2003. Newspaper online. Available from http://seattlepi.nwsource.com/business/144617_source20.html. Accessed 22 November 2004.

Ricciuti, Mike. "Gates wades into open-source debate." *CNet News*, 19 June 2001. Magazine online. Available from http://news.com.com/2100-1001-268667.html?legacy=cnet. Accessed 21 November 2004.

Ricciuti, Mike. "Open source: Rebels at the gate." *CNet News*, 14 October 2002. Magazine online. Available from http://news.com.com/2009-1001-961354.html. Accessed 21 November 2004.

Shankland, Stephen. "Sun plans patent protection for open-source Solaris." *CNet News*, 18 November 2004. Magazine online. Available from http://news.com.com/Sun+plans+patent+protection+for+open-source+Solaris/2100-7344_3-5456451.html?tag=nefd.lede. Accessed 22 November 2004.

Story, Derrick. "O'Reilly Network: Mac OS X Opens Apple to a New Audience." *Linux Today*, 31 Jan 2001. Magazine online. Available from http://linuxtoday.com/news_story.php3?ltsn=2001-01-13-012-06-NW-SW&tbovrmode=1. Accessed 30 November 2004.

Wagner, Jim. "Suspect Charged in Connection to MS Windows Leak." *InternetNews*, 9 November 2004. Magazine online. Available from http://www.internetnews.com/bus-news/article.php/3433581. Accessed 1 December 2004.

West, Joel. "How open is open enough? Melding proprietary and open source platform strategies." *Research Policy 32.* 2003.

Wilcox, Joe, Alorie Gilbert and Mike Ricciuti. "Open source closes in on Microsoft." *ZDNet News,* 14 October 2002. Magazine online. Available from http://news.zdnet.com/2100-3513_22-961903.html Accessed 21 November 2004.

Wilcox, Joe. "Microsoft's Integrated Innovation: Weighing up Customer Benefits, Risks." *Jupiter Research Concept Report,* 6 October 2003. Available from http://www.jupiterresearch.com/bin/item.pl/research:concept/1093/id=94567. Accessed 1 December 2004.

Wilcox, Joe. "Office Looks Beyond Good-Enough." *Microsoft Monitor,* 12 July 2004. Available from http://www.microsoftmonitor.com/archives/003326.html. Accessed 1 December 2004.

9 Appendix

9.1 The GPL License

The GNU General Public License (GPL)
Version 2, June 1991

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Preamble
The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.
To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.
Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.
The precise terms and conditions for copying, distribution and modification follow.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term

"modification".) Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) 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.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.
3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software

interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under any particular circumstance,

the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.
NO WARRANTY
11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR

INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

9.2 The BSD License

The BSD License
The following is a BSD license template. To generate your own license, change the values of OWNER, ORGANIZATION and YEAR from their original values as given here, and substitute your own.
Note: The advertising clause in the license appearing on BSD Unix files was officially rescinded by the Director of the Office of Technology Licensing of the University of California on July 22 1999. He states that clause 3 is "hereby deleted in its entirety."
Note the new BSD license is thus equivalent to the MIT License, except for the no-endorsement final clause.
<OWNER> = Regents of the University of California
<ORGANIZATION> = University of California, Berkeley
<YEAR> = 1998
In the original BSD license, both occurrences of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".
Here is the license template:
Copyright (c) <YEAR>, <OWNER>
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Microsoft's shared source initiatives: http://www.microsoft.com/sharedsource

10 Contributions

10.1 Division of Labor

We organized our team around our overall topic of "open source software from a commercial perspective." Each team member proposed a sub-topic section of the paper that they would like to work on. We mutually agreed on these sections and the paper organization. Each group member posted a summary of their section to the project Wiki. These were formatted into our project proposal.

During the writing of the sections, we collaborated mostly through email and in person and our use of the Wiki fell off greatly after we had submitted the proposal. Email was more efficient to communicate updates, suggestions and have conversations over – because many of the group members monitor their mailbox continuously at work and home.

Prior to the rough draft deadline, each group member emailed their section rough draft to the group. The rough drafts were formatted into a single draft paper and turned in by the deadline.

Group members and Tapin Parikh made comments on the rough draft sections and emailed these to the group. Each group member made final changes to their section rough drafts and submitted final drafts to the group by email. An introduction and conclusion were authored among group members via email collaboration. The introduction, final section submissions, and conclusion were formatted into the final paper.

The division of labor was as follows:

Abstract - James
Introduction - Bipin
1. Open Source Definition, Benefits, History, and Alternatives - Bipin
2. Open Source Licensing - Rodrick
3. Commercial Software and Open Source Coexistence - Patrick (also wrote 1.1)
4. Open Source from a Business Point of View - Song
5. Open Source Legal Issues - Magdalene
6. Case Study: Microsoft Windows and Open Source Platforms - James
7. Conclusion - Rodrick, Song, James

Group Scheduling/Coordination - Bipin
Proposal Formatting/Editing - Rodrick
Rough Draft Formatting/Editing - James
Rough Draft Comments/Suggestions- Patrick, James, Rodrick
Final Draft Formatting/Editing - Patrick